IMT School for Advanced Studies, Lucca

Lucca, Italy

Supervised and Semi-Supervised Learning in Vision using Deep Neural Networks

PhD Program in Institutions, Markets And Technologies

Curriculum in Computer Science

XXX Cycle

By

Soumali Roychowdhury

2019

The dissertation of Soumali Roychowdhury is approved.

Program Coordinator: Prof. Rocco De Nicola, IMT School for Advanced Studies, Lucca

Supervisor: Prof. Michelangelo Diligenti, Università De Siena

Supervisor: Prof. Rocco De Nicola, IMT School for Advanced Studies, Lucca

The dissertation of Soumali Roychowdhury has been reviewed by:

Prof. Marco Lippi, Dipartimento di Scienze e Metodi dellà Ingegneria, Universita reggio emilia e modena

Prof. Michele Lombardi, Department of Computer Science and Engineering, Università di Bologna

IMT School for Advanced Studies, Lucca

2019

I dedicate this thesis to my parents, and to my beloved pets for their unconditional love. I love you all dearly.

Contents

List of Figures

List of Tables

Acknowledgements

There are many people that have earned my gratitude for their contribution in my PhD studies. More specifically, I would like to thank six groups of people, without whom this thesis would not have been possible: my thesis supervisors, my PhD colleagues, my industrial collaborators, my funding institution, my family and my pets.

First, I am indebted to my PhD advisors, Professor Michelangelo Diligenti and Professor Rocco De Nicola, for supporting me throughout during the past four and half years. I worked closely with Prof. Michelangelo and he is my mentor, teacher and guide. It all started in Spring 2015 when he agreed to supervise me and it was my honor to get the great opportunity to work with him and his other PhD students. On the academic level, Prof. Michelangelo taught me the fundamentals of conducting a scientific research in the area of machine learning. Under his supervision, I learned how to understand a research problem in depth, and then try to find a solution to it, and finally publish the results. His suggestions have always been invaluable for me. He has always been very lively, enthusiastic, and energetic. On a personal level, he inspired me by his hardworking and passionate attitude. He has been supporting me through all my thick and thins. He was very instrumental in helping me shape up all my publications and also the thesis. I am looking forward to continue working with him even after my PhD studies. He is also a very nice person to talk to and we always have very insightful discussions about different kinds of topics starting from research to travel to weather.

My other advisor, Prof. Rocco has always been my support system in IMT School of Advanced Studies. He encouraged me at every step and have always given me the freedom to pursue various projects without any objection. I am very grateful to him for all his scientific advices as well. I am

also thankful to the head of Artificial Intelligence group of the Università De Siena, Professor Marco Gori. Prof. Marco has been the person who actually inspired all of us with his idea of using constraints in deep learning which is the core of this research work. He will always remain as my role model for a scientist.

I also like to thank the members of my PhD committee, Professor Marco Lippi and Professor Michele Lombardi for all their suggestions and comments on my thesis. Their feedbacks have helped me to better connect all the different parts of the thesis and design a smooth workflow.

I would like to thank my labmates in the Artficial Intelligence lab of Università De Siena for their support whenever I needed. I would like to specially mention Francesco Giannini who was so kind that he once presented a paper on my behalf when I could not attend a conference. Also this thesis, would not have been possible without the intellectual contribution of all the members of that lab who often gave me suggestions and comments on the experiments. I would also like to express my gratitude to my other PhD colleagues at IMT School of Advanced Studies without whom I would not have survived in a country whose language, people and culture were all new to me. They became my close friends in the past few years and they all made me feel as if I know them for years. I will always cherish the times spent with them as the best memories of my life.

I am also grateful to my industrial collaborators, D-Wave Technologies in Canada. The projects I worked on during my internship formed a significant part of my thesis. I spent a year working with them on two different projects and had a great learning experience. There I had the chance to collaborate with fantastic researchers in deep learning and computer vision. More specifically, I would like to thank Dr. Arash Vahdat for his great mentoring, continuous support and all his guidance.

I also want to extend my love and thanks to all my childhood friends (the list is too long!) for their lifelong friendship as they have been wonderful throughout my life. They always

made me smile despite any situation. Sometimes, I miss them a lot as I am miles away from them.

I also owe my PhD thesis to IMT foundation for providing me the financial support for pursuing my PhD dream. I thank all the administrative staffs of the IMT community without whose collaborative efforts, timely reminders and follow up schedules, I would not be able to attend the conferences, summer schools or internships. I was able to secure travel grants in the timely manner, get all the documentations ready for every research period I spent outside IMT because of the wonderful support I received from them.

Last but not least, I would like to express my deepest gratitude to my family and relatives and love to my pets. Although I am far away from them, I remember them each and every day and miss them terribly. This dissertation would not have been possible without their warm love, continued patience, and endless support. Every single morning my mom and dad remembered to call or text me just to ask if I am doing good. Their unconditional love and support is something that always makes me strong. My PhD is as much their dream as it is mine. I also express my regards and care for my partner, Abhi who often at times helped me with his jokes to ease the pressure off, accompanied me during my summer school and internship.I know I always have my family to count on when times are rough. I also want to express my love to my beloved cat, Shonai, who has passed on. I still miss her, especially because of the 11 long years of friendship that I shared with her while growing up. Her playful activities made me happy even at the most difficult days. I wished she was still with me. Lastly, I want to mention about my present pet, Shonai's daughter Mushi whom I also love a lot and pray for her good health and well being every single day. I also really wish that she grows up as brave as her mother.

Vita

February 13, 1987	Born, Durgapur, India
2009	Bachelor degree in Computer Science Engineering Final mark: 8.96/10 West Bengal University of Technology, India
2013	Masters in Cybernetics and Artificial Intelligence Distinction University of Reading England, UK
2019	Currently pursuing PhD in Computer Science IMT - School for Advanced Studies Lucca Working in collaboration with Università De Siena D-Wave Technologies, Canada

Publications

1. Integrating Prior Knowledge into Deep Learning, ICMLA 2017: 920-923.
2. Image Classification Using Deep Learning and Prior Knowledge, AAAI Workshops 2018: 336-343.
3. CATARACTS : Challenge on Automatic Tool Annotation for Cataracts Surgery, Medical Image Analysis, Volume 52, February 2019, Pages 24-41.
4. Regularizing Deep Networks with Prior Knowledge: A Constraint-based Approach, submitted to IEEE Transactions of Neural Networks and Learning Systems, TNNLS-2019-P-11577.
5. Improving Image and Video Classification using Prior Knowledge, Submitted in ICMLA, 2019.
6. Instance Segmentation approaches from Contaminant Separation in Cleanliness Analysis, Submitted in ICMLA, 2019.

Research Internships

1. DeepLearn 2017 Summer School, Bilbao, Spain (July, 2017).
2. Summer Internship in D-Wave Technologies Inc., Vancouver, Canada (August, 2017 - November, 2017).
3. Winter Internship in D-Wave Technologies Inc., Vancouver, Canada (December,2017 - May, 2018).
4. Poster Presentation in 1st IMT Research Symposium, Lucca, Italy (July, 2018).

Abstract

Deep learning has been a huge success in different vision tasks like classification, object detection, segmentation etc., allowing to start from raw pixels to integrated deep neural models. This thesis aims to solve some of these vision problems using several deep neural network architectures in different ways. The first and the core part of this thesis focuses on a learning framework that extends the previous work on Semantic Based Regularization (SBR) to integrate prior knowledge into deep learners. Deep neural networks are empirical learners and therefore heavily depend on labeled examples, whereas knowledge based learners on the other hand are not very efficient in solving complex vision problems. Therefore, SBR is designed as a semi-supervised framework that can tightly integrate empirical learners with any available background knowledge to get the advantages of learning from both perception and reasoning/knowledge. The framework is learner agnostic and any learning machinery can be used. In the earlier works of SBR, kernel machines or shallow networks were used as learners. The approach of the problem, concept of using multi-task logic functions are borrowed form the previous works of SBR. But for the first time, in this research work, the integration of logic constraints is done with deep neural networks. The thesis defines a novel back propagation schema for optimization of deep neural networks in SBR and also uses several heuristics to integrate convex and concave logic constraints into the deep learners. It also focuses on extensive experimental evaluations performed on multiple image classification datasets to show how the integration of the prior knowledge in deep learners can be used to boost the accuracy of several neural architectures over their individual counterparts. SBR is also used in a video classification problem to automatically annotate surgical and non-surgical tools from videos of cataracts surgery. This framework achieves a high accuracy

compared to the human annotators and the state-of-the-art *DResSys* by enforcing temporal consistency among the consecutive video frames using prior knowledge in deep neural networks through collective classification during the inference time. *DResSys*, an ensemble of deep convolutional neural networks and a Markov Random Field based framework (CNN-MRF) is used, whereas SBR replaces the MRF graph with logical constraints for enforcing a regularization in the temporal domain. Therefore, SBR and DResSys, two deep learning based frameworks discussed in this thesis, are able to distill prior knowledge into deep neural networks and hence become useful tools for decision support during interoperative cataract surgeries, in report generation, in surgical training etc. Therefore, the first part of the thesis designs scientific frameworks that enable exploiting the wealth of domain knowledge and integrate it with deep convolutional neural networks for solving many real world vision problems and can be used in several industrial applications. In the present world, a range of different businesses possess huge databases with visuals which are difficult to manage and make use of. Since they may not have an effective method to make sense of all the visual data, it might end up uncategorized and useless. If a visual database does not contain meta data about the images or videos, categorizing it, is a huge hassle. Classification of images and videos through useful domain information using these unified frameworks like SBR is a key solution. The second part of the thesis focuses on another vision problem of image segmentation and this part of the thesis is more application-specific. However, it can still be viewed as utilizing some universal and basic domain knowledge techniques with deep learning models. It designs two deep learning based frameworks and makes a head to head comparison of the two approaches in terms of speed, efficiency and cost. The frameworks are built for automatic segmentation and classification of contaminants for cleanliness analysis in automobile, aerospace or manufacturing industries. The frameworks are designed to meet the foremost industry requirement of having an end-to-end solution that is cheap, reliable, fast and accurate in comparison to the traditional techniques presently used in the contami-

nant analysis and quality control process. These end-to-end solutions when integrated with the simple optical acquisition systems, will help in replacing the expensive slow systems presently existing in the market.

Chapter 1

Introduction

1.1 Motivation of the thesis

The title of the thesis, *Supervised and Semi-Supervised Learning in Vision using Deep Neural Networks*, gives a glimpse of the motivation of this research work. It exploits the power of the deep neural networks in different vision tasks using supervised and semi-supervised learning techniques. From a broad perspective, the thesis focuses on the integration of domain specific knowledge with deep learning. Deep learning has been a breakthrough in tasks like classification, object detection and segmentation of images and videos, starting from raw pixels, learning complex feature representations and developing trained neural models in an integrated manner. Integration of domain knowledge in deep learning has several practical applications in the field of computer and machine vision. A range of different industries in different domains from healthcare to security, possess huge databases with visual content which are difficult to use in a meaningful way without an effective method to make sense of all the visual information. Therefore, most of the times the information remain uncategorized and useless. If a visual database does not contain meta-data about the images or videos in it, categorizing or labeling it is an expensive and tedious job. Classification and segmentation of images and videos through domain knowledge integrated with deep learning is an effective solution to this problem. It will become possible to easily organize and categorize the visual database because it will allow for automatic classification and segmentation of images and videos

in large quantities. This will help the industry leaders to monetize their visual content without investing countless hours for manual sorting, annotating and tagging. This thesis therefore explores some of these vision problems and uses different deep neural networks and domain knowledge to solve them. The thesis can be divided into two parts:

- The first part of the thesis discusses about the supervised and the semi-supervised frameworks for different image and video classification tasks. It focuses on a learning framework known as Semantic Based Regularization (SBR) to integrate prior knowledge into deep learners. Suppose a deep convolution network is to be used for a classification problem, it always needs to learn from a lot of labeled examples. Acquiring huge amounts of data is time consuming and therefore, using of any available background knowledge to train the learner is an efficient alternative as mentioned above. Injecting prior knowledge into a learner provides any available high-level knowledge like class dependencies to the learner. SBR, is therefore a semi-supervised framework that can integrate empirical learners like kernel machines or deep convolutional neural networks with prior knowledge to get the advantages of learning from both perception and reasoning. Learning in SBR is formulated as a multi-task learning problem and the prior knowledge is expressed as a collection of first-order logic clauses (FOL), where each task to be learned corresponds to a predicate in the knowledge base. Each knowledge statement is then translated into a constraint which can be integrated into the learning process or used during inference to enforce the consistency of the predictions of the deep learners. This methodology is learner agnostic and therefore, any learning architecture can be used. Previously in the literature, the SBR framework used kernel machines as the learning machinery. Kernel machines are shallow networks and hence the feature learning capability is limited. The definition of the SBR framework defined in this thesis, the formulation of the multi-task learning problem are equivalent to the prior works of SBR. But in this research work, for the first time several deep convolutional neural networks (rich feature learners) is integrated in the SBR framework. The theoretical foundation of SBR is extended for joint training of neural networks with logical constraints. The thesis defines a novel and efficient back-propagation schema, which computes gradient over the expression tree of any arbitrary grounded FOL expression and

aims to seamlessly integrate prior knowledge into deep learners. It also highlights the plans and heuristics of using non-convex logical constraints in SBR. SBR is also able to enforce regularization during the test time using collective classification on the outputs of the deep learners and the thesis aims to experimentally verify the benefits of collective classification through different types of simulations. The extensive experimental evaluations of this thesis use SBR on multiple image classification datasets and a video classification dataset to show how the integration of the prior knowledge can help to boost the accuracy of several deep architectures over the networks trained without knowledge in different classification problems. This work for the first time makes a detailed study with a large variety of deep CNN networks on large scale image datasets like CIFAR-100 and ImageNet having numerous output classes to learn. SBR is also used to solve a video classification problem, where it competes with the state-of-the-art Markov Random Field based framework working on the predictions of an ensemble of deep neural networks to establish temporal consistency among video frames. The task is to accurately annotate tools from videos of cataracts surgery for decision support during interoperative surgeries or in report generation etc. This research work for the first time aims to apply collective classification using the SBR framework on neural network predictions to enforce a regularization for establishing temporal coherence. This work aims to make a significant contribution in the field of automation of tool detection during surgeries using deep learning. Therefore, Chapter 3, Chapter 5 and Chapter 6 constitutes the first part of the thesis as described and they aim to seamlessly combine domain knowledge with deep neural nets in different vision applications through a general and fundamental approach, the Semantic Based Regularization.

- The second part of the thesis focuses on image segmentation tasks using deep convolutional neural networks. It can still be viewed as using domain specific techniques (for example: watershed transform and connected component analysis algorithms) that are integrated with neural networks in a cascaded fashion resulting in the development of an end-to-end solution for industry based contaminant separation problem. In this research work, for the first time, deep learning solutions are proposed for automatic segmen-

tation and classification of contaminants for cleanliness analysis and quality control in automobile, aerospace or manufacturing industries. It adapts and uses several deep learning architectures like Mask RCNN and UNET in building end-to-end segmentation frameworks to solve the contaminant separation task. These frameworks are designed with the aim to build end-to-end solutions for these industries so as to replace the multi-stage expensive slow systems presently existing in these industries. This part of the thesis is more application oriented. However, theoretically, it also highlights the novel techniques like improved region selection methodology in Mask RCNN, a superior objective function in UNET, integrating domain specific post-processing techniques with the deep learning frameworks etc. that aims to meet the challenges faced in adapting the existing frameworks to the current problem. Chapter 7 constitutes the second part of the thesis which aims to design frameworks for solving the specific contaminant separation but not limited to it.

Hence, both the parts of the thesis aim to integrate some domain knowledge with deep learning for different types of vision tasks. The first part introduces a fundamental and scientific approach whereas the second part is a more application oriented solution.

1.2 Overview of the thesis

The thesis is structured into the following chapters as listed below:

- Chapter 2 includes a review of the literature of logic based learning systems, empirical learning systems and hybrid systems that integrate logic and learning.

- Chapter 3 includes a detailed description and the mathematical formulations of the Semantic Based Regularization (SBR) with deep neural networks as its learning machinery. The chapter also discusses about different fuzzy logics, joint training and optimization schema in SBR, the differences in constant and vanilla SBR, the collective classification mechanism in SBR and convexity of constraints using different fuzzy logics.

- Chapter 4 includes a review of different deep neural networks and machine learning techniques used for the vision tasks for this research work. There are two types of frameworks, individual deep CNN architectures like Residual networks, Inception networks etc. used in image/video classification tasks and also the segmentation frameworks like Mask RCNN and UNET used for image segmentation. This chapter also describes the methods like semantic and instance segmentation with examples.

- Chapter 5 includes the different experimental simulations on image classification task using the Semantic Based Regularization framework on Winston Animal, CIFAR-10, CIFAR-100 and ImageNet datasets.

- Chapter 6 describes a video sequencing problem for cataracts tool annotation during cataracts surgery. It reports the experimental simulations performed with an end-to-end framework, DResSys, which is a combination of a CNN ensemble and a probabilistic Markov Random Field (MRF) graph. This framework is compared with its counterpart where the Markov Random Field graph is replaced by the Semantic Based Regularization to enforce a mechanism of regularization on the outputs of the test set in order to establish temporal consistencies among video frames.

- Chapter 7 discusses an industrial problem of identifying particle contamination, measuring them and aiding their automatic removal for cleanliness analysis in automobile, aerospace or manufacturing industries etc. Two competing end-to-end deep learning based approaches for object detection and segmentation are proposed namely a Mask RCNN based approach with modified region proposal selection criteria and a UNET based approach with efficient post processing to separate instances of the same class.

- Chapter 8 summarizes the contributions of this research work and also examines the pathway towards future works.

- The Appendix A contains the detailed experimental results (the ones that could not be included in the Chapter 5 due to limited space) of image classification tasks on Winston Animal, CIFAR-10, CIFAR-100 and ImageNet datasets using different deep CNN architectures integrated in constant and vanilla SBR with different

fuzzy logics. The Appendix A also contains per-tool basis AUC-ROC scores of the cataracts tool annotations using different experimental designs of the DResSys and SBR framework (Chapter 6).

- The Appendix B tabulates the different configuration parameters and methods used in training the image segmentation frameworks like Mask RCNN and UNET in Chapter 7.

Chapter 2

Integration of Logic and Learning

In todays' world of research, efforts have been continuously made towards the development of artificially intelligent systems that can replace human labor successfully in many contexts to perform jobs faster and with higher accuracy. Designing artificially intelligent systems require modeling the laws of human psychology and cognitive learning abilities so that they can mimic how humans learn. Humans have two different approaches to learn new things, known as inductive and deductive learning.

The systems that learn about an object or about a situation from examples without actually having any particular explanation why the examples belong to that category of the object are referred to as non-symbolic, sub-symbolic learners (Ben09; LeC15), perception based (DXYZ18) learners or empirical learners (TS94). They are data hungry systems that require intensive amount of labeled data or empirical examples, and the availability of lot of data generally give them a competitive advantage over the deductive learners in many applications. Artificial deep neural networks (DNN) are examples of sub-symbolic learners that have attracted the most attention in artificial intelligence (AI) community in the last few years. DNNs have achieved extraordinary performance in recognizing human faces (TYW14), objects (KSH12a; STE13) and in speech recognition (HDY$^+$12).

On the other hand, in a deductive learning approach, the system

learns from domain theory, prior knowledge or from a set of structured logic rules that explain why a particular object belongs to a definite category. Deductive learners are actually symbolic reasoners (RKT07; KBB+12; Mug) and are typically based on logical and probabilistic inference, allowing to perform high-level reasoning (possibly under uncertainty) without having to deal with thousands of hyper-parameters. Logic based AI systems have achieved human-level abilities in proving mathematical theorems (NS56; CL97) and also in performing inductive reasoning using relations (Mug) or logical information.

Since, the human mind flawlessly incorporates both these sources of information to fill the gaps in the knowledge acquired, it is necessary that if an artificially intelligent system mimicking the human mind is to be designed, a synergy between both the phases of learning needs to be established. There is a widespread belief that the ability to represent and reason about structured objects and structure-sensitive processes is crucial for any learner, and logic learning is well-suited in achieving this objective. On the other hand, empirical or sub-symbolic learners have properties which are not easily found in logic based systems such as, the ability to learn from perceived information, noise robustness, the ability to adapt to new environments and the ability to have enhanced memories. They are also more tractable and faster compared to their logic counterparts. However, in the recent works in deep learning, the sub-symbolic approaches are mostly seen as *black-boxes*, whereas symbolic approaches are generally easier to interpret and understand as long as they do not contain recursive and complex rules. In modern AI, both approaches co-exist with different levels of advantages and disadvantages. Both approaches assume to deal with complete and correct knowledge about the problem to be solved. But for many real world applications, completeness and correctness are extremely difficult to achieve. The advantages in both of these types of learners should be exploited to the fullest to design a good artificially intelligent system. Therefore, building hybrid learning models where perception and reasoning are glued together, is one of the most challenging and an open problem in AI and, recently, a lot of works, often referred to as neuro-symbolic approaches (GLG08), have been proposed. It is crucial to see how the symbolic and sub-symbolic approaches affect each other in a single system and how advantages can be gained from both.

2.1 Related Works

The connections between logic and learning have been studied extensively in different systems in the last few decades. In the early years, most emphasis had been given to hybrid models, where perceptual and logic information was handled separately in different modules, whereas in the recent years, the focus has shifted towards achieving a truly tight integration between the two modules.

2.1.1 Symbolic Learning Methods

In symbolic learning, Statistical Relational Learning methods have been very popular. The field of Statistical Relational Learning (SRL) provides a large set of methods that integrate learning and logic like Markov Logic Networks (MLN) (RD06b; GT07; KD05; RD06a; DR04; WD08) and Probabilistic Soft Logic (PSL) (KBB$^+$12; BMG10). Markov Logic Network combine first-order logic and probabilistic graphical models. In MLN, a weight is attached to each FOL formula that actually reflects how strong a constraint is. Therefore, it can be defined as L which is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number together with a set of constants or datapoints $C = \{c_1, c_2, ...c_{|C|}\}$ that can be viewed as the template for constructing the Markov Network $M_{L,C}$. An example of MLN graph is given in Figure 1 that shows some grounded logic with two grounded datapoints A and B and some arbitrary predicates like $Smokes, Friends, Cancer$ etc. The first order logic rules corresponding to this MLN can be $\forall(x)Smokes(x) \rightarrow Cancer(x)$, $\forall(x)(\neg(\exists yFriends(x, y) \rightarrow Smokes(x))$ etc.

With a different set of constants, it will produce a different network of different size. Each state of $M_{L,C}$ represents a possible interpretation where each state has its own set of objects, relations that hold between these objects and truth values of the grounded relations. MLNs are defined as log linear models or products of potential functions. MLN employs a differentiable approximation of FOL and allows to learn the weight of each formula in the KB by maximizing the log likelihood or the pseudo log likelihood of the training data from one or more relational databases. MLNs modularly incorporates a wide range of domain knowledge, it can learn complex feature representations from the data and soundly handle uncertainty. MLNs can tackle many SRL problems like collective classification, link prediction, object identification etc.

Hybrid Markov Logic Networks (HMLNs) (WD08) extend MLNs to

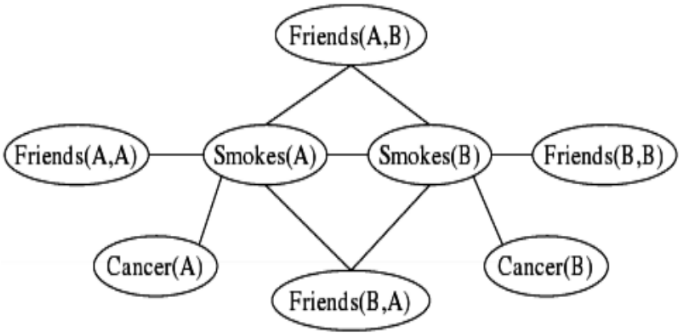


Figure 1: Markov Logic Network representing some arbitrary predicates on two datapoints that can be written in the form of first order logic rules (RD06b).

deal with continuous variables or fuzzy first-order logic clauses. Knowledge Based Model Construction (KBMC) is a combination of logic programming and Bayesian networks (KDR02) where given a Horn clause in the knowledge base, a KBMC answers a query by finding all the possible backward chaining proofs of the query and evidence atoms from each other using a Bayesian network over all atoms in the proofs and then performing an inference over this network. Actually a KBMC model can be treated as a special case of Markov Logic Networks. Similarly, Probabilistic Relational Models (PRM) (Kol99) are a combination of frame-based systems and Bayesian networks that can also be converted into MLNs by defining a predicate for each attribute of each class. Stochastic Logic Programmings (SLP) is also a special case of MLNs that is a combination of logic programming and log linear models as described in (Mug96).

Probabilistic Soft Logic (PSL), a probabilistic logic framework like hybrid MLNs, is relaxed to continuous fuzzy values in the range of $[0, 1]$ and considers only the FOL formulas with a conjunctive body and a single literal head. Furthermore, the rule weights are restricted to only non-negative values that capture the rule's relative importance. PSL use an undirected graphical model to represent a grounded FOL knowledge base, and restricted the logic that can be processed to a fragment of FOL corresponding to convex constraints. It builds a joint probabilistic model over all its grounded FOL clauses and because of the soft truth values, inference in PSL is a continuous optimization problem which can be solved efficiently.

All these frameworks and many more like Relational Dependency Networks (RLN) (NJ07), Structural Logistic Regression (SLR) (PPU03) and Relational Markov Networks (RMN) (TAWK) that are architecturally similar to MLNs are more focused on logic reasoning without any integration with deep learners. They are required to deal with large number of weights and groundings in order to learn a complex model.

2.1.2 Integration of Logic and Learning : Hybrid Learning Methods

Injecting symbolic knowledge into sub-symbolic learning via neural networks or kernel machines is not a new approach, there have been several such hybrid learning systems designed since 1990. KBANN (Knowledge Based Artificial Neural Network) (TS94) proposed by Towell et al., encoded symbolic rules into a neural network layer during network initialization, which is later refined using a back-propagation schema. KBANN is an example of hybrid learning system which is built with the motivation that the system should be able to use the information provided by one source to offset the information missing from the other source. In KBANN, the symbolic rules have a Prolog (Bra14) like notation. The work flow of KBANN (also represented diagrammatically in Figure 2) with the corresponding inputs and outputs are listed as :

- Given: A list of features used to describe examples, an approximately correct domain theory in the form of propositional logic (specifically Horn clauses) describing the problem to be solved and a set of classified training examples.
- The domain theory is translated into a neural network and this algorithm is called as the Rules-to-Network. This tries to effectively insert approximately correct symbolic rules into the neural net.
- The neural network is then trained with the inserted knowledge and external empirical training examples. A neural learning algorithm is used for training the network.
- The trained network or the trained classifier is used to classify any future example during the inference time.

The main disadvantage of KBANN is that, once the network is initialized, the symbolic knowledge layer is totally frozen not allowing the two

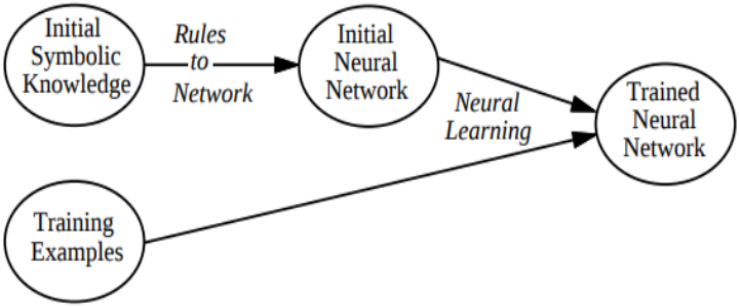


Figure 2: Knowledge flow in KBANN (TS94).

layers to mutually improve the results of each other, therefore limiting the generalization capabilities of KBANN. Also the rules that are used in KBANN have several restrictions. When KBANN was developed, it was difficult to train artificial neural networks due to limited hardware capabilities and very high overhead costs.

In the works of Hitzler et al. (HHS04) in the year 2004, integration of logic with neural networks have been demonstrated. They call this as the connectionist system where certain semantic operators for propositional logic programs are computed by feed forward neural networks, in a way similar to KBANN.

However, in both of the above mentioned works, perceptual and logic information are mostly handled in independent modules. The logic layer is always frozen when learning is performed in the neural networks. This limitation arises because of the barriers between the classical mathematical models that handle real numbers and the ones that handle logic reasoning. The knowledge improvement and the neural network weight updates are two mutually exclusive modules and no joint learning takes place.

There also have been several attempts in literature to integrate shallow networks like kernel machines with inductive logic programming. In the works of (Hau99), convolutional kernels are built on infinite sets whose elements are discrete structures like strings, trees and graphs. The different family of kernels that can be generated include radial basis kernels, kernels defined on joint Gibbs probability distributions, kernels built from hidden Markov fields, regular expressions and ANOVA decompositions.

Muggleton et al. (MLAS05) proposed a general method for constructing kernels for Support Vector Inductive Logic Programming (SVILP). The kernel captures the semantic and syntactic relational information contained in the data as well as provides the flexibility of using arbitrary forms of structured and non-structured data coded in a relational way. The SVILP approach is a form of generalization relative to background knowledge, although the final combining function for the learned clauses is a support vector machine rather than a logical conjunction.

Landwehr et al. (LPDRF06) introduced a very well known inductive logic programming system combined with kernel machines known as kFOIL, based on dynamic propositionalization where structure learning correspond to inferring a suitable kernel on logical objects, and parameter learning correspond to function learning in the resulting reproducing Kernel Hilbert space.

All these approaches incorporated structures expressed in different forms into kernels but most of the integrations does not reveal tight connections. These approaches are also not suitable for multi-task learning environments.

Another approach that imposes constraints in the perceptual space was introduced in the studies of Fung et al. (FMS02). Knowledge Based Support Vector Machine Classifiers seamlessly injects prior knowledge in the form of polyhedral knowledge sets (that results in building convex constraints) in the input space into a linear support vector machine classifier. The modeled constraints represent the knowledge base quite extensively and they have been applied to many real-world applications like DNA sequencing and breast cancer prognosis. However, the experimental results are not as good as a neural network based system like KBANN (TS94)(architecture described earlier). Although, this approach shows a simplified injection of logic into linear inequalities outperforming purely logic based systems but they are not equipped to solve multi-task learning using prior knowledge.

On the other hand, the initial works on Semantic Based Regularization (SBR) (DGMR12) attempted to inject logic constraints into kernel machines for multi-task classification problems by integrating perceptual data and prior knowledge. SBR introduced the concept of a constraint, which is also sufficiently general enough to represent different kinds of sensorial data along with their relations and express abstract knowledge on the tasks. The tight connections of logic and learning in SBR evolved from the preliminary studies on FOL constraints and kernel machines given in the works of Diligenti et al. (DGMR10b; DGMR10c; DGMR10a).

SBR encodes a multi-layer architecture using kernel machines at the input layer, which provides input to the higher-level layers implementing a fuzzy generalization of the FOL knowledge. Since, the fuzzy generalization helps in defining real valued constraints, the information can flow down to the lower layers and hence the whole framework can be jointly trained and optimized. This actually helps SBR to preserve the compactness and efficiency of the kernel machines to deal with the feature space representations at the input level, while also exploiting the full expressive power of the FOL describing the higher level semantics.

Another predominant work referred to as Feature Description Logic (FDL), described by Cumby et al. (CR03a; CR03b), which is actually a relational language with clarity in syntax and semantics that can be easily used, via feature generation functions, to efficiently represent world observations in a way that is suitable for any general purpose learning algorithm. It has been shown that FDL allows to efficiently learn complex relational representations where the size of the representations to be learned are very large.

All the shallow and single layered neural architectures responsible for transforming the raw input signals or features into problem specific feature space, are effective in solving only the well constrained problems. They have limited modeling and representational power both in terms of features extracted from inputs and information learnt from these inputs. Therefore, they are incapable of solving more complicated real world applications like those involving natural images or visual scenes effectively. Also the information processing mechanisms in the human brain suggest the need of deep architectures for extracting complex features and building internal representation from rich sensory inputs.

2.1.3 Integration of Logic into Deep Learners

Deep Learning refers to the class of machine learning techniques, developed largely since 2006 (LeC15; Ben09), where many stages of non-linear information processing in hierarchical architectures are exploited for classification and recognition problems. A deep learning network is composed of simple but non-linear modules and each of them transform the representation from one level (for example starting with raw input pixels in case of images) into a representation at a higher level, a slightly more abstract level. Deep learning lays at the intersection among the research areas of neural networks, graphical modeling, optimization, pattern recognition and signal processing. The popularity of deep learning

nowadays is mainly due to the availability of enormous amount of data, increased chip processing abilities (example: GPU units), lower costs of computational hardware and a big machine learning community actively participating in information processing research.

One of the most traditional yet successful deep architecture is the Convolutional Neural Network (CNN) (DYP12) with each module consisting of alternate convolution and pooling layer having different filter sizes to build complex feature maps from the inputs. Many such modules are stacked one above the other, followed by a few fully connected layers, a loss function and a classifier layer thus forming a deep convolutional neural network. A diagrammatic representation of a deep CNN framework is shown in Figure 3.

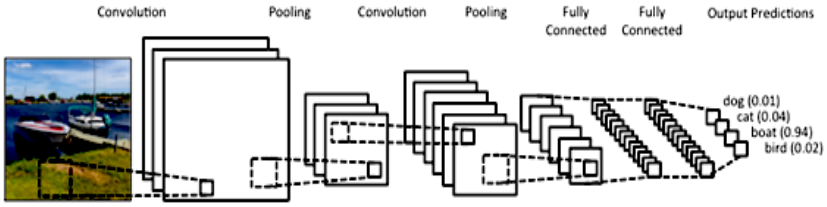


Figure 3: Alternating convolution and pooling layers with fully connected and classifier layers at the end, typically forms a deep CNN model (LeC15).

Deep CNNs have been found to be very effective in complex pattern recognition tasks like image and video classification, keypoint detection, image segmentation etc. Deep CNNs, like most other deep neural architectures, are commonly trained using a large number of labeled data points also known as supervised examples. When the datasets are mostly unsupervised, this data can only be used to drive a proper initialization of the weights in a pre-training phase (EBC$^+$10; BLP$^+$07). Since these deep learners are sub-symbolic or empirical learners, the development of powerful feature representations in these learners are still considered to be very challenging without abundant supervised data. Also, in spite of extensive studies in this field, the process is opaque and it is not clear as how much amount of data is required for training any deep neural network. Unfortunately, it is difficult and labor intensive to manually annotate huge datasets in the era of Big data. Therefore, to convert the empirical success of deep learning into a scalable solution, prior knowledge can be incorporated into the learners along with scarce amount of

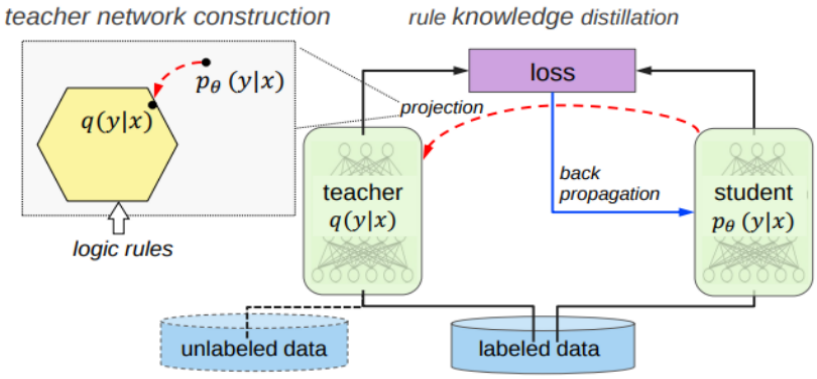


Figure 4: The teacher network is iteratively obtained by projecting the student network to a rule regularized subspace (HML$^+$16).

labeled examples with respect to the number of classes or categories to be learnt. Injecting prior knowledge in the learning framework can potentially allow to find better models by restricting the space where the learner should search its optimal parameters. This mode of learning is possibly more aligned to how the human brain learns, where high level knowledge and low level sensory inputs play a joint role in learning. After the popularity of deep learning, several works in this area have emerged with specific applications and unified frameworks that integrate logic and learning using deep neural networks.

The works of Hu et al. (HML$^+$16) described an iterative distillation method that transfers structured information of first-order logic rules into the weights of the neural networks. This is achieved by forcing the network (called as the student network) to emulate the predictions of a rule-regularized teacher network, and evolving both networks iteratively throughout the training. The teacher network is constructed at each iteration by adapting to the posterior regularization principle in the logical constraint setting, where the formulation is a closed-form solution with low increase in the computational overhead. The entire framework is agnostic to the type of neural architecture that can be used. The trained network has been deployed to perform classification tasks like sentiment analysis and sequence learning tasks like name entity recognition. The diagrammatic representation from the original works is shown in Figure 4 where the neural network is defined as the conditional prob-

ability $p_\theta(y|x)$, a prediction vector parameterized by weights θ with inputs x and outputs y. A rule regularized projection of $p_\theta(y|x)$ is denoted as $q(y|x)$ which explicitly includes the rule constraints as regularization terms.

Logic Tensor Networks (LTN) (SDdG17) is another similar framework that integrated logic reasoning and learning based on fuzzy generalization of FOL. LTNs have been applied on the task of semantic image interpretation that extracts structured semantic description from images. LTNs use a deep tensor network that allow learning from noisy data in the presence of logical constraints and reasoning with logical formulas that describe the general properties of the input data. A LTN is defined as a first-order logic language L, composed of three disjoint sets denoting constants, functions and predicates. An example of the logical formula can be $\forall x(Cat(x) \rightarrow \exists y(partOf(y,x) \wedge Tail(y)))$ meaning that every cat should have a tail and exceptions are handled as these formulas are expressed in fuzzy FOL. Suppose, there is an image of a tailless cat, then the FOL formula can be interpreted as every cat should normally have a tail. In this formula, $Cat(x), Tail(y)$ are predicates, the fuzzy generalization of these predicates are the formulas and x, y are the constants or the objects of the domain. Here, the outputs of the neural network predictions are used as grounding values on which the logical constraints are applied and thus the deep network is trained. Inference performed on these logical tensor networks allows approximate reasoning on unseen data to predict new facts.

There are also other independent studies of logic programming with artificial neural networks applied to specific applications like breast cancer detection (Nev15), short text classification (WWZY17) etc.

All the above proposed solutions considers the reasoning layer as frozen, without allowing to jointly train its parameters. Because of this, it can work better only with hard constraints, while being less suitable in presence of reasoning under uncertainty. They are also limited in the kind of knowledge that can be integrated like usage of only quantified variables mostly universally quantified variables, having a small set of logic operators etc. Furthermore, in these networks the knowledge can be injected only at training time but it can not be enforced during classification.

Another application of semantic knowledge and learning is Semantic Sensitive Tensor Factorization (SSTF) described in the works of Nakatsuji et al. (Nak16) that incorporates the semantics expressed by an object vocabulary or taxonomy to factorize any tensor.

Minervini et al. (MR18) proposed to use prior knowledge in Natural Language Processing (NLP) to correct the inconsistencies of an adversarial example generator. They investigated the problem of automatically generating adversarial examples that violate a set of given first-order logic constraints in natural language inference. All these applications show the immense power of prior knowledge that can be exploited to improve the performance of any supervised learning system in a semi-supervised manner.

Integration of logic is also shown in probabilistic frameworks like TensorLog (Coh16). In TensorLog, each logic clause in the domain theory is converted into a certain type of factor graph. Each logical variable present in the clause gets associated with a random variable in the factor graph. Then, for each type of unknown variable to the factor graph, the message passing steps that are required to perform Belief Propagation (BP) are unrolled into a differentiable function. Each function answers for a particular combination of known and unknown variables in the factor graph. This way Tensorlog develops a probabilistic deductive database, such that both compilation as well as inference modes are efficient. In Tensorlog, compilation is linear in the size of the domain theory and proof depth, whereas inference is linear in size of the deductive database and the message passing steps used in BP. Tensorlog has its roots in some variants of Markov Logic Networks like Stochastic Logic Programs (SLP) (Mug96). However, TensorLog is limited to reasoning with less developed semantics and does not allow to optimize the learners while performing inference. The integration of the deep learners and the knowledge base is not efficiently tight.

One other recent work, that involves a probabilistic framework with logic constraints is called as Deep ProbLog (MDK$^+$18) that extends the popular ProbLog (RKT07; RFKE08), a probabilistic programming framework. ProbLog is an efficient solver that defines a distribution over logic programs by specifying for each clause the probability that it belongs to a randomly sampled program, and these probabilities are mutually independent. In a ProbLog program, there is a set of grounded probabilistic facts and a set of rules. Deep ProbLog extends this framework to deep learners and hence combines domain knowledge specified as probabilistic facts and rules with the outputs of neural networks. The parameters of the probabilistic facts and the neural networks are jointly trained in Deep ProbLog. Given a Deep ProbLog program, its neural network models, and a query used as a training example, first the program is grounded with respect to the query and then learning is per-

formed using ProbLog (RKT07) machinery to calculate the gradient of the loss. Following this, the parameters of probabilistic logic programs and the neural network weights are updated after each optimization step performed using standard gradient descent (whereas in ProbLog optimization, it is done via Expectation Maximization or maximizing the log likelihood). Deep ProbLog makes the assumption that each probabilistic fact corresponds to an independent Boolean random variable (a mutual independence assumption also made in ProbLog) to make the inference tractable. This assumption limits the capability of both of these frameworks as this is a strong restriction, since the sub-symbolic layer often consists of several neural network layers sharing the weights.

Another most prominent recent line of related work focuses on developing differentiable framework for logical reasoning. It is a Prolog based architecture introduced for theorem proving by Rocktäschel et al. (RR16; MBRR18) known as Neural Theorem Prover (NTP). NTP is differentiable with respect to symbol representations in a knowledge base (KB) and therefore, can learn any representations of predicates, constants and also first-order logic rules of predefined structure using backpropagation. NTPs use the backward chaining algorithm of Prolog to find substitutions of free variables with constants of facts in a KB. During training, it iterates over a set of known facts and optimizes the negative log-likelihood of the success of proof of every fact based on all the other facts. This has a huge limitation and does not scale to larger KB as it investigates all possible proofs. A NTP encodes relations as vectors using a frozen pre-selected function (like cosine similarity). This can therefore be ineffective in modeling relations of complex and multifaceted nature (for example a relation `friend(A,B)` can be triggered by different relationships of the representations in the embedding space). However, most of the models discussed above requires to fully ground a KB (like SBR, LTN, PSL etc.), while NTPs expands only the groundings on the explored frontier, which can be more efficient in some cases. As they operate on the distributed representations of symbols, a single handcrafted rule can be leveraged for many proofs of queries with similar symbolic representations.

An idea similar in spirit to Deep ProbLog and Neural Theorem Provers but on a different task of visual question answering with neural networks is addressed by Andreas et al. (ARDK15). In this work, each question is decomposed into their linguistic sub-structure and these structures are used to dynamically instantiate and learn neural module networks. All these modular networks are jointly trained for question answering. The

schematic diagram in Figure 5 shows a natural image input to the CNN that does object recognition to identify the *dog*, then there is also a natural language parser to answer questions about the linguistic sub-structure of the query like *where is the dog?*. Rather, than relying on a single deep network architecture, this approach assembles a deep network on the fly from a collection of specialized, jointly learned modules where each module is responsible for an individual task.

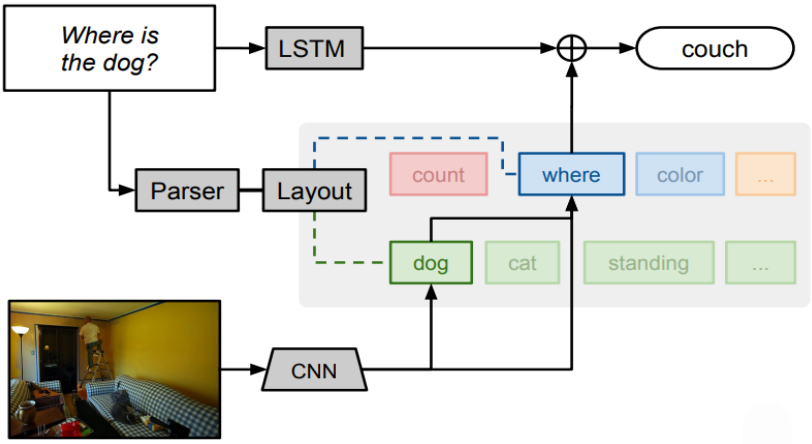


Figure 5: A schematic representation of the Andreas et al. Deep Compositional Question Answering Framework (ARDK15).

In the recent study of Xu et al. (XZF+18), a method is introduced that involves a semantic loss function aimed to bridge the gap between neural learning and logical constraints. This paper attempted to solve multi-task learning problem which is also the main motivation of Semantic Based Regularization (DGMR12). In the works of Xu et al., the semantic knowledge is in the form of a logical constraint in Boolean logic. These constraints can simply represent one-hot output encodings, or a more complex structured output prediction constraint for combinatorial objects such as rankings, subgraphs and paths. The semantic knowledge is applied to the outputs of the neural network through the semantic loss function. It has been pointed out in their work that to preserve the precise logical meaning of the rule, instead of making it continuous by using fuzzy generalization, the differential semantic loss function is introduced to capture the information of how well the outputs of the neu-

ral networks match a given constraint. The semantic loss is defined as $L_s = (\alpha, p)$, a function of a constraint α in propositional logic, over variables $x = x_1, \ldots, x_n$, and a vector of probabilities p for the same variables x where element p_i denotes the predicted probability of variable x_i, and corresponds to an output of the neural net. Semantic loss is added as another regularization term that can be directly added to the supervised loss to get the overall loss. This work is quite interesting when the output space is simple but for multi-class classification problems, they use circuit compilation techniques (Dar11) to build a Boolean circuit representing the semantic loss which seems to have a higher complexity and overhead compared to the benefits obtained through integration of semantic knowledge in deep learners. Also, this work is not able to apply constraints during the inference time and their experimental simulator is only restricted to solve classification problems in AI.

A slightly different way of combining perception (sub-symbolic learning) and reasoning (symbolic learning) have been seen in the recent work of Dai et al. (DXYZ18) through a process called as Abductive Learning. Abductive learning phenomena is close to human learning ability that involves both perception and reasoning capability. When a differentiable neural perception model (i.e a neural network) is coupled with a non differentiable logical abduction module, learning becomes extremely difficult and to this solution, Dai et al. used a heuristic based trial and error search approach. The neural network involved in perception learning outputs symbols that aim to make symbolic hypothesis (formed by joining symbols as first-order logic rules) consistent with each other. However, during training when the hypothesis is inconsistent, the logical abduction module finds the incorrect output from the neural perception module and corrects it. The neural network module and the abduction module which are implemented as the learning machinery in this work is known as Neural-Logical Machine (NLM). The NLM can exploit symbolic knowledge while processing sub-symbolic data even in the form of raw pixel images. This work is actually very similar to the Deep Prolog framework which deals with uncertainty of neural network's output with probabilistic reasoning. In NLM, a revision of hypothesis is done iteratively replacing the neural network's output with anonymous variables until a consistent hypothesis can be developed.

The integration of deep learning with Conditional Random Fields (CRFs) (MH16) is also an alternative approach to enforce some structure on the network output developed by Ma et al. This approach has been proved to be quite successful on sequence labeling for natural language

processing tasks. Ma et al. built an end-to-end model with a combination of different networks. At first, they have a CNN that encodes character-level information of a word into its character-level representation. Then they combine character and word-level representations and feed them into a bi-directional LSTM for modeling contextual information of each word. Finally, there is a sequential CRF to jointly decode labels for the whole sentence. This end-to-end network actually integrates large amounts of task-specific knowledge and empirical data in a single system.

Deep Structured Models (LSRvdH15; CSYU14) use a graphical model to bridge sensorial and semantic information. These models extend deep learners to learn complex feature representations considering the dependencies between the output random variables. In the work of Chen et al. (CSYU14), a learning algorithm is proposed that builds a structured model with arbitrary graphs along with deep features that together form the potentials of a Markov Random Field (MRF). They showed the effectiveness of these models in image tagging tasks whereas Lin et al. (LSRvdH15) applied deep structured models in semantic image segmentation tasks. However, both of these works are mainly focused on imposing correlations or dependencies on the output random variables, without any focus on logic reasoning.

Each of these frameworks have limitations either in their scope of usage or in the diversity of logic reasoning that can be integrated into the learners or in the joint training capability of the frameworks or in enforcing consistency of the prior knowledge during the inference. The main motivation of the defining Semantic Based Regularization for deep learners as described in the next section is to overcome all these limitations.

2.2 Semantic Based Regularization : Motivation

As mentioned, Semantic Based Regularization (DGMR12; DGS15) is a Statistical Relational Learning framework, that integrates the perception and reasoning modules in a hybrid learning system. It is able to learn from data examples and logic rules. Prior knowledge in SBR is expressed via a set of FOL clauses and then relaxed into their continuous fuzzy representations. The clauses can express different aspects of the learning task like relationships among the patterns or among the classes or can provide a partial definition of the mapping between the input and the output.

Most of the previous literature on Semantic Based Regularization has focused on Kernel Machines to implement the functions. SBR finds its roots and inspirations in the works like Markov Logic Networks (RD06b; LF09), Hidden Markov Logic Networks (WD08), ProbLog (RKT07) etc. described in Section 2.1. SBR is also related to KBANN (TS94) as the rules in SBR can be hierarchically structured and in this research work it uses neural networks as the learning machinery similar to KBANN. In both of these frameworks, many of the rules are used for providing intermediate conclusions. These intermediate conclusions are further used to determine final conclusions or other intermediate conclusions during learning. The motivation of using neural networks in SBR, is the richness of complex features that can be extracted by the deep neural networks from raw data. For the first time, a multilayer perceptron was integrated in the SBR framework in the works of Diligenti et al. (DGS16). Although, the concepts of constraints in SBR, the multi-layer architecture and the definition of multi-task functions to integrate logic rules in SBR is identical in this research work to the previous works of SBR, but in the present work for the first time, SBR is integrated with several benchmark deep learners where it can learn complex feature representations as well as learn any type of logical reasoning in the form of independent task functions that are jointly trained in this hybrid architecture using a novel and efficient back-propagation schema. This actually helps SBR to preserve the compactness and efficiency of the deep neural networks to extract rich and complex feature representations from the input, while also exploiting the full expressive power of the FOL rules describing the higher level knowledge. This framework is also structurally similar to Logic Tensor Networks (SDdG17) and uses building blocks like constants, functions, predicates and formulas from first-order propositional logic. However, this framework has a competitive advantage over LTN as it can also apply the constraints during inference. In this thesis, it is shown for the first time that SBR has the flexibility of integrating concave constraints during learning with the help of different optimization heuristics. This makes SBR scalable and effective for many real world industrial problems.

This thesis will show the application of SBR to image classification and video classification tasks using benchmark deep learners to demonstrate the power of the SBR framework in integration of prior knowledge into these learners for vision tasks. The focus is on classification as in the recent years, not only has deep learning achieved new levels of performance on classification (KSH12b) but also classifying images and videos

is one such area where annotating huge datasets is an expensive and tedious work. Image and video classification can be found at the core of everything from Facebook's photo tagging to self-driving cars. Classification and recognition problems are behind the scenes in everything from healthcare to security. But, to solve these problems and train supervised neural networks, the quality of annotations available for most of the image and video datasets are poor and noisy. As, for the deep neural networks, the main limitations include heavy dependency of the predictive accuracy on the labeled data, uninterpretable and counter intuitive results of deep models and also the difficulty to encode human intentions (SLJ+15; NYC14) for guiding models during learning without direct supervision or ad-hoc initialization. These limitations can be addressed by integrating logic rules into DNNs and transferring human intention and domain knowledge into neural models to regulate the learning process. Also, there have been several different types of CNN architectures designed by the machine learning community to target image and video classification for different applications, therefore, using SBR framework and demonstrating its effectiveness with these learners is a big contribution to the community. This research work, exploits the power of SBR in dealing with classification problems using small and large scale datasets and also training very deep neural networks efficiently with logical constraints. Although the focus is on classification tasks, the SBR framework is flexible enough to be extended in future to other machine learning tasks as well.

Chapter 3

Semantic Semi-Supervised Learning

In this chapter the Semantic Based Regularization (SBR) framework, an unified approach to semi-supervised learning from constraints is described in details.

Semantic Based Regularization, a Statistical Relational Learning framework, that integrates the ability to learn from supervised examples and data distribution with the exploitation of high level background knowledge, that are an abstract and partial representation of the environment. Prior knowledge is expressed as a collection of First-Order Logic (FOL) clauses that are converted into real-valued constraints and applied on any learner so that it learns in a semi-supervised manner. In SBR, the constraints can also be applied during the test phase as a collective classification method of any learning problem that enforces the consistency of predictions of the learned model using prior knowledge. The SBR methodology is learner agnostic. It can integrate both shallow or deep learners. In the earlier works, kernel machines was used as the mathematical apparatus during learning whereas in this research work deep neural networks are used for the first time as mentioned in the previous chapters. The motivation of using deep learners as the learning apparatus in SBR is explained in Section 2.2. The novelty of this research work lies in the integration of deep learners into SBR framework. This integration introduces several challenges in terms of training and optimization of the neural networks, in terms of the type of constraints to be added etc.

The methodologies to address these challenges in a meaningful way are the contributions of this research work that are described in this chapter.

SBR enforces a multi-task learning scheme, where each task to be learned corresponds to a statement in the knowledge base. It provides a very tight integration of learning and logic. This framework can be used in vast range of machine learning applications like classification, generative or adversarial machine learning, collective classification etc. As already said, it imposes no restrictions on the type of learners to be used or on the type of knowledge base that can be integrated. Efficiently integrating deep learners with any explicit knowledge about the task at hand, actually eliminates the necessity of labeling huge amounts of data which is required for training any deep learning framework. Therefore, this semi-supervised framework integrated with deep learners opens new doors of practical applications. Some of the examples are mentioned in Section 2.2.

Although framework design of SBR helps in bridging abstract descriptions of the environment with collection of supervised and semi-supervised examples, but learning is still hard in SBR, and depending on the complexity of the added constraints, the function to be optimized may not be convex. Hence classical optimization approaches cannot be used. This chapter also describes the conditions under which SBR training becomes easy, how SBR training can be made end-to-end with seamless flow of information in either directions, how a large set of knowledge can be expressed in SBR as a set of convex constraints and how optimization strategies and heuristics can be used to achieve good results in SBR when integrated with deep learners. Although the definition of SBR and its multi-layer architecture is identical to the previous works in the literature, but training with neural networks using an efficient back propagation schema and all the optimization strategies used are novel contributions of this research work.

3.1 Definitions and Terminologies

1. *Pattern* : defined as an individual or an element of the dataset. It can be a raw input or a pre-processed feature vector. Patterns are constants and are the inputs to the SBR system. For example: A 50×50 pixel image of a *dog* can be a pattern called $p0$.

2. *Variable* : are defined as holders for patterns. They can range over

patterns of a corresponding type and can take any value. For example: x is a variable of image type that can hold any 50×50 pixel image $p0, p1, \ldots$ which can be an image of a *dog* or a *cat* etc.

3. *Domain* : determines a collection of individuals that share the same space of representation and thus can be analyzed and manipulated in a homogeneous way. The domains are filled with individual patterns on which learning and reasoning are carried out. Each domain contains patterns of a particular type. For example: A domain can collect a set of considered 50×50 pixel images, or the sentences of a book as bag-of-words.

4. *Term* : defined as any expression that represents an object in the domain. A term can consist of a pattern, a variable or a function applied to a couple of terms. For example: $p0$, x, $ID(x)$ are all terms where $p0$ is a pattern, x is a variable and ID is a function that tries to identify the category of the object.

5. *Grounding* : defined as a term containing no free variables. When truth values are assigned to the variables, or to the functions containing variables, it is considered to be grounded. For example: $ID(x = p0)$ is a ground term but $ID(x) + ID(x = p0)$ is not a ground term.

6. *Predicate* : defined as a function that maps or grounds the elements of the input domains or the patterns to the truth values. So the function ID used above is actually a predicate. Predicates are also called as atoms. There are two types of predicates :

   - *LEARN* or query or unknown.
   - *GIVEN* or evidence or known.

   LEARN predicate must be estimated during the learning process whereas GIVEN predicate is already known for all of its groundings.

7. *Arity*: defined as the number of arguments a predicate takes. Predicates can take any number of arguments, but for each predicate the number is fixed. For example: if A is an unary predicate, it can always take only one argument.

8. *Example*: defined as grounding of a predicate or function with the patterns in the domain. For example: *ID* can be a predicate that is grounded using the patterns *p0, p1* in the domain of *images*. The grounded predicates are $ID(p0) = 1$ and $ID(p1) = 0$ which form the two examples.

9. *Function* : is equivalent to predicate but is actually a real-valued manifestation of a predicate. It maps elements from the input domain into elements of an output domain. A unary function takes as input an element from a domain and transforms it into an element of the same or another domain whereas a n-ary function takes as input *n* elements, mapping them into an element of its output domain.

10. *Rule* : is any first-order logic clause in the Prenex Normal Form (PNF) with universal and existential quantifiers. Rules are also known as Formulas. Rules express the state of knowledge about the tasks to be solved in the learning problem. The atoms of the rules are predicates. For example: If there is a domain of animal *images*, and there are two predicates $BIRD$ and FLY defined on it, the knowledge can be expressed as *all the birds should fly* using a rule $\forall x, BIRD(x) \rightarrow FLY(x)$.

11. *Connective* : used to define relationships between predicates. The logical connectives are of different types: $\wedge$ for weak conjunction, $\vee$ for weak disjunction, $\otimes$ for strong conjunction, $\oplus$ for strong disjunction, $\rightarrow$ for implication and $\neg$ for negation.

12. *Quantifier* : defined as a statement that expresses some property of a FOL formula and is true for some or all choices that could be made for the formula. There are two types of quantifiers : universal quantifier and existential quantifier. For example : In $\forall x$, $\forall$ is the universal quantifier that says *for any choice of x*, the following is true. In $\exists x$, $\exists$ is the existential quantifier that says *for some choice of x*, the following is true.

13. *Atomic Formula or atom* : is just an alias of predicate defined as building blocks of rules or formulas. Each FOL formula is recursively constructed using logical connectives and quantifiers. Each atomic formula is therefore a predicate applied to the objects in the domain. For example : In the rule $\forall x, BIRD(x) \rightarrow FLY(x)$,

$BIRD(x)$ and $FLY(x)$ are referred to as atomic formula or atoms or predicates.

14. *Literal* : defined as an atomic formula (atom) or its negation in first-order logic. Literals are of two forms : positive literal which is an atomic formula and a negative literal which is the negation of the formula.

15. *Constraint* : defined as a continuous real valued conversion of a logical rule using t-norms (described in details in Section 3.3). The atoms of the constraints are the functions.

3.2 First-Order Logic

First-order logic is also known as predicate logic. It is actually composed of sentences or FOL formulas that uses quantified variables and logical connectives over objects or elements of the data. These sentences are formed using predicates or atoms as defined in Section 3.1. While propositional logic deals with only simple declarative propositions, first-order logic additionally covers predicates and quantifications.

The adjective *first-order* distinguishes first-order logic from higher-order logic where there are predicates having predicates or functions as arguments, or in which one or both of the predicate quantifiers or function quantifiers are permitted.

In first-order logic, it is often more convenient to deal with formulas in which all the quantifiers have been moved to the front of the expression known as the Prenex Normal Form (PNF). A formula is in prenex normal form if it is of the form $(Q_1, Q_2 \ldots, Q_n B)$ where $Q_i (i = 1, \ldots n)$ are $\forall$ or $\exists$ and the formula B is quantifier free. The string $Q_1, Q_2 \ldots Q_n$ is called the prefix and B is called the matrix. A formula with no quantifiers is regarded as a trivial case of a prenex normal form. Clausal form is a subset of first-order logic which is defined as a normal form in which a FOL formula has an universal prefix (a string of universal quantifiers) and a matrix (a quantifier-free conjunction of a clause). The clausal form of first-order logic is also called as the Conjunctive Normal Form (CNF). There is a more restrictive form of first-order logic called as Horn clauses, which are clauses (a disjunction of literals) with at most one-positive literal. Horn clause was first pointed out by Alfred Horn in 1951 (Hor51).

Fuzzy logic allows the formulas of first-order logic to have continuous values which makes first-order logic effective for mathematical opti-

mization. The term fuzzy logic was coined by L.A. Zadeh in 1965 (Zad65). Classical first-order logic and propositional logic deal with boolean values or truth values but fuzzy logic, is in the form of probabilistic logic, where variables have a truth degree that ranges in $[0, 1]$ such that, 0 stands for *False* with certainty and 1 is *True* with certainty. Novak (Nov87) first proposed fuzzy generalization of first-order logic. In Semantic Based Regularization, logical constraints are expressed using fuzzy first-order logic.

3.3 Learning with Constraints

The learning framework of Semantic Based Regularization (DGMR12; DGS15) considers a multi-task learning problem where each task works on an input domain of labeled and unlabeled data examples. Tasks can be n-ary relations where the input is a tuple of patterns. Each input pattern is described by a tuple $\mathcal{X}_k = x_k | x_k \in D_k, k = 1, \ldots n$ which are vectors of features and D_k is the input domain of the values of the k-th predicate.

Let $\boldsymbol{f} = \{f_1, \ldots, f_T\}$ indicate T multi-variate functions such that f_k is the function implementing the task k. It is associated to the set $\mathcal{X}_k$, which is a set of pattern inputs to the function with domain D_k (as described above). The vector of values obtained by applying the function f_k to the set of patterns $\mathcal{X}_k$ is indicated as $f_k(\mathcal{X}_k) = f_k([x_{(1,k)} \ldots x_{(n,k)}])$, while $\boldsymbol{f}(\mathcal{X}) = f_1(\mathcal{X}_1) \cup f_2(\mathcal{X}_2) \cup \ldots$ collects the groundings for all the functions. From a set of T functions that must be learned to solve the tasks, some are known a priori. (evidence or GIVEN functions) whereas others are unknown functions (query or LEARN functions) for the tasks.

When the T task functions have to meet a set of constraints that can be expressed by functionals, $\phi_h : f_1, f_2 \ldots f_T \rightarrow [0, 1]$, then the set of H functionals are provided such that $f_1 \in H_1, \ldots f_T \in H_T$, where $\phi_h(\boldsymbol{f}), 0 \leq \phi_h(\boldsymbol{f}) \leq 1, h = 1, \ldots H$ describes the prior knowledge about the learning task. These functionals define how the query functions should behave as they force the function values to meet a set of first-order logic constraints. These functionals can express properties of a single function or can correlate the subset of $(P|P \in T)$ functions to be learned. Learning can be helped by exploiting these correlations, which limit the parameter space where good solutions can be found and, potentially, allows to learn with less examples. The function spaces H_k are specific for each function since the function domains are generally different from each other.

Each of the task function expressed using the corresponding functionals can share the same sample of patterns (e.g. $X_j = X_i, i \neq j$). Although, the explanation provided in this chapter is limited to unary predicates (detailed explanations of n-ary predicates are avoided for simplicity), but the framework can be trivially extended to predicates of any arity, expressing relations across multiple patterns. In the latter case, the pattern representations associated to these functions are expressed as the combination of the patterns from a set of finite domains: $X_j = X_{j1} \times X_{j2} \times \ldots$.

When the functions are grounded a priori. over a set of available patterns, the groundings form the set of examples $\mathcal{E}_k \subset \mathcal{X}_k$ provided as the supervised labeled data for the learning task at hand. The learning task is formulated as a constrained optimization or a risk minimization problem following the classical penalty approach. Therefore, the overall cost function is given in Equation 3.1, as the summation of the three terms.

$$C_e[\boldsymbol{f}(\mathcal{X})] = \sum_{k=1}^{T} \Big(\overbrace{||f_k||^2}^{Reg} + \overbrace{\lambda_l \sum_{x \in \mathcal{E}_k} L(f_k(x), y_k(x))}^{Labeled} \Big) + \overbrace{\sum_{h=1}^{H} \lambda_h L_c \left(\Phi_h\big(\boldsymbol{f}(\mathcal{X})\big)\right)}^{Logic} \tag{3.1}$$

The first term is a regularization term penalizing the non-smooth solutions, $\mathcal{E}_k$ is the set of labeled data examples ($\subset \mathcal{X}_k$) available for the k-th function, $L(,)$ is the loss function, $y_k(x)$ is the target output value for the pattern x for task k, λ_l is the weight of the labeled portion of the cost function, $L_c()$ is the loss function used for the constraint part and λ_h is the weight for the h-th constraint. A higher value of λ_h makes it more costly not respecting the constraint, and the constraint becomes hard as $\lambda_h \to \infty$. Thus the cost function enforces the constraints by penalizing their violation on the sample of data and also forces the fitting of the supervised data for each function.

There are two versions of SBR implemented in this research work based on the values of weight of the labeled portion (λ_l). This is one of the new addition to the SBR framework which was never before implemented in the SBR literature with kernel machines.

- *constant weight based SBR* : The λ_l of the labeled portion is constant and unified over all the predicates or functions of the learning task.
- *vanilla or variable weight based SBR* : The λ_l is function based where different functions are assigned different weights denoted as λ_l^k.

The weights assigned to the functions are proportional to the outputs of the learner that learns only from supervisions.

In this constraint minimization problem, the completely supervised examples are likely to carry little information since the task constraints are already expressed in the provided supervisions that are supposed to be consistent with the given rules. However, the use of fully labeled examples when exploited to enforce constraints may yield some benefits when the labels are affected by noise. For example: Flickr is a large database of images tagged by travelers and photographers but often the labels are found to be noisy, mainly due to the collaborative tagging activities of the users. In these cases, enforcing constraints on the fully labeled learning process, can be beneficial in handling the errors that may occur with the learning model due to the presence of the noisy labels.

Also, the functionals $\Phi_h(\boldsymbol{f})$ implementing the constraints involve all the values computed by the functions in $\boldsymbol{f}$ on their whole domains. It may be complex to provide a closed form that can be efficiently dealt with in the training process. Hence, it is assumed that these functionals can be conveniently approximated by considering an appropriate sampling in the function domains. In particular, the exact constraint functionals are to be replaced by their approximations that exploit only the values of the unknown functions.

Therefore, the learning problem can be better casted in a semi- supervised framework where it is assumed that a set of partially labeled examples are exploited together with an usually larger set of unlabeled examples. In particular, the choice of unsupervised examples can be exploited in order to maximize the information available in the joint knowledge of the a priori. rules and the labeled examples.

In the following section, the methodology of how to define the constraints and how to optimize the cost function are discussed in details.

3.3.1 Constraints and Logic

Let us assume that a first-order logic knowledge base (KB) is given, whose predicates are either fully known a priori. (*given*) for all possible groundings or are approximated via the functions that are learned. For example, in classification problems, a first-order logic knowledge base can be defined to express how the classifiers should behave. In particular, each classifier corresponds to a predicate in the KB, while other

known predicates can be used during the training process. These additional predicates are assumed to be given, meaning that their output value is known a priori. for all possible groundings.

Any FOL clause in the knowledge base has an equivalent version in PNF, that has all the quantifiers $(\forall, \exists)$ and their associated quantified variables at the beginning of the clause. With no loss of generality, FOL clauses in the PNF forms are considered in this research work. Any quantifier free expression defined over the set of atomic formulas is equivalent to a sentence in the propositional logic. The expression is mapped in the KB into a form suitable for optimization, that is a fuzzy generalization of the expression.

Fuzzy FOL can transform any FOL knowledge into a real valued constraint. Fuzzy FOL uses t-norms (KMP00) to compute the degree of satisfaction of the rule for a given grounding of the variables. A degree of satisfaction of the FOL formula is obtained by iteratively grounding the variables in a formula and aggregating the values using the average and maximum operations over the obtained values for the universal and existential quantifiers, respectively as discussed below.

T-norms and Grounded Expressions T-norms (known as triangular norms) (KMP00) are used for converting a propositional logic expression into a continuous and differentiable logical constraint.

A *t-norm* is a function $t : [0, 1] \times [0, 1] \rightarrow [0, 1]$, which has the following properties:

- is continuous,
- is commutative (i.e. $t(x, y) = t(y, x)$),
- is associative (i.e. $t(x, t(y, z)) = t(t(x, y), z)$),
- is monotone (i.e. $y \leq z \Rightarrow t(x, y) \leq t(x, z)$),
- features a neutral element 1 (i.e. $t(a, 1) = a$).

A strict t-norm is also strictly monotone. A t-norm fuzzy logic is defined by its t-norm $t(a1, a2)$ that models the logical AND. A t-norm expression behaves as classical boolean logic when the variables assume crisp values: 0 (False) or 1 (True).

Given a variable $\bar{a}$ with continuous generalization a in $[0, 1]$, its negation $\neg\bar{a}$ corresponds $1 - a$. Once the t-norm functions corresponding to

Table 1: The operations performed by single units of an expression tree depending on the inputs x, y and the t-norm used.

op \ t-norm	Prod	Min	Luka	Weak-Luka
$x \wedge y$	$x \cdot y$	$\min(x, y)$	$\max(0, 1 - x - y)$	$\min(x, y)$
$x \vee y$	$x + y - x \cdot y$	$\max(x, y)$	$\min(1, x + y)$	$\min(1, x + y)$
$\neg x$	$1 - x$	$1 - x$	$1 - x$	$1 - x$
$x \Rightarrow y$	$\min(1, \frac{y}{x})$	$x < y?1 : y$	$x < y?1 : y - x$	$\min(1, 1 - x + y)$

the $\wedge$ and $\neg$ are defined, they can be composed to generalize any logic proposition. Different t-norm fuzzy logics have been proposed in the literature. For example, given two Boolean values $\bar{a_1}$, $\bar{a_2}$ and their continuous generalizations a_1, a_2 in $[0, 1]$:

- *Product t-norm* is defined as: $(\bar{a_1} \wedge \bar{a_2}) \rightarrow t(a_1, a_2) = a_1.a_2$.
- *Minimum t-norm* is defined as: $(\bar{a_1} \wedge \bar{a_2}) \rightarrow t(a_1, a_2) = min(a_1, a_2)$.
- *Lukasiewicz t-norm* is defined as: $(\bar{a_1} \wedge \bar{a_2}) \rightarrow t(a_1, a_2) = max(0, a_1 + a_2 - 1)$.

The $\vee$ operator modeling logical OR is called the t-conorm and is consequently defined by using the De-Morgan's rule: $(\bar{a_1} \vee \bar{a_2}) \equiv \neg(\neg\bar{a_1} \wedge \neg\bar{a_2}) \rightarrow t(a_1, a_2) = 1 - (1 - a_1)(1 - a_2)$.

The equivalence $\bar{a_1} \Rightarrow \bar{a_2} \equiv \neg\bar{a_1} \vee \bar{a_2}$ is used in classic logic to represent implications (modus ponens). However, this equivalence is not appropriate to perform deductions with fuzzy variable values. Any t-norm has a corresponding binary operator $\Rightarrow$ called residuum, which is used in fuzzy logic to generalize implications when dealing with continuous variables.

A t-norm residuum provides a natural way to express human fuzzy reasoning (Nov87), while being equivalent to modus ponens when fuzzy variable values approach the extremes of the $[0, 1]$ range. The residuum allows to relax the condition of satisfaction for the implication, which is satisfied as soon as the t-norm expression of the head has a higher truth degree than the t-norm expression of the formula body.

For example, the product t-norm has a residuum defined as:

$$(\bar{a_1} \Rightarrow \bar{a_2}) \rightarrow t(a_1, a_2) = \begin{cases} 1 & a_1 \leq a_2 \\ a_2/a_1 & a_1 > a_2 \end{cases}$$

while in Lukasiewicz t-norm, the residuum is defined as:

$$(\bar{a_1} \Rightarrow \bar{a_2}) \rightarrow t(a_1, a_2) = \begin{cases} 1 & a_1 \leq a_2 \\ 1 - a_1 + a_2 & a_1 > a_2 \end{cases}$$

In propositional logic, a grounded expression is any fully grounded FOL rule. An expression tree is built for each considered grounded FOL rule, where the basic logic operations ($\neg, \wedge, \vee, \Rightarrow$) are replaced by a unit computing the logic operation using a t-norm. The expression tree can take as input, the output values of the grounded predicates, and then recursively compute the output values of all the nodes in the expression tree. The value obtained in the root node is the result of the evaluation of the expression given the input grounded predicates. Table 1 details the operations computed by the units in the forward step given the inputs for different selections of the t-norm.

For example, consider the rule $\forall x[\neg A(x)] \vee [B(x) \wedge C(x)]$ where A, B, C are three predicates that must be approximated (via learning) by the unknown functions f_A, f_B, f_C. For any given grounding, the expression tree returns the output value: $t_E(\boldsymbol{f}(x)) = 1 - f_A(x) + f_A(x) \cdot f_B(x) \cdot f_C(x)$. Figure 6 shows the expression tree and the computation that is performed for the previous FOL rule grounded with $x = \bar{x}$. As for another example, lets consider the rule $\forall x[\neg A(x)] \wedge [B(x) \vee C(x)]$. Using the Weak Lukasiewicz t-norm as reported in Table 1, the expression tree returns the output value: $t_E(\boldsymbol{f}(x)) = min(1 - f_A(x), min(1, f_B(x) + f_C(x))$ for a given grounding x.

Quantifiers For the conversion, of any first-order logic expression into PNF, the quantified variables are moved in the beginning of the expression. For example: If there is an expression, $\exists x(P(x) \wedge \forall y(Q(y) \rightarrow R(x, y)))$, then moving the quantifier to the front gives the following, $\exists x \forall y(P(x) \wedge (Q(y) \rightarrow R(x, y)))$. The quantified portion of the expression is then processed recursively by moving backward from the inner to the outer quantifiers in the PNF. As, already mentioned there are two types of quantifiers: the universal quantifier ($\forall$) and the existential quantifier ($\exists$). The degree of truth of a formula containing an expression E with a universally quantified variable x_i is the average of the t-norm generalization $t_E(\cdot)$, when grounding x_i over $\mathcal{X}_i$ given as:

$$\Phi(\boldsymbol{f}(\mathcal{X})) = \frac{1}{|\mathcal{X}_i|} \sum_{x_i \in \mathcal{X}_i} t_E\big(\boldsymbol{f}([x, \mathcal{X}/\mathcal{X}_i])\big) \tag{3.2}$$

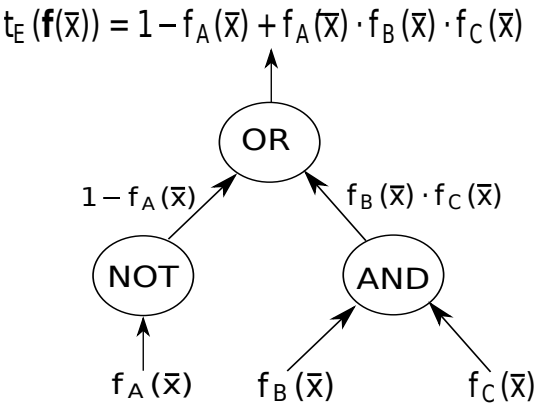


Figure 6: Forward propagation of the values in the expression tree of a FOL formula for the grounding $x = \bar{x}$ when using the product t-norm. The output of the root node returns a value in $[0, 1]$ corresponding to the evaluation of the rule for the given grounding.

In general, the universally quantified variable x_i is defined as the minimum of $t_E(\cdot)$ obtained as the t-norm generalization of E when grounding x_i over $\mathcal{X}_i$, but in this research work, the *min* operator over the t-norm values has been replaced by the average over the set. This definition allows in faster convergence during training of the model because the *min* operation directly depends only on one item over the set of groundings, whereas the average depends on all elements and allows parallel optimization of the entire set during training. The two formulations are also consistent with each other.

In the previous example represented in Figure 6, would yield the following functional on replacing the quantifiers:

$$\Phi(\boldsymbol{f}(\mathcal{X})) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} t_E\big(\boldsymbol{f}(x)\big) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} 1 - f_A(x) + f_A(x) \cdot f_B(x) \cdot f_C(x)$$

For the existential quantifier, the truth degree is instead defined as the *maximum* of the t-norm expression over the domain of the quantified variable. When multiple universally or existentially quantified variables are present, the conversion is recursively performed from the inner to the outer variables as already stated. In particular, when only universal quantifiers are present the aggregation is the overall average over each

grounding x:

$$\Phi(\boldsymbol{f}(\mathcal{X})) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} t_E\big(\boldsymbol{f}(x)\big)$$

3.3.2 Translation of FOL clauses into constraints

Summarizing the conversion process of a FOL clause into a constraint functional consists of the following three steps:

1. PREDICATE SUBSTITUTION: substitution of the predicates with their continuous implementations realized by the functions $\boldsymbol{f}$, mapping the output values into the interval $[0, 1]$.
2. CONVERSION OF THE PROPOSITIONAL EXPRESSION: conversion of the quantifier free expression where all the variables are grounded following the process detailed above.
3. QUANTIFIER CONVERSION: conversion of the universal and the existential quantifiers as explained above.

The form of the constraints depends on which t-norm has been used to generalize the FOL rules. Whereas it is always true that all t-norms are consistent with classical logic when variables assume crisp values $[0, 1]$, but the behavior of the various t-norms differs for the intermediate values, ultimately leading to the formulation of different constraints.

3.4 Backpropagation with Logic Constraints

Equation 3.1 representing the cost function of the SBR can be optimized via gradient descent, where the derivative of the cost function with respect to the j-th weight of the i-th function w_{ij} is given as:

$$\frac{\partial C_e}{\partial w_{ij}} = \sum_{x \in \mathcal{E}_i} \frac{\partial L(f_i(x), y_k(x))}{\partial f_i} \cdot \frac{\partial f_i}{\partial w_{ij}} + \sum_k \frac{\partial C_e}{\partial L_c} \cdot \frac{\partial L_c}{\partial \Phi_k} \cdot \frac{\partial \Phi_k}{\partial f_i} \cdot \frac{\partial f_i}{\partial w_{ij}} \quad (3.3)$$

the regularization term has been omitted to keep the notation simple. Therefore, only the second and the third terms of the Equation 3.1 are represented in the optimization Equation 3.3. Assuming that the $f_i(\cdot)$ are implemented by a neural network, the first term corresponds to the classical labeled error which can be minimized via backpropagation.

Since t-norms guarantee that Φ_k is in the range $[0, 1]$, setting $L_c(\cdot) = L^1(1, \cdot) = 1 - \cdot$ yields $\frac{\partial C_e}{\partial L_c} = -1$. The last step of the gradient computation $\frac{\partial f_i}{\partial w_{ij}}$ can also be computed via standard backpropagation step using the underlying neural network. This learning schema natively generalizes to the collective classification case (described in the next section) with the only difference that when learning the output values in collective classification, no backpropagation down to the model weights is performed (e.g. $\frac{\partial f_i}{\partial w_{ij}}$ is dropped).

This section shows how to efficiently compute the $\frac{\partial \Phi_k}{\partial f_i}$ using a backpropagation schema over the expression tree. It is demonstrated with rules having universally quantified variables but the same ideas can be trivially extended to rules including existentially quantified variables. This backpropagation schema which helps to optimize any neural network integrated with SBR using gradients over expression trees, is one of the most important fundamental addition to the SBR framework, unique from the previous works of SBR. A universally quantified rule is aggregated over the groundings via the average operator and, therefore, all the groundings should respect the grounded FOL formula. Learning can be reduced to find the function values that respect the rule for all possible groundings and backpropagation can be performed over the expression tree built for the selected t-norm and each given grounding:

$$\frac{\partial \Phi_k}{\partial f_i} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \frac{\partial t_E\left(\boldsymbol{f}(x)\right)}{\partial f_i}$$

In particular, the forward propagation over the network is performed for each single grounding of the variables, then the satisfaction error E of the rule is computed based on the value at the root of the expression tree. The computation of the gradient with respect to the model weights is performed by using the chain rule backward over the expression tree:

$$\frac{\partial E}{\partial o_n} = \frac{\partial E}{\partial o_{p(n)}} \cdot \frac{\partial o_{p(n)}}{\partial o_n}$$

where o_n is the output of the node n in the expression tree, $p(n)$ indicates the parent of node n in the tree. The root node is assumed to be the node 0 for which $o_0 = t_E\left(\boldsymbol{f}(\boldsymbol{x})\right)$ and the derivative error at the root node is: $\frac{\partial E}{\partial o_0} = -\frac{\partial L_c}{\partial \Phi_k}$.

For any node n, the derivative $\frac{\partial o_{p(n)}}{\partial o_n}$ is determined by the used t-

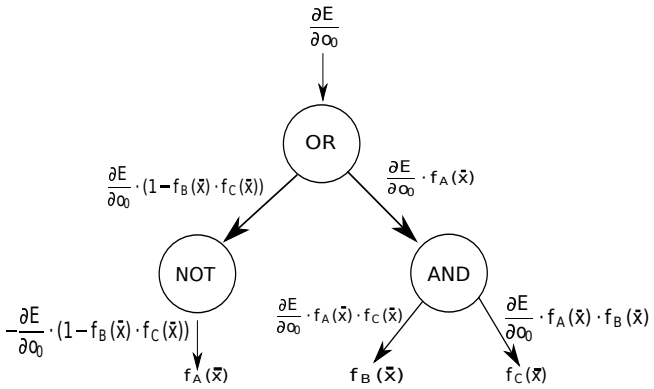


Figure 7: The backpropagation of the error over the expression tree for the grounding $x = \bar{x}$ of the FOL rule $\forall x[\neg f_A(x)] \vee [f_B(x) \wedge f_C(x)]$ using the product t-norm. The backpropagated error reaches a leaf node and it is passed as error derivative for a further backpropagation pass over the network implementing the function in the leaf node.

norm and the logic operation $t\text{-}norm(n)$ computed by the unit, such that

$$\frac{\partial o_{p(n)}}{\partial o_n} = t\text{-}norm(p(n))'_{\{l,r\}}$$

where the subscripts $\{l, r\}$ indicate whether node n is the left or right child of $p(n)$. This is needed because the inputs to the logic operations are generally not symmetric. The negation is a unary operation and requires no subscripts. In particular, Table 2 details how the gradients are backpropagated depending on the specific operation and the considered t-norm.

This establishes an efficient gradient computation schema over the expression tree, where the error of the considered FOL constraint is backpropagated from the root to the leaves. Figure 7 shows the backpropagation of the error for the example used across this paper $\forall x[\neg f_A(x)] \vee [f_B(x) \wedge f_C(x)]$. At the bottom of the expression tree, the backpropagated error reaches a leaf node, and it triggers a further backpropagation pass over the network implementing the function stored in that node.

Table 2: The derivatives used in the backpropagation step depending on the t-norm used and the operation implemented by the unit. Prod, Min, Luka and Weak-Luka are abbreviations for Product, Minimum, Lukasiewicz and Weak-Lukasiewicz t-norm.

$t\text{-}norm$ / $t\text{-}norm'_{op}$	Prod	Min	Luka	Weak-Luka
$(x \wedge y)'_l = \frac{\partial(x \wedge y)}{\partial x}$	y	$y < x ? y : 0$	$x + y < 1 ? -1 : 0$	$y < x ? y : 0$
$(x \wedge y)'_r = \frac{\partial(x \wedge y)}{\partial y}$	x	$x < y ? x : 0$	$x + y < 1 ? -1 : 0$	$x < y ? x : 0$
$(x \vee y)'_l = \frac{\partial(x \vee y)}{\partial x}$	$1 - y$	$y > x ? y : 0$	$x + y < 1 ? 1 : 0$	$x + y < 1 ? 1 : 0$
$(x \vee y)'_r = \frac{\partial(x \vee y)}{\partial y}$	$1 - x$	$x > y ? x : 0$	$x + y < 1 ? 1 : 0$	$x + y < 1 ? 1 : 0$
$(\neg x)' = \frac{\partial(\neg x)}{\partial x}$	-1	-1	-1	-1
$(x \Rightarrow y)'_l = \frac{\partial(x \Rightarrow y)}{\partial x}$	$x < y ? 0 : -\frac{y}{2 \cdot x^2}$	0	$y > x ? y : 1$	$x + y < 1 ? 0 : -1$
$(x \Rightarrow y)'_r = \frac{(\partial x \Rightarrow y)}{\partial y}$	$x < y ? 0 : -\frac{1}{x}$	$x < y ? 0 : 1$	$x > y ? x : -1$	$x + y < 1 ? 0 : 1$

3.4.1 Collective Classification

Collective classification (SNB$^+$08) is also a Statistical Relational Learning (SRL) method. It is defined as the task of performing inference over a set of instances that are connected among each other via a set of relationships. In this context, collective classification is used to refine the function output to be consistent with the FOL knowledge used during training. One of the main advantages of the proposed method is that it is by no means limited in the type of relationships that can be modeled. There are several real world applications of collective classification like enforcing temporal consistency during the test time on the consecutive video frames in any video classification or video sequencing problem like tracking a player in the soccer game videos, hyperlinked document classification, predicting political affiliations based on online purchases and interactions in social network analysis etc.

In particular, let $f_k(\mathcal{X}'_k)$ indicate the vector of values obtained by evaluating the function f_k over the data points of the test set $\mathcal{X}'_k$. The set of vectors is indicated as: $\boldsymbol{f}(\mathcal{X}') = f_1(\mathcal{X}'_1) \cup \ldots \cup f_T(\mathcal{X}'_T)$. If no neural network has been trained for f_k (no examples or no feature representations are available during training), $f_k(\mathcal{X}'_k)$ is assumed to be just filled with

default values equal to 0.5.

The prior knowledge in SBR is employed during the training time, assuming that the learning process encodes the knowledge into the parameters via the training set. However, if the number of examples are small, there is no guarantee that the prior knowledge will be respected by the outputs on the test patterns. Hence, collective classification in SBR helps to enforce the constraints also on the test data. Any types of relationships between patterns, labels, set of features (if the features are externally extracted as for SVMs) can be modeled.

Collective classification can be formulated as a minimization problem that searches for the values $\bar{\boldsymbol{f}}(\mathcal{X}'_k) = \bar{f}_1(\mathcal{X}'_1) \cup \ldots \cup \bar{f_{\mathcal{T}}}(\mathcal{X}'_T)$ respecting the FOL formulas on the test data, while being close to the prior values established by the neural networks over the test data:

$$C_{cc}[\bar{\boldsymbol{f}}(\mathcal{X}'), \boldsymbol{f}(\mathcal{X}')] \;=\; \frac{1}{2}\sum_{k=1}^{T} |\bar{f}_k(\mathcal{X}'_k) - f_k(\mathcal{X}'_k)|^2 \;+\; \lambda_l \sum_h L_c\left(\Phi_h\big(\bar{\boldsymbol{f}}\Big(\mathcal{X}'\Big)\Big)\right)$$

The gradient computation in Equation 3.3 transparently generalizes to the collective classification case and hence the collective classification objective function can be optimized using gradient descent. The only difference is that during collective classification, the weights of the trained neural network are fixed, and no backpropagation down to the network weights is performed. Hence, this provides a very elegant solution as there is no additional complexity in the implementation of collective classification.

3.4.2 Optimization

Constraints and Local Minima The constraints resulting from a FOL formula can be hard to optimize during learning. T-norms used to translate first-order formulas into real valued constraints, do not guarantee convexity unless attention is restricted only to Horn clauses. Every individual atom in the FOL clause, can have a number of possible assignments for a given grounding of variables, and hence there can be several local minimas in the expression generalizing the FOL formula to a continuous domain. The intractability of the FOL inference gets translated into the SBR cost function and therefore that gets plagued by many local minima.

Convexity of Constraints Optimization of the SBR cost function converges relatively faster by increasing the proportion of convex constraints that can be effectively exploited during learning. There are different possibilities to increase the proportion of convex constraints or build convex constraints from FOL formulas.

T-norm-residua are consistent with modus ponens of classical logic at the extremes of the variable range and also soften the conditions under which the formula is verified. Also, the application of t-norm residua to translate logic implications, significantly increases the proportion of convex constraints that can be exploited during learning than that would happen using a modes ponens based translation.

Application of negation ($\neg$) to single atomic formulas results in forming convex constraints but when negation is applied to arbitrary terms (here a term is referred to an expression consisting of groups of atomic formulas), convexity of constraints is not guaranteed.

Using specific t-norms like Lukasiewicz and Weak Lukasiewicz over other t-norms help to exploit theorems and fundamental properties of the chosen logic to get advantage of the learning strategy. Lukasiewicz logic in either its strong (using strong conjunctions and disjunctions denoted as $\otimes$ and $\oplus$) or weak (using weak conjunctions and disjunctions denoted as $\wedge$ and $\vee$) form (GDGM17) has advantage over other fuzzy logics like Product and Minimum as it provides an equivalent prenex normal form and a continuous involutive negation preserving the De-Morgan laws. It is also to be noted, that the distributive property of connectives (conjunctions and disjunctions) actually helps to define weak connectives from strong connectives and vice versa in every fuzzy logics. Mc-Naughton Theorem provides a functional representation of Lukasiewicz formulas by piecewise linear functions. Exploiting this property, it is possible to characterize the fragment of Lukasiewicz formulas to corresponding convex functional constraints (GDGM17) and also an equivalent set of linear constraints.

Optimization heuristics Employing the cost function defined by Equation 3.1 forces two issues. Direct optimization is not simple because the $\Phi_h(\cdot)$ can be non-convex and with a large number of local minima (DGS15). Furthermore, the equation introduces many meta- parameters $\lambda_l^k,\ \lambda_h$ whose values must be determined. This section discusses some heuristics that have been employed in SBR to mitigate these issues.

The optimization problem in SBR is generally intractable. However,

there is a large class of logic statements that have been shown to be able to translate into a convex constraint (GDGM17). Therefore, the following optimization strategy has been employed:

1. solve the optimization problem stated by Equation 3.1 using only the convex constraints (e.g. $\lambda_h \neq 0$ iff $L_c(\Phi_h(\cdot))$ is convex). This allows to efficiently find a good initial approximation of the best solution.

2. add the other non-convex constraints, setting the corresponding λ_h to non-zero values and continue the optimization until convergence.

The optimization strategy explained above have never been used before in the literature where kernel machines were integrated in the SBR framework. Therefore, it was not possible to add concave constraints during learning which resulted in exploitation of limited number of logic rules from the knowledge base.

Collective classification also aims to correct the output of the networks that are not consistent with the provided prior knowledge during the test time. The first term of Equation 3.1 determines how much to penalize deviating from the output provided by the neural networks. It is clear that this cost should not be constant for all predicates, but it depends on how reliable the output of a predicate is. In particular, the cost should be high for the predicates that have been trained successfully and provide a very good accuracy metric measured over a validation set. On the other hand, a predicate scoring poorly on the validation set is not reliable in its estimates and deviating the final output from the predicate values should be allowed more easily during collective classification. Hence this framework of SBR having predicate based λ_l^k values, known as vanilla SBR makes the framework more flexible. In vanilla SBR, λ_l^k are set such that

$$\lambda_l^k = \lambda_l * Acc(f_k(\mathcal{X}_k^v), y_k(\mathcal{X}_k^v)) \ ,$$

where $\mathcal{X}_k^v$ is the set of data points for the domain of f_k in the validation set, $y_k(\mathcal{X}_k^v)$ are the corresponding desired outputs for the patterns in the validation set for f_k, $Acc(\cdot)$ is the accuracy metric and λ_l is the only remaining meta parameter to scale the overall regularization cost, which can be set by maximizing the overall performance on the validation set.

Therefore, all these optimization heuristics and test time strategies described above used to tightly integrate the deep neural networks with

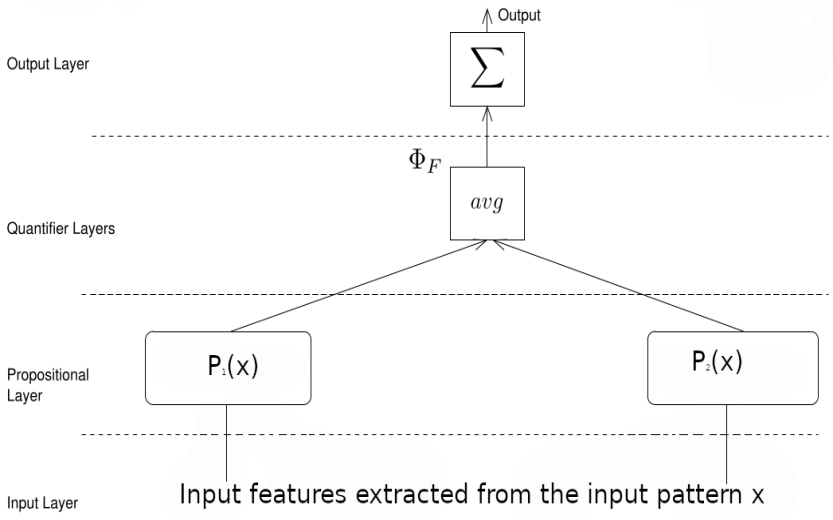


Figure 8: Semantic Based Regularization multi-stage architecture.

logical constraints make the SBR framework more powerful and unified for different kinds of application than its earlier versions.

3.5 SBR as multi-layer architecture

Given an arbitrary set of FOL logic clauses in the KB, SBR can be generally encoded by a multi-stage architectural framework. Let the FOL formula to be computed using SBR be $F : \forall x(P_1(x) \wedge P_2(x))$ as shown in Figure 8.

- *Input Layer*: does the computation of the query and evidence functions for all possible groundings of the atomic formulas. The groundings are the feature representations computed on the inputs of the neural network learners. For example: In the Figure 8, pattern x is the input to the neural network whose features are computed in the input layer that is grounded to the predicates P_1 and P_2.
- *Propositional Layer* : the value of the t-norm expression for the propositional part of each FOL formula is computed for each of the com-

patible combination of the atomic formulas. For example: Using appropriate t-norm, the grounded predicates are converted into real valued functions $f_1(x)$ and $f_2(x)$.

- *Quantifier Layer* : the t-norm values computed at the propositional layer are aggregated by the average or max operator for universal or existential quantifiers respectively. The number of quantifier layers are not fixed as the aggregation of the outputs are recursively nested according to the number of quantifiers in the FOL formula. For example: For the rule F, the universal quantifier $\forall$ is replaced by average operator in this layer.

- *Output Layer* : accumulates the summation of the contributions of each atomic formula to build the entire FOL formula. For example: In F, the output value of the FOL formula F is finally obtained in this layer.

3.6 Summary

This chapter presents the SBR framework that integrates general prior knowledge into deep learners, allowing to distill the knowledge into the model weights during training. The backbone of the SBR framework is based on the previous works of SBR on shallow networks. As already stated, for the first time in this research work, it exploits the advantage of rich feature learners that allows to learn from raw inputs rather than pre-processed feature vectors. It also introduces several novel concepts that forms the main contributions of this work like defining of the expression tree based backpropagation schema for joint training of deep CNN networks with constraints, the comparison of constant weight based and vanilla SBR (empirically compared in Chapter 5), comparison of using different fuzzy logics during joint training in neural nets, combining different fuzzy logics on different subsets of rules to obtain optimum performance gain (empirical results in Chapter 5) and most importantly the ability of addition of concave constraints heuristically in the deep learners that helps to keep the optimization tractable and effective. It also extends the collective classification mechanism of SBR for neural network predictions that was defined initially for kernel machines. All these extensions make the SBR framework a powerful mathematical approach for semi-supervised learning, a scalable and unified pathway for large

scale classification tasks. This framework is used in the subsequent chapters for experimental evaluation on different image and video classification tasks and is shown to improve the accuracy of different state-of-the-art deep neural architectures through injection of background knowledge over their individual counterparts.

Chapter 4

Machine Learning Methods and Frameworks for Vision Tasks

This chapter presents an overview of deep learning and then briefly describes all the different deep learning frameworks used in the vision tasks like image/video classification, image segmentation etc. in the following chapters of the thesis. The chapter is broadly divided into two parts. The first part introduces different deep convolutional neural networks and probabilistic frameworks used in image and/or video classification tasks described in Chapter 5 and Chapter 6 (the different neural learning machineries of the semantic based regularizer) and in Chapter 7 (describes the backbone architectures of the segmentation frameworks like Mask RCNN and UNET). The second part briefly describes the different image segmentation frameworks like Mask RCNN and UNET used for solving a real world problem of contaminant separation described in Chapter 7.

Deep Learning

Deep learning (BLP+07; LeC15; Hin12) is considered as a part of machine learning research with the focus on learning data representations. Deep networks successfully integrate low/middle/high level features and classifiers in an end-to-end multi-layer fashion. Deep learning has been successfully applied to many applications like computer vision

(LHB04; LGRN09; KSH12b), phonetic recognition (KSH12a; FCNL13; TJLB14; SLJ+15), audio processing (MDP+11; Hin12; SMKR13), drug discovery (BLP+07) and many others.

In particular, Deep Convolutional Neural Networks (CNN) (LHB04; LGRN09; KSH12b) have been shown to outperform shallow architectures like support vector machines, random forest classifiers in many vision applications like image and video recognition, medical image analysis, document analysis, keypoint detection etc. Convolutional networks are inspired by biological processes in which the connectivity pattern between the neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they can cover the entire visual field. The convolutional neural networks also have the convolutional layer filters known as kernels that convolve over any input image in a similar manner to build feature maps.

LeCun et al. established the framework of CNNs by developing a multi-layer artificial neural network known as LeNet (LHB04) in the year 2004. LeNet was initially used for classification of handwritten digits and could be trained with the backpropagation algorithm which made it possible to recognize patterns directly from raw pixels thus eliminating a need for a separate feature extraction mechanism. But even with all these advantages, due to the lack of large training data and computational power at that time, LeNet failed to perform well on complex problems such as video classification. Since the proposal of LeNet, the advanced architecture, AlexNet was also developed and it became the winner of the ImageNet challenge (KSH12a) in 2012, achieving an error rate considerably lower than its non CNN competitors. Subsequently, ZFNet (ZF13), VGG (SZ14), Network in Network (NIN) (LCY13), GoogleNet (SLJ+15), Residual Networks (HZRS16a), Inception Networks, Pyramidal Residual Networks (HKK16) and Wide Residual Networks (ZK16) were successfully proposed to demonstrate advances in neural network architectures. These convolutional neural network models have been built by leveraging their capacity to varying depths and breadths without any loss in their performance. Various techniques have been suggested to enable training of deeper neural networks, such as well- designed initialization strategies (GB; HZRS15b), better optimizers, skip connections, knowledge transfer, layer-wise training etc. With the invent of higher computational power, using these deep neural networks in different applications have now become a reality. They have been applied to

several complex vision tasks and many of them achieved huge empirical success.

4.1 Methods and Deep Neural Network architectures used in Vision Tasks

This section presents a brief description of the CNN models used in the experimental simulations in the Chapter 5, Chapter 6 and Chapter 7. Some of them have been used as an exact replication of their original works while the others have been adapted to the classification and segmentation tasks discussed in the following chapters.

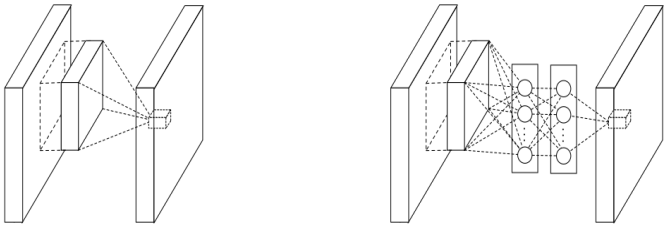
For example: the implementation tweaks made to the original works of the backbone networks used as the learning apparatus of Semantic Based Regularization (SBR), mainly involves the addition of independent softmax or sigmoid classification layers to address each level of hierarchy in the datasets represented as the intermediate classes in the experiments. To explain this further lets consider the CIFAR-100 dataset. It has two taxonomical levels of hierarchy namely the coarse classes and the fine classes. In SBR to exploit the hierarchical relationship between the coarse and the fine classes, two softmax layers are added to each of the deep CNN models (original works just have one layer outputting the fine classes) used. One of these output layer contains the 20 coarse classes whereas the other has 100 neurons for each of the fine classes.

Another example of how the deep CNN networks are adapted is also given in Chapter 7 where a Mask RCNN (HGDG17) framework is used for segmenting the contaminant particles for cleanliness analysis. In this modified Mask RCNN framework, a deep CNN architecture ResNeXt-101-FPN is used as a feature extractor, a combination of a ResNeXt (XGD$^+$16) and FPN (KGHD19) CNN model.

Network-In-Network Models

Network In Network model (NIN) (LCY13) is a specialized model where the linear filters of the traditional convolution layer have been replaced by micro neural networks to abstract the data within the receptive field. It helps to enhance the model discriminability for local patches within the receptive field. The micro neural network is actually a multilayer perceptron and is called as a mlpconv layer, which is an universal nonlinear function approximator. The mlpconv layer is used to obtain fea-

ture maps by sliding it over the input like any normal convolution layer. It is desirable to use a universal function approximator for feature extraction of the local patches, as it is capable of approximating more abstract representations of the latent concepts. Rectified linear unit is used as the activation function in the mlpconv layers. Figure 9 shows the structural difference between a linear convolutional layer and a mlpconv layer.



(a) Linear convolution layer (b) Mlpconv layer

Figure 9: Compares a linear convolutional layer with a mlpconv layer (LCY13).

Deep NIN models are implemented by stacking several mlpconv layers and putting sub-sampling layers in between mlpconv layers. A deep NIN network so formed, is shown in Figure 10. The spatial average of the feature maps from the last mlpconv layer is obtained through global average pooling and then fed into the softmax logistic regression layer.

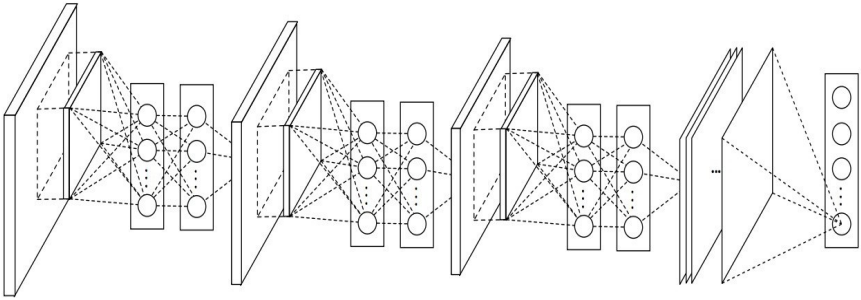


Figure 10: A deep Network in Network Model (NIN) (LCY13).

Resnet Models

Resnet models originally designed in the works of He et al. (HZRS16a) allow to find a compromise between large network depth that is needed to perform complex vision tasks and the resulting complexity of the training involved. The residual networks address the problem of degradation of training accuracies in deep convolutional neural networks by learning residual functions with reference to the layer inputs that are easier to optimize and have accuracy gains from considerably increased depth. Instead of hoping that each few stacked layers will directly fit into an underlying mapping as in most deep neural networks, in residual networks, layers are explicitly fitted to a residual mapping. Formally, the desired underlying mapping is denoted as $H(x)$, and let the stacked nonlinear layers fit another mapping $F(x) := H(x) - x$. The original mapping is recasted into $F(x) + x$. The formulation of $F(x) + x$, can be realized by feedforward neural networks with shortcut connections.

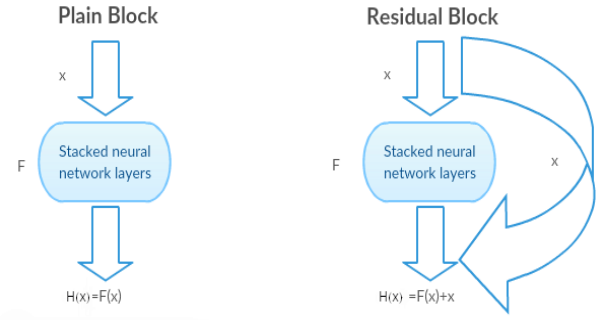


Figure 11: A comparison of a plain block with a residual block having identity connections (HZRS16a).

In residual networks, shortcut connections skip one or more layers, they perform identity mapping and their outputs are added to the outputs of the stacked layers as shown in Figure 11. These shortcut connections neither add extra parameters nor any extra computational complexity and the entire network can be trained using backpropagation mechanism. Deep residual networks are found to be more easily optimizable than their plain counterparts. Any plain network can be easily converted to its residual version by inserting shortcut connections when the inputs and the outputs are of same the dimensions. But when the dimensions

increase, either extra zero entries are padded for extra dimensions or projection shortcuts are used to match the dimensions by performing 1×1 convolutions. Residual networks can be either basic residual networks or bottleneck residual networks. For each residual function F, a stack of 3 layers are used in the bottleneck architecture whereas a stack of 2 layers are used in basic residual networks. The comparison of the basic block and bottleneck block is shown in the first two architectures of Figure 13. The number of parameters at different depths of the residual networks are summarized in Table 3.

Pre-activated Resnet Models

Pre-activated Resnets (HZRS16b) propose a new type of residual unit that further improves generalization of the residual networks and makes the training easier. In original residual networks, batch normalization (BN layer) is used after each weight layer and rectified linear unit activation (ReLU) is adopted after each BN except that the last ReLU is after element wise addition. However, in Pre-activated resenets (PreResnets) the activation functions (ReLU and/or Batch Normalization) are rearranged. A comparison of the original Resnet with different types of arrangement of the ReLU and BN layers is shown in Figure 12.

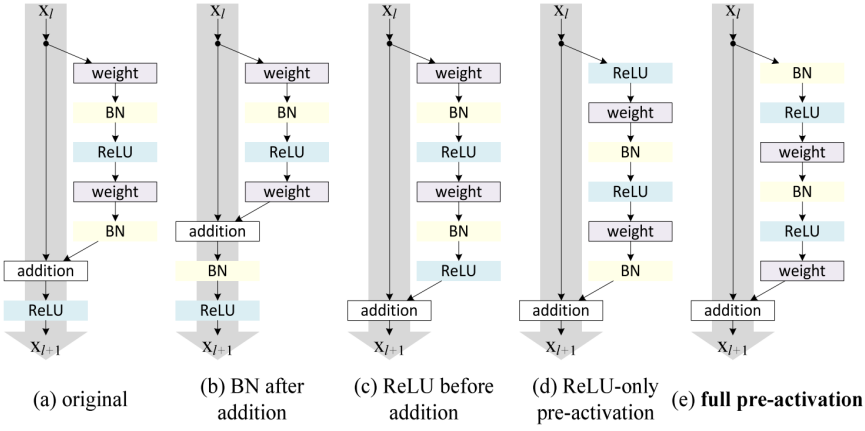


Figure 12: Comparison of original resents with different types arrangement of the ReLU and BN layers finally lead to a full pre-activation architecture (HZRS16b).

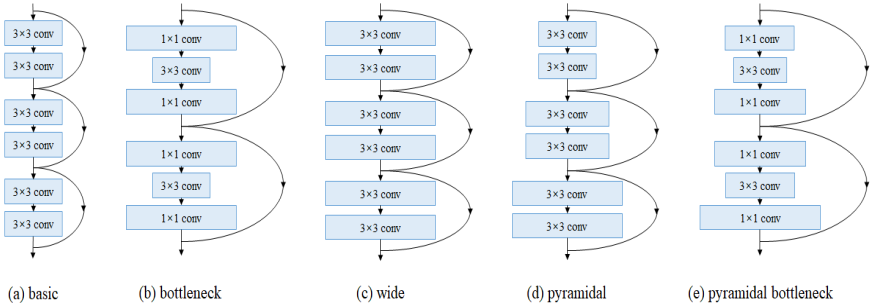


Figure 13: A comparison of different resents like basic, bottleneck, wide and pyramidal residual architectures.

In PreResnets, both BN and ReLU are used as pre-activation of the weight layers which improves regularization (reaches slightly higher training loss at convergence) of the model and also eases optimization. In the original residual unit, although the BN layer normalizes the signal but this is soon added as the input to the next weight layer, but in the pre-activation units, the inputs to all the weight layers have been normalized. PreResnets are also able to further exploit the dimension of network depth. The number of parameters at different depths of the PreResnets are compared with the parameters of other residual network configurations in Table 3.

Pyramidal Resnet Models

One of the best performing network on CIFAR-10, CIFAR-100 and ImageNet datasets is the Pyramidal Residual network (HKK16). In these networks, the feature map dimension of all units are gradually increased as a function of depth at which the layer occurs, resulting in a structure similar to a pyramid.

There are two types of pyramidal residual nets in the original works (HKK16) namely the additive and multiplicative networks with addition based and multiplication based widening step factor respectively. In the additive network, the feature map dimension increases linearly whereas in multiplicative network, it increases geometrically. In pyramidal networks projection identity shortcuts cannot be used since the feature map dimension differs among the individual residual units, therefore zero

padded identity shortcuts are used which are the skip connections. These shortcuts do not add any additional parameters and hence do not lead to the overfitting problem. Therefore, these networks have better generalization capabilities. The pyramidal residual networks also have basic block and bottleneck architectures as the normal residual networks and their comparison is shown in Figure 13. The number of parameters at different depths of the additive and multiplicative pyramidal residual nets are compared with the parameters of other residual networks in Table 3.

Resnets with Modified Residual Units

Pre-activated Resnets added a small modification to the original residual units by placing the activation layers before the weight layers, similarly, it was also found empirically in the works of Han et al. (HKK16), that the performance of the deep CNN networks are impacted largely by the location and the number of ReLUs used in the each block of the residual units. Therefore, a modified residual unit was designed by Han et al. (HKK16) removing the first ReLU in the blocks of each residual unit and also pre-activating each residual unit in the modified architecture. This enhanced the performance during inference without adding any extra hyper parameters. However, atleast one ReLU should exist in each residual block to maintain the non-linearity.

Wide Residual Networks

Wide Residual networks (ZK16) are an improved version of the original residual networks where the depth of the network is decreased while increasing the width of the network. In very deep networks, it was seen that although the gradient flowed through the network, nothing was learned in many residual blocks, and this was formulated as the diminishing feature reuse problem. Therefore, the wide residual networks have been designed to use fewer but wider blocks that learn more useful representations than deeper networks. They are developed using three simple modifications:

- Adding more convolutional layers per block.
- Widening the convolutional layers by adding more feature planes.
- Increasing the filter sizes in the convolutional layers.

There have been different widening factors for different wide residual architectures and some of them are summarized in Table 3 along with the number of hyper parameters in each of the architectures. WideResnets also have variations in configurations with and without dropouts in each residual blocks. These networks are also faster to train than their deeper counterparts as increasing width helps effectively balance computations in a much more optimal way. With the original works in wide residual networks (ZK16), it has been observed that widening consistently improves performance across residual networks of different depths. Also, increasing both depth and width helps until the number of parameters becomes too high and then stronger regularization like dropouts are needed.

ResNeXt Networks

ResNeXt (XGD$^+$16) is another modified architecture that takes its inspiration in different deep CNN models like Resnets, Wide Resnets, VGG networks (SZ14) etc. It is a simple and highly modularized architecture constructed by repeating any building block (like a residual block) by a set of transformations of the same topology and then aggregating them by summation. This results in a design of a homogeneous multi-branch architecture that has few hyper parameters to be tuned. However, there is a new dimension introduced known as the *cardinality*. It refers to the size of the set of transformations used in the repeated blocks. It is an essential factor in addition to the dimensions of depth and width of the network. The repetition of the blocks follow two simple rules:

- When producing the spatial maps of the same size, the blocks share the same hyper parameters,
- Each time when the spatial map is down sampled by a factor of 2, the width of the blocks is multiplied by a factor of 2. This ensures that the computational complexity is roughly the same for all blocks.

In the original works (XGD$^+$16) of ResNeXt, it has been empirically demonstrated that because of the aggregated transformations, the new architecture outperforms the original resnet module. The ResNeXt module with a cardinality of 32 is designed using a set of 32 transformations from a original resnet module as shown in Figure 14. The parameters

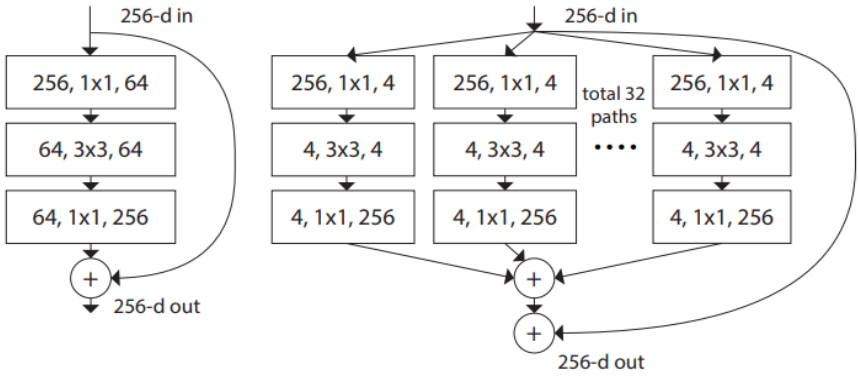


Figure 14: A block of ResNet is shown in the left whereas in the right, it is converted into a block of ResNeXt with cardinality of 32, having roughly the same complexity using a set of 32 transformations and aggregating through a summation (XGD$^+$16).

of few different sizes of ResNext networks are also given in Table 3 and compared with other network architectures used in different vision tasks.

Inception Networks

Inception module was firstly introduced in Inception-$v1$/GoogLeNet (SLJ$^+$15). The input of every block or module goes through 1×1, 3×3 and 5×5 convolution layers, as well as simultaneous max pooling layers and then concatenated together as the output for that module. A simple inception module is represented in the Figure 15. Inception networks are built using a collection of these inception modules.

Inception-$v2$ is the next version after Inception-$v1$ and it first used batch normalization in the inception modules. Factorization was used in the next version known as Inception-$v3$ to the reduce dimensionality of the modules and to solve the overfitting problem. Each of these versions show improvement over each other in different visual recognition tasks especially in image classification. The most advanced version of Inception network without residual connections is the Inception-$v4$ (SVI$^+$16) which has similar architecture to Inception-$v3$ but has more inception modules than the $v3$ version. However, there are also other versions of inception architectures with residual or skip connections like Inception-

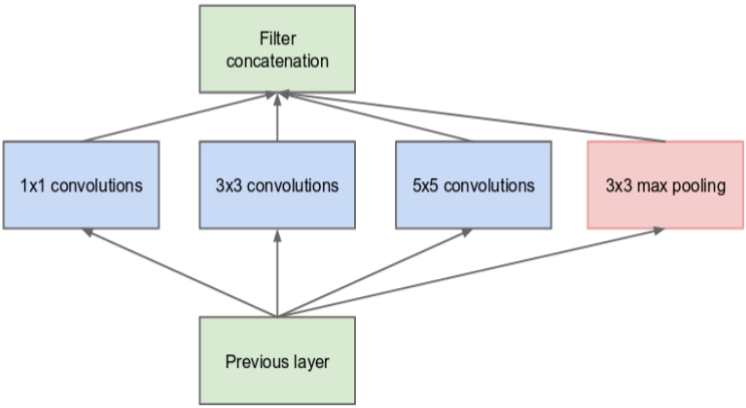


Figure 15: Inception module (SLJ$^+$15).

Resnet-$v2$, Inception-Resnet-$v4$ etc. The parameters of some of the different versions of Inception networks are summarized in Table 3.

NasNet Networks

NasNet are networks built from an efficient neural architecture search (NAS) algorithm (ZVSL17) that denotes the process of automatically designing artificial neural networks. Various approaches of NAS have designed networks that are on par or even outperform the hand-designed architectures. Methods for NAS can be categorized according to the following factors:

- Search space that defines which type of artificial neural network (ANN) can be designed and optimized in principle.
- Search strategy defines which strategy is used to find optimal ANN's within the search space.
- Finally the performance estimation strategy is used to finalize on the design.

In NAS, networks are searched on particular datasets. A controller Recurrent Neural Network (RNN) samples child networks with different convolutional architectures and then the child networks are trained

Table 3: Number of parameters in each of the different configurations of residual network architectures. # refers to the number in the Layers and Parameters column. M is the abbreviation for millions, α and γ are parameters specific to the network architectures.

CNN Networks	# Layers	# Parameters
Resnet	32	0.46 M
	110	1.7 M
	164	1.7 M
	1202	19.4 M
	50	25.6 M
	101	44.5 M
	152	60.2 M
PreResnet	110	1.7 M
	164	1.7 M
	1202	19.4 M
	50	25.6 M
	101	44.5 M
	152	60.2 M
Additive PyramidNet	110, α = 84	3.8 M
	110, α = 270	28.3 M
	164, α = 84	3.8 M
	164, α = 270	27.0 M
	236, α = 220	26.8 M
	272, α = 200	26.0 M
	200, α = 200	62.1 M
	200, α = 450	116.4 M
Multiplicative PyramidNet	110, α = 27	28.3 M
Wide Resnet	16, γ = 8	11.0 M
	28, γ = 10	36.5 M
	40, γ = 4	8.7 M
ResNeXt	50	25.0 M
	101	44.0 M
Inception	$v1$	7.0 M
	$v2$	7.0 M
	$v3$	2.3 M
NasNet-A		3.3 M

to converge and obtain some accuracy on the validation set of the particular dataset. The resulting accuracies are used to update the controller

so that the controller will generate better architectures over time. The controller weights are updated with policy gradient as described in the works of Zoph et al. (ZVSL17). NasNet-A is a NAS architecture with approximately 3.3 M (shown in Table 3) parameters searched and built on CIFAR-10 dataset.

Regularization of inputs

Residual Networks are sophisticated and efficient form of convolutional neural networks that are used in several vision tasks. But for the complex convolutional neural networks like residual models, with the increased representational power gains that they possess, they also have the increased probability of overfitting, leading to poor generalization ability during inference. A learned model tends to represent random error or noise instead of features from the underlying data distribution. To improve the generalization, several regularization techniques such as data augmentation, judicious addition of noise to activations, dropouts and batch normalizations have been implemented.

In the image classification experiments using deep CNN in SBR, some input based regularization techniques have been used, based on the works of Zhong et al. (ZZK$^+$17) and Terrance et al. (DT17). In the works of Zhong et al., a technique called Random Erasing (ZZK$^+$17) is proposed in Wide Residual Networks. Occlusion is a critical factor in generalization of CNNs and it is desirable that invariance to various levels of occlusion is achieved. It is hypothesized that when some parts of the object are occluded, a strong classification model should be able to classify the object correctly. During training, an image within a mini-batch is either randomly kept unchanged or a rectangular region in an image is chosen and the real pixels in them are replaced with random values (in the experimental evaluations, the mean-pixel values of the dataset are assigned). For an image I in a mini-batch, the probability of it undergoing random erasing is p, and the probability of it being kept unchanged is $1 - p$. Random erasing randomly selects a rectangle region I_e in an image, and erases its pixels with random values. Assuming the size of the training image is $W \times H$, the area of the image is $S = W \times H$. A random erasing region S_e is initialized, where S_e/S is in the range specified by minimum s_l and maximum s_h values. The aspect ratio of erasing rectangle region is randomly initialized between r_1 and r_2 and it is set to r_e. There are three parameters in random erasing: the erasing probability p, the area ratio range of erasing region s_l and s_h, and

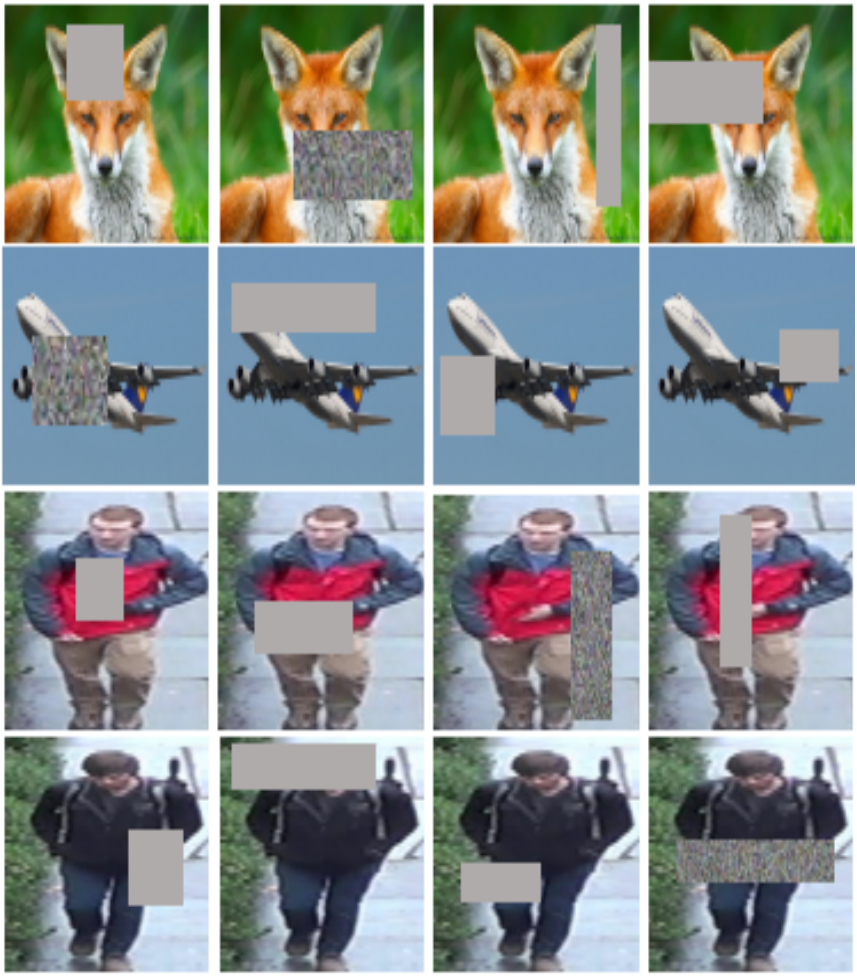


Figure 16: Random Erasing and Random Cutout applied to few selected natural images (ZZK$^+$17; DT17).

the aspect ratio range of erasing region r_1 and r_2 which are selected via cross-validation. In this process of random erasing, data is augmented on a huge scale as the training images are generated with various levels

of occlusion which helps to prevent the problem of overfitting in large CNNs. Random erasing preserves the global structure of every input image as it can be viewed as addition of noise to the image. Random Erasing is also a lightweight method that does not require any extra parameter learning or memory consumption.

Another very similar regularization technique applied at the input level is called as *Random Cutout* based on the works of Terrance et al. (DT17). This technique randomly masks out squared regions of the input image with zero values during each epoch of training. This method also effectively augments the dataset with partially occluded versions of the existing data samples. It is similar to the dropout technique except here the dropout occurs at the input stage rather than at the intermediate layers. A fixed-size zero mask is applied to a random location of each input image where the size of the cutout region is a hyper parameter that is determined via cross-validation. This improves the model robustness and ultimately yields better model performance. This method performs slightly better than random erasing, however the motivation is exactly the same and this yields the present state-of-the-art results with wide residual networks on CIFAR-10 and CIFAR-100 datasets. An example of random erasing and random cutout are shown on natural images in Figure 16. The squares showing noisy masks are random erasing masks and the zero gray masks represent the cutout technique.

Markov Random Fields

A Markov Random Field (MRF) is a probability distribution p over variables $x_1, \ldots, x_n$ defined by an undirected graph G, in which the nodes correspond to variables x_i. The probability p has the form:

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \tag{4.1}$$

where C denotes the set of cliques (i.e. fully connected subgraphs) of G and it is often referred to as the Markov blanket. The value of Z is:

$$Z = \sum_{x_1, \ldots, x_n} \prod_{c \in C} \phi_c(x_c) \tag{4.2}$$

and it is called as a normalizing constant or a partition function that ensures that the distribution sums upto to 1. Thus, given a graph G, the probability distribution may contain factors whose scope is any clique in

G, which can be a single node, an edge, a triangle, etc. Optimization in a MRF problem involves finding the maximum of the joint probability over the graph, usually with some of the variables given by some observed data. Equivalently, as seen from the equations above, this can be done by minimizing the total energy, which in turn requires the simultaneous minimization of all the clique potentials. The joint distribution over the random variables of a first-order Markov chain can be factored into a product of conditional distributions. This permits efficient inference. The key to MRFs is that, with the use of local connections, information can propagate a long way through the graph. This communication is important if we want to express some structure in which knowing the value of one node tells us something important about the values of the other nodes, possibly distant, nodes in the graph. Therefore, MRFs can be used to force coherence of any structured classifier outputs. For example, in videos when there is a temporal coherence among the adjacent frames, if we use a non-linear classifier to classify the frames independently, then MRF models can be used at the second stage to establish a temporal relationship among the consecutive frames and therefore improve the final classification outputs of the first stage classifier. This kind of temporal smoothing on the adjacent video frames in a video sequencing and tool annotation task is performed in Section 6.2.

4.2 Methods and Frameworks for Image Segmentation

Object detection and segmentation is another important vision task that is being actively pursued by the deep learning community. This section briefly describes the types of segmentation techniques like semantic and instance segmentation used for different tasks. It also outlines the original works of some of the segmentation model architectures specifically Mask RCNN and UNET that are used in building end-to-end solutions for the contaminant separation problem as described in Chapter 7. The adaptations of these methods and frameworks and the tweakings from their original works are described in more details in Chapter 7.

Feature Pyramid Networks

Detecting objects at different scales is a challenging task particularly for small objects. One of the solution is to use a pyramid of the same image

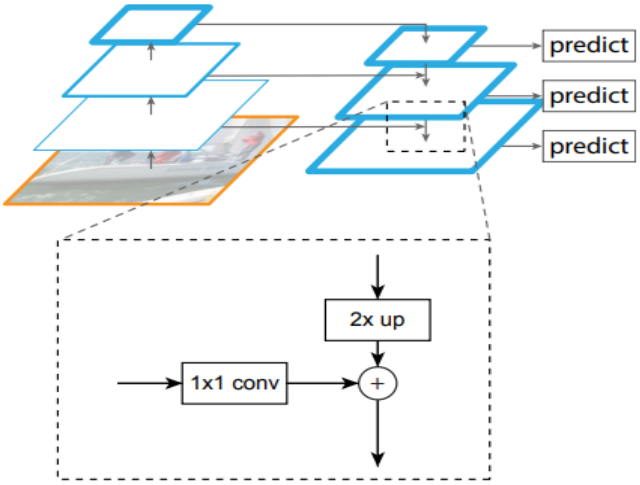


Figure 17: A building block of FPN illustrating the lateral connection from the bottom-up to the top-down pathway, merged by addition (LDG$^+$16).

at different scales to detect objects but processing multiple scale images is time consuming and not memory efficient if trained end-to-end. Alternatively, a pyramid of features from a single image input can be created and used for an object detection and segmentation task. Feature Pyramid Network (FPN) (LDG$^+$16) is a feature extractor that is designed for such pyramidal concept and is the present state-of-the-art feature extractor used in most object detectors.

FPN composes of a bottom-up and a top-down pathway. The bottom-up pathway is the usual convolutional network for feature extraction. At each level upward, the spatial resolution decreases with more high-level structures being detected and also the semantic value for each layer increases. However, the top-down pathway is used to construct higher resolution layers from a semantically rich layer. While the reconstructed layers are semantically strong but the locations of objects at the lower layers are not precise after the rigorous downsampling and upsampling processes. To take the advantages of both the pathways, lateral connections are added between the reconstructed layers and the corresponding feature maps to extract more location accurate information. It also acts as skip connections to make the training of the FPN easier. A schematic representation of the bottom-up and the top-down pathway with the lat-

eral connections in a building block of FPN is shown in Figure 17. FPN is not an object detector by itself. It is a feature detector that works with object detectors like Region Proposal Network (RPN) (RHGS15) of Mask RCNN.

In this thesis, Chapter 7 shows the effective use of feature pyramid network that provides the input to the RPN of the image segmentation framework, Mask RCNN (discussed in the Section 4.2) by extracting rich feature maps from different image scales.

Semantic vs Instance Segmentation

Figure 18: An Example of Semantic Segmentation.

Semantic segmentation aims at grouping pixels of an image in a semantically meaningful way into different categories. For example: Let us consider an image as shown in Figure 18, pixels belonging to road, persons, cars or trees each from a separate category and are therefore grouped separately. So semantic segmentation does a pixel-wise classification as if the pixel is a part of the object like road, some car or some person but it does not have any information about the boundaries of the individual objects.

Instance segmentation on the other hand is the process of detecting different instances of the same category. Instance segmentation systems are looking for the look alike objects in the scene and any objects that look different are ignored even if they belong to the same class as the object considered. It has information about the object boundaries. It can

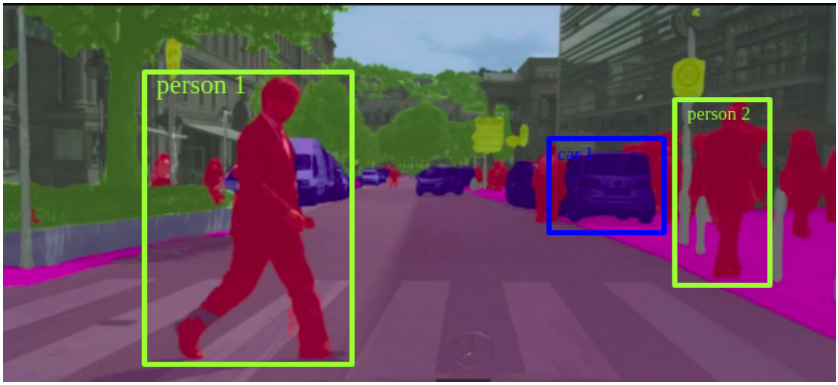


Figure 19: An Example of Instance Segmentation.

separate two or more overlapping objects of the same category. The instance level segmentation of the image 18 is shown in Figure 19 where different objects of the same category like persons are detected as *person 1*, *person 2* etc. Another example differentiating semantic and instance segmentation is also illustrated in Section 7.2.

Mask RCNN Model

Mask RCNN (MRCNN) is the state-of-art instance segmentation model originally proposed by He et al. (HGDG17). It is a two stage object detection and segmentation framework (shown in Figure 20) where the first stage scans the image and generates proposals using the Region Proposal Network (RPN) and the second stage classifies the proposals and generates bounding boxes and segmentation masks of the objects. Mask RCNN extends its predecessor Faster RCNN (RHGS15) by adding an extra head to architecture for predicting segmentation masks on each Region of Interest (ROI) generated by the RPN network, in parallel to the existing branches of Faster RCNN for classification and bounding box regression.

The first stage of Mask RCNN contains a feature extractor and a RPN. The feature extractor is a convolutional neural network that provides feature maps as inputs to the RPN network. The feature extractor used in the Mask RCNN framework for the experiments of this thesis, is a ResNeXt-101-FPN (ResNeXt and FPN have been briefly explained ear-

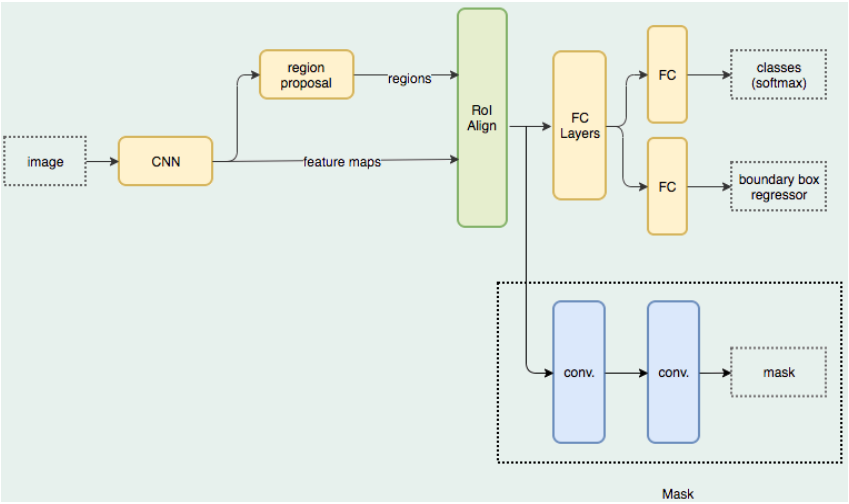


Figure 20: A schematic diagram of Mask RCNN illustrating the three different heads: classification head, bounding box regressor and the mask segmentation head.

lier in this chapter and their implementation in the contaminants problem is given in Section 7.3) network which is a combination of ResNeXt deep CNN and a FPN, a pyramidal feature extractor. The RPN is a lightweight neural network that scans the feature maps of the image in a sliding-window fashion and finds areas that contain the objects. The regions of interest scanned by the RPN are called as anchors which are defined as the bounding boxes of different sizes and aspect ratios, distributed over the feature map. The anchor selection criteria in the original works (HGDG17) and the framework used in the experiments of the thesis (discussed later in Section 7.3) are significantly different. Due to the bounding box refinement step in the RPN that generates the Regions of Interest (ROI) for the next stage of the Mask RCNN, there can be ROIs of different sizes and shapes. Mask RCNN uses a quantization free technique called as ROI Align to resize all these ROI regions to fixed sizes.

The next stage of the Mask RCNN contains a mask branch, a bounding box regression branch and an object classification branch. The mask head is decoupled from the class and the bounding box head and therefore, it is able to segment objects from each ROI in a class agnostic fash-

ion. The mask head is a fully convolutional neural network that can predict one mask per class independently from the other masks for every ROI. The entire Mask RCNN framework is trained end-to-end by minimizing the total training loss which is a multi-task loss function given as the summation of 5 losses: the RPN classification loss, RPN bounding box regression loss, the overall classification loss, the overall bounding box prediction loss and the mask segmentation loss as in Equation 4.3.

$$L = L_{R\text{-}cls} + L_{R\text{-}box} + L_{cls} + L_{box} + L_{mask} \tag{4.3}$$

where, $L_{R\text{-}cls}$ is the RPN classification loss, $L_{R\text{-}box}$ is the RPN box prediction loss, L_{cls} represents the classification loss, L_{box} represents the bounding box regression loss and L_{mask} represents the segmentation loss.

UNET Model

The UNET is an *U* shaped network developed by Ronnenberger et al. (RFB15) for biomedical image segmentation. The schematic diagram of the architecture is given in Figure 21.

The networks consists of a contraction path and an expansion path. The contracting path is a typical convolutional network that consists of repeated convolutions, each followed by a Rectified Linear Unit (ReLU) and a max pooling operation. During the contraction, the spatial information is reduced while feature information is increased. The expansive pathway combines the feature and the spatial information through a sequence of upsampling and concatenations with high-resolution features from the contracting path. The contraction path is also called as the encoder of the network that help in capturing the context from images whereas the expansion path is called as the decoder that enables precise localization. UNET network is trained end-to-end to perform semantic image segmentation and requires lesser amounts of labeled data than any two stage segmentation frameworks like Mask RCNN.

In the original works of UNET (RFB15), a pre-computed weight map is used and this is multiplied with the softmax weighted cross entropy loss to train the model. The pre-computed weight map helps the network to learn the small separation borders between the touching objects. The separation border is extracted using morphological operations and then the weight map is computed for each groundtruth segmentation mask by assigning higher weights to these borders. With the use of a weight

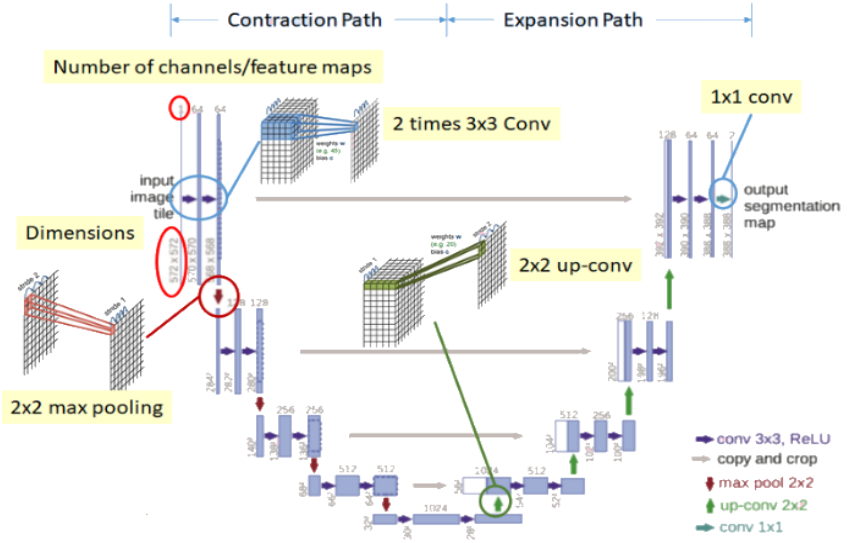


Figure 21: A schematic diagram of UNET based on its original works in (RFB15). The U-shpaed framework formed using a contracting and an expanding path of alternate convolution and pooling layers (RFB15).

map, a semantic segmentation framework is forced to behave closely to an instance segmentation framework. The weight map computation is done according to the Equation 4.4.

$$w(x) = w_c(x) + w_0 \cdot exp\Big(- \frac{(d_1(x) + d_2(x))^2}{2\sigma^2} \Big) \tag{4.4}$$

where c represents the number of classes in the segmentation task, w_c represents the weight map to balance the class frequencies, d_1 denotes the distance to the border of the nearest object and d_2 represents the distance to the border of the second nearest object.

4.3 Summary

To summarize this chapter, it outlines the original works of different deep learning frameworks and methods used and adapted in image/video

classification and image segmentation tasks in the following chapters of the thesis. Some of these frameworks are used as learning machinery integrated in SBR simulator for experimental analysis in Chapter 5 and Chapter 6, whereas others are used as image segmentation frameworks and backbone networks in Chapter 7.

Chapter 5

Image Classification

This chapter forms the core of the first part of the thesis and demonstrates image classification with different benchmark datasets performed using the simulators of SBR that are SBRS (Semantic Based Regularization Simulator : Caffe and C++ framework) and LYRICS (Learning Yourself Reasoning and Inference with Constraint Simulator: Tensorflow and Python framework). The goal of this work is to exploit the power of SBR when integrated with different deep CNN networks on both small and large scale image datasets in supervised and semi-supervised setting. For each of the datasets, different experimental conditions are set, different fuzzy logics are compared, different backbone networks are tested and most importantly optimization challenges are solved through different heuristics depending on the number and type of constraints used for the particular dataset. The hypothesis that integration of prior knowledge with rich feature learners like deep neural networks in scarcity of supervised examples is greatly beneficial, is proved through the different ablation studies described in this chapter. Therefore, this chapter provides empirical evidences to support the usefulness of all the novel theoretical additions of the SBR framework made through this research work.

An overview of the datasets used in this chapter are tabulated in Table 4. The four datasets namely Winston Animal, CIFAR-10, CIFAR-100 and ImageNet datasets are used with SBR framework to perform image classification using logical constraints. # in the Table 4 refers to the total number of samples or images present in the respective dataset.

Table 4: Datasets used in experiments of SBR. # Samples refer to the number of training patterns present in each corresponding dataset.

Dataset Name	# Samples	Categories	Domain
Winston ANIMAL	5005	7	Image
CIFAR-10 [1]	60K	10	Image
CIFAR-100 [1]	60K	100	Image
ImageNet [2]	1.2M	1000	Image

5.1 Simulators and Metrics

Semantic Based Regularization Simulator (SBRS) is a software implementing SBR in C++ and Caffe. Neural network learners are integrated in SBRS using the deep learning package, Caffe. It can be used to express first-order logical theory in Prenex Normal Form and declare domains with different sorts of constants, predicates and functions and hence can be used in any classical classification problem.

LYRICS is also another SBR based simulator which defines a declarative language in Python and has a Tensorflow based environment that drops the barrier to build very deep and complex neural network models and also integrates the available domain knowledge in any machine learning tasks including classification, generative and adversarial ML, sequence to sequence learning etc.

All the experiments of the ANIMAL dataset and the collective classification experiments using pretrained models of CIFAR-10, CIFAR-100 and ImageNet and also joint training in small convolutional neural networks like Resnet-20 are done in SBRS. Whereas all the joint training experiments involving deep networks are performed in LYRICS as Tensorflow helps in efficient integration of these frameworks into SBR. Different ablation studies are performed to show how to use the SBR framework with scarce data points in semi-supervised settings or in transductive learning tasks, how to heuristically add logical constraints yielding better classification results, how to implement collective classification over the test set etc. The wide range of the experimental evaluations performed is one of the biggest strength of this thesis.

To measure the performance of the multi-label semi-supervised image classification task, different metrics are used as listed below:

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`

[2] `http://www.image-net.org/`

- *Accuracy :* Metric for the final or leaf classes in the classification task that are exclusive and only one single class can trigger for each pattern or datapoint (selected via argmax).

- *Error rate :* Metric obtained by subtracting accuracy from 100 and expressed in terms of percentage.

- *Precision :* In multi-label classification, this metric tells about the proportion of positive classifications that are actually correct. It also gives an intuition of the false alarms in the classification task.

- *Recall :* In multi-label classification, this metric tells about the proportion of actual labels that are classified correctly. It gives an intuition of the misses with respect to the actual labels in the classification task.

- *F1-score :* Defined as the harmonic mean of precision and recall. It corresponds to the prediction of all the predicated classes including the intermediate ones. Considering all the classes, the F1 metric ($F1 = \frac{2 \times precision \times recall}{precision + recall}$) is used to measure the performance with a 0.5 threshold over the outputs.

5.2 Experimental Analysis on ANIMAL Dataset

This image classification problem is based on the original benchmark proposed by P.Winston (WH86), which was initially designed to show the ability of logic programming for the determination of the class of an animal from some partial clues. Unlike in the original challenge, no clues are given as input to the classifier, but only the raw images are provided, leaving it to the convolutional neural network (used as a learner in SBR) the responsibility of developing the intermediate clues over which to perform the inference. The ANIMAL dataset has been previously used by Diligenti et al. (DGS16) in kernel machine integrated SBR. Kernel Machines (KM) are shallow networks and cannot act as automatic feature extractors like neural networks, these experimental evaluations will empirically compare the previous results using KM with the neural network integrated SBR.

The dataset is composed of 5005 images, a small subset from the ImageNet database equally divided into 7 categories, each one representing an animal class namely:

- ALBATROSS
- CHEETAH
- GIRAFFE
- OSTRICH
- PENGUIN
- TIGER
- ZEBRA

The images have been split into training, validation and test sets with 3325, 710 and 910 images respectively. The effect of logical constraints in different semi-supervised settings are demonstrated through different experimental simulations described in the Section 5.2.4. The experiments are also performed by incremental addition of constraints using the optimization heuristics where convex constraints are added before the non-convex constraints.

The images of each animal present in the dataset occur at different distances, angles, magnifications and poses. All images used in these experiments have been resized to two different resolutions: 32×32 and 64×64 pixels.

5.2.1 Knowledge Domain

The knowledge about the domain is expressed in terms of first order logic rules as shown in Table 5. The rules are divided into two groups:

- The first set of rules are the ones that can be converted into convex constraints as described in the Section 3.4.2. Let these rules be called as *ConvR* denoting the convex rules.
- The second set of rules are the ones that are not guaranteed to be converted into convex constraints. Let this set be named as *ConcR* denoting the concave rules.

Different experiments are performed. Some of them use only ConvR rules where all the constraints are guaranteed to be convex using any t-norms and others use the whole set (ConvR and ConcR rules) where convex as well as non-convex constraints are present depending on the

Table 5: Prior knowledge used to train neural networks on Winston Benchmark.

ConvR rules referring to the rules generating Convex constraints.

$\forall x$ HAIR$(x) \Rightarrow$ MAMMAL(x)
$\forall x$ MILK$(x) \Rightarrow$ MAMMAL(x)
$\forall x$ FEATHER$(x) \Rightarrow$ BIRD(x)
$\forall x$ FLY$(x) \wedge$ LAYEGGS$(x) \Rightarrow$ BIRD(x)
$\forall x$ MAMMAL$(x) \wedge$ MEAT$(x) \Rightarrow$ CARNIVORE(x)
$\forall x$ MAMMAL$(x) \wedge$ POINTEDTEETH$(x) \wedge$ CLAWS$(x) \wedge$ FORWARDEYES$(x) \Rightarrow$ CARNIVORE(x)
$\forall x$ MAMMAL$(x) \wedge$ HOOFS$(x) \Rightarrow$ UNGULATE(x)
$\forall x$ MAMMAL$(x) \wedge$ CUD$(x) \Rightarrow$ UNGULATE(x)
$\forall x$ MAMMAL$(x) \wedge$ CUD$(x) \Rightarrow$ EVENTOED(x)
$\forall x$ CARNIVORE$(x) \wedge$ TAWNY$(x) \wedge$ DARKSPOTS$(x) \Rightarrow$ CHEETAH(x)
$\forall x$ CARNIVORE$(x) \wedge$ TAWNY$(x) \wedge$ BLACKSTRIPES$(x) \Rightarrow$ TIGER(x)
$\forall x$ UNGULATE$(x) \wedge$ LONGLEGS$(x) \wedge$ LONGNECK$(x) \wedge$ TAWNY$(x) \wedge$ DARKSPOTS$(x) \Rightarrow$ GIRAFFE(x)
$\forall x$ BLACKSTRIPES$(x) \wedge$ UNGULATE$(x) \wedge$ WHITE$(x) \Rightarrow$ ZEBRA(x)
$\forall x$ BIRD$(x) \wedge \neg$ FLY$(x) \wedge$ LONGLEGS$(x) \wedge$ LONGNECK$(x) \wedge$ BLACK$(x) \Rightarrow$ OSTRICH(x)
$\forall x$ BIRD$(x) \wedge \neg$ FLY$(x) \wedge$ SWIM$(x) \wedge$ BLACKWHITE$(x) \Rightarrow$ PENGUIN(x)
$\forall x$ BIRD$(x) \wedge$ GOODFLIER$(x) \Rightarrow$ ALBATROSS(x)
$\forall x$ LAYEGGS$(x) \Rightarrow$ BIRD(x)

ConcR rules referring to the rules that are not guranteed to be Convex constraints.

$\forall x$ MAMMAL$(x) \oplus$ BIRD(x)
$\forall x$ HAIR$(x) \oplus$ FEATHER(x)
$\forall x$ DARKSPOTS$(x) \Rightarrow \neg$ BLACKSTRIPES(x)
$\forall x$ BLACKSTRIPES$(x) \Rightarrow \neg$ DARKSPOTS(x)
$\forall x$ TAWNY$(x) \Rightarrow \neg$ BLACK$(x) \wedge \neg$ WHITE(x)
$\forall x$ BLACK$(x) \Rightarrow \neg$ TAWNY$(x) \wedge \neg$ WHITE(x)
$\forall x$ WHITE$(x) \Rightarrow \neg$ BLACK$(x) \wedge \neg$ TAWNY(x)
$\forall x$ WHITE$(x) \Rightarrow \neg$ BLACK(x)
$\forall x$ WHITE$(x) \Rightarrow \neg$ TAWNY(x)
$\forall x$ BLACK$(x) \Rightarrow \neg$ WHITE(x)
$\forall x$ BLACK$(x) \Rightarrow \neg$ TAWNY(x)
$\forall x$ TAWNY$(x) \Rightarrow \neg$ BLACK(x)
$\forall x$ TAWNY$(x) \Rightarrow \neg$ WHITE(x)
$\forall x$ CHEETAH$(x) \oplus$ TIGER$(x) \oplus$ GIRAFFE$(x) \oplus$ ZEBRA$(x) \oplus$ OSTRICH$(x) \oplus$ PENGUIN$(x) \oplus$ ALBATROSS(x)

choice of the t-norms used. Seven predicates in the knowledge base as listed in the Table 5 correspond to the final animal classes, while the others are intermediate predicates that help in determining the final classes during the inference process. For example: HAIR, MAMMAL, MILK, CARNIVORE etc. are intermediate predicates whereas CHEETAH, OSTRICH are final predicates. The rules are handcrafted based on prior knowledge about the animal images. Almost all the rules in ConvR are definitive rules which involve a residuum operator or a final class in it. Most of the rules in ConcR set that involve an intermediate class or predicate are called intermediate rules. The last rule in ConcR set with an exclusive-OR operator states the fact that one and only one class should be assigned to each image.

Rules are converted to constraints using different t-norms as mentioned in Table 1 in Chapter 3 using the Lukasiewicz fragment, the constraints are convex under certain conditions. Whereas when other t-norms like Product and Minimum are used that does not guarantee convexity, optimization heuristics as described in Section 3.4.2 are employed during learning for better and faster convergence. In the experimental process, the rules are added incrementally (rules of ConvR set followed by the ConcR set) to study the effect of prior knowledge and also to demonstrate the effect of convexity of constraints.

5.2.2 Experimental Settings

The learning problem is also studied in different experimental settings:

- Transductive Fully Labeled: All the training, the validation and the test patterns are available during training but only the class labels of the training patterns are fully known a priori. FOL groundings however, can contain training, validation and test data and the constraints can be enforced over all the available data patterns. Transductive learning has several practical applications in the field of document classification, image classification from web where reading through the content and deciding on the subject or category of millions of documents or images is a tedious task to be executed manually in order to obtain labeled data. Therefore, using a small subset of labeled data in combination with a large set of unlabeled data, creating a semi-supervised environment for transductive learning and then exploiting the domain knowledge is a very feasible solution.

- Transductive Partially Labeled: All the patterns are available during training in the transductive setting as described above but transductive partially labeled differs from its fully labeled counterpart, as here a random subset of class labels are available for each pattern during training. This emulates a scenario that could have happened on an image hosting site, where images are tagged by the users of the social media, but the tags are incomplete and inconsistent, therefore, it is required to fill the missing tags by learning and inference. The other real world applications are present in the areas of anomaly detection, or in the detection of operating modes in industrial equipment (pumps, turbines, etc.). For this experimental setting, in particular, one third of the labels are randomly selected to be kept as the training examples, while the others go to the validation and the test sets.

- Non Transductive: The training data is composed of only the training patterns and all the training patterns are fully labeled during training. The validation and the test patterns do not form a part of the training data. The constraints are therefore expressed only on the training patterns during training. For a non transductive setting, when the labels are poorly annotated or are noisy, using of logic rules can be beneficial. This happens in social network websites where the labels associated with social images are valuable source of information for tasks of image annotation, understanding and retrieval but these labels are often found to be noisy, mainly due to the collaborative tagging activities of the users. Therefore, exploiting background knowledge in a fully labeled non transductive environment can help in designing a robust noise-free learning model.

- Collective Classification: Theoretically described in Section 3.4.1. In transductive context although constraints are already enforced on the during training of the model, but still there is no guarantee that the input representations are powerful enough to allow the neural networks to respect the constraints on the test data, therefore collective classification searches for the values of the functions in the test data respecting the FOL formulas while being close to the prior values established by the neural networks. As already stated in Section 3.4.1, since collective classification is applied during the inference time, it can be used to improve any sort of independent classification applications like image recognition in

photo organization apps, classifying inappropriate and offensive images automatically for social websites, classification of content in news/sports videos etc.

5.2.3 CNN Models

The CNN model[3], used for this dataset was previously one of the top performers on the CIFAR-10 benchmark. The considered model contains an input layer, followed by two convolutional layers with alternate pooling layers and fully-connected layers (which forms the output layers) making it a 6-layered neural network. The outputs from the first fully connected layer gives the predictions of the final class labels and therefore contain 7 output neurons for the 7 final classes. Additional output layers are added to classify the intermediate predicates and a sigmoid function is used in these classification layers. The activation functions in all the hidden layers are rectified linear units (ReLU), whereas a softmax or sigmoid function is used in the output layers. Max pooling (overlapping pooling with a kernel size of 3 and stride of 2) and local response normalization are applied to the output of each of the convolution layers. The first convolutional layer convolves the input image with 32 kernels of size $5 \times 5 \times 3$ and a stride of 1. The second convolutional layer takes pooled and response normalized output of the first convolutional layer and filters it again with 32 kernels of the same size as the previous ones. The weights of each of the convolutional layers and the fully connected layers are initialized from a zero mean Gaussian distribution with a standard deviation of 0.006 and 0.025 respectively. The neuron biases are initialized with a constant value of 1.

5.2.4 Results

The general goal of these experimental simulations is to integrate logical constraints with the deep CNN network using all the constraints (convex and concave) to improve the classification accuracy over the kernel machines trained with constraints and over the baseline CNN network on this dataset. Although the CNN network used is a not a very deep network, but still there are optimization challenges when integrated with

[3] `http://www.github.com/BVLC/caffe/tree/master/examples/cifar-10`

constraints in SBR because of the presence of the concave constraints. Using optimization heuristics to meet these challenges and obtain a faster convergence during training is an important contribution of this thesis. The benefits of these heuristics are demonstrated through these experimental evaluations. Another notable contribution of these experimental studies, is the usefulness of using logical constraints in the presence of scarce data examples. These studies are performed to highlight the fact that SBR is a powerful semi-supervised framework.

The experimental simulations are performed in transductive mode with fully and partially labeled examples and in non transductive mode with collective classification. During joint learning, convex constraints are generally added after 50 iterations followed by non convex constraints at a later stage. The simulations are done with constant weight based SBR (where the regularization values are constant for all predicates) and also with variable regularizers in vanilla SBR, both described in Section 3.3. The benefits of using variable regularizers is also demonstrated through the experimental results as it improves the accuracy over its constant regularizing counterpart. Collective classification, another important part of this work is also performed for 300 iterations for all the simulations.

To measure the performance of this image classification task in the SBR framework, accuracy over the final classes and F1 score over all the 33 predicates are reported. The performance of the CNN is compared with the kernel machine classifier using Gaussian kernels with SIFT image feature representations which was first proposed in Diligenti et al.(DGS16). The runtime, prediction time and the collective classification time in SBR are also compared as an indicator of the performance. The CNN model is trained in a GeForce 1080Ti GPU.

Experimental Simulation I :

This is a fully labeled transductive experiment on two different selection of image resolutions: 32×32 and 64×64 pixels. The training, validation and test patterns are all available during the training time and only the training patterns are fully labeled. There are 3325, 710 and 910 images in the training, validation and test sets respectively. The validation set is used for the selection of parameter values like λ_l and λ_h for a constant weight based SBR and values of λ_l^k and λ_h for a vanilla SBR.

Results are reported for predictions over the test data for all predicates using kernel machines and convolutional neural networks for dif-

Table 6: Accuracy over 7 final classes and F1 score for all classes for a Fully Labeled Transductive setting using shallow and deep classifiers with and without prior knowledge. #Pat: number of training patterns. KM and KM+R: kernel machine classifiers without and with rules respectively. CNN, CNN+R and CNN+R+CC: deep CNN models without rules, with rules and with collective classification respectively using ConvR(I) and ConcR(II) sets.

(32×32 pixel Images)								
	#Pat	KM	KM+R	CNN	CNN+R		CNN+R+CC	
					I	I + II	I	I + II
Acc	3325	87.2	87.9	92.2	93.0	93.3	93.5	94.1
F1		43.2	44.0	48.0	49.5	50.4	56.0	57.3
Acc	2450	80.1	81.4	83.6	89.7	90.8	89.9	90.9
F1		41.7	43.2	47.3	47.8	49.5	55.1	55.9
Acc	1076	78.3	79.4	83.0	83.7	84.2	83.8	84.9
F1		39.0	40.3	41.9	41.9	44.8	41.9	45.0
Acc	500	76.4	78.7	80.5	80.6	81.0	81.6	81.8
F1		30.4	30.9	31.9	31.9	32.0	31.9	32.0
Acc	100	64.0	65.0	67.2	67.7	68.2	67.7	68.4
F1		20.0	20.0	21.0	21.8	22.3	21.8	22.9
(64×64 pixel Images)								
	#Pat	KM	KM+R	CNN	CNN+R		CNN+R+CC	
					I	I + II	I	I + II
Acc	3325	89.6	89.9	95.7	95.8	96.4	96.9	97.2
F1		49.0	50.1	58.1	60.0	60.9	61.1	61.4
Acc	2450	81.7	82.6	85.2	91.7	92.0	92.4	93.9
F1		48.2	48.8	50.3	53.9	54.1	52.6	57.6
Acc	1076	79.9	80.7	83.4	83.9	84.9	84.7	85.2
F1		41.2	42.3	43.4	44.1	44.6	44.9	45.2
Acc	500	77.1	79.1	81.7	82.0	82.6	82.4	83.0
F1		33.3	33.5	34.2	34.9	35.3	34.7	35.9
Acc	100	65.2	65.9	68.4	68.9	69.0	69.6	70.0
F1		23.8	24.0	25.3	25.4	26.0	26.0	26.2

ferent resolutions of input images. F1 represents the F1 score over all predicates in the KB and Acc refers to the accuracy measured over only the 7 final classes. KM refers to the Kernel machines integrated in SBR, whereas KM + R refers to the rules integrated with Kernel machines. CNN, CNN + R refers to the convolutional neural networks without and with rules in SBR respectively. CNN + R + CC refers to the collective classification process which further boosts up the accuracy during test time over the corresponding neural network classification accuracy. I and II in Table 5 refer to the set of rules, ConvR and ConcR.

A performance improvement is observed when more rules are used. Addition of convex constraints followed by non-convex constraints yield faster convergence of the training loss and also improves the over all test accuracy when both the models are trained on equal number of iterations. Following the optimization heuristics, the rules that are guaranteed to translate into convex constraints are tactfully added earlier in the learning stage than the rules that do not guarantee convexity (ConvR set of rules added before the ConcR set). To further demonstrate the effect of convexity clearly, the experiment can be performed under different conditions:

- Without addition of any constraints : No constraints scenario.
- Addition of only the constraints that are guaranteed to be convex with any selection of t-norms : Rules of the ConvR set.
- Addition of all constraints at the beginning of the learning process : ConvR and ConcR rules are added without considering the importance of convexity.
- Addition of convex constraints followed by non-convex constraints using optimization heuristics : ConcR rules are added for further convergence in the learning process after ConvR rules. The heuristic of adding the ConcR rules at a later stage have aided in minimizing the loss during joint learning.

Table 6 reports the accuracy and F1 results of 32×32 and 64×64 pixel images using Weak Lukasiewicz t-norm as it is observed to perform better than the other t-norms. Weak Lukasiewicz fragment with weak connectives have a number of positive implications during optimization over Product and Minimum t-norms like it can be converted into PNF and has involutive negation (GDGM17). It is also observed

Table 7: Comparison of the collective classification accuracies for the 7 final classes for Fully Labeled Transductive setting with 32×32 and 64×64 images of Winston benchmark. Each row represents different subsets of training patterns used. WL and WL(λ_l^k): classification error rates for transductive training with rules using Weak Lukasiewicz t-norm with constant and vanilla SBR, CC-WL and CC-WL(λ_l^k): collective classification using Weak Lukasiewicz t-norm for constant and variable (λ_l^k) values respectively. The bold numbers in the table refer to the best performing t-norm with two different image resolutions.

#Pat	WL	CC-WL	WL(λ_l^k)	CC-WL(λ_l^k)
(32×32 pixel Images)				
3325	93.3	94.1	94.2	**94.4**
2450	90.8	90.9	91.2	91.4
1076	84.2	84.9	85.0	85.3
500	81.0	81.8	81.9	82.0
100	68.2	68.4	68.8	69.0
(64×64 pixel Images)				
3325	96.4	97.2	97.5	**97.8**
2450	92.0	93.9	94.0	94.1
1076	84.9	85.2	85.3	85.4
500	82.6	83.0	83.0	83.0
100	69.0	70.0	70.1	70.3

that neural networks outperform kernel machines under any combination of rules used. This confirms the usefulness of the main objective of this work, integrating neural networks in SBR. Deep neural networks are able to learn complex features compared to kernel machines are therefore are better trained models. It is also seen that collective classification improves accuracy at inference time over the results of independent classification.

To also demonstrate the performance of SBR and the effect of logical constraints with a scarcity of training data points (higher degree of semi-supervision), the neural networks are trained with different subsets of patterns like 2450, 1076, 500 and 100. Table 7 shows Weak Lukasiewicz t-norm integrated with rules applied on scarce training examples followed by collective classification in constant weight based SBR and vanilla SBR for two different image resolutions. The results demonstrate that when the number of training patterns decrease from 3325 to 2450, the effect of logical constraints is almost increased by $7\% - 8\%$. This also demon-

strates how helpful and effective is prior knowledge in scarcity of labeled examples in SBR. This demonstrates that SBR can be used efficiently in any practical computer vision applications to distinguish objects, facial expressions, food, natural landscapes, sports etc. having limited amount of supervised examples in combination with a larger set of unsupervised examples.

Although, in these experiments, a higher amount of increase in accuracy is predictable with further decrease of labeled examples, but probably this is not noticed as the accuracy drops during joint training with lesser datapoints because the network suffers from overfitting with training samples lower than 500. However, the comparative results of all the different t-norms (including Weak Lukasiewicz) are given in Table 53 and in Table 54 in Appendix A. For vanilla SBR, the λ_l^k values are chosen through cross-validation. In these comparisons, all the constraints are applied from the ConvR and ConcR sets following the optimization heuristics. In these tables, CC-WL and CC-WL (λ_l^k) refers to the collective classification error rates, whereas WL and WL (λ_l^k) refers to classification using rules with Weak Lukasiewicz t-norm in constant and vanilla SBR respectively. The error in classification is found to be considerably lowest when predicate based vanilla SBR is used with Weak-Lukasiewicz t-norm compared to any other setting. It is also observed that vanilla SBR framework performs better than its constant counterpart irrespective of the selection of the t-norm. This empirically shows that the novel contribution of exploiting variable regularizers during training in SBR, makes it a stronger framework for image classification problems.

The experimental simulation therefore draws in four important conclusions:

- Constraints show a greater effect with limited number of supervised examples, therefore confirming SBR as a semi-supervised learner.

- Using the optimization heuristics, that allows the addition of non-convex constraints after convex constraints, not only aid in faster convergence but also help to improve the accuracy during the inference time. This helps to gain the maximum advantage from the constraints.

- Weak Lukasiewicz logic performs better than Product and Minimum fuzzy logics in improving accuracies in normal as well as

in collective classification settings because of its advantages in providing greater proportion of convex constraints (comparison shown in Table 53 and in Table 54 in Appendix A).

- Vanilla SBR performs better than constant weight based SBR as the regularizers effect learning to a great extent. When the regularizers are applied depending on the output of the neural network learner, they help to correct the misclassified labels in a more efficient way with the help of rules, therefore increasing the overall accuracy of classification.

Experimental Simulation II :

This experiment simulates a partially labeled image classification task in the transductive setting on both constant and vanilla SBR. The split of the training, validation and test sets remain the same but all the data points along with only one third of random labels from the three sets are used in partially labeled transductive training. Results are reported with the same naming conventions as in the previous simulation in Table 8 for 32×32 and 64×64 pixel images respectively for constant weight based SBR with Weak-Lukasiewicz t-norm. The overall accuracy and F1-score are seen to be higher in partially labeled simulation for any subset of datapoints and for any types of learners than in fully labeled simulation. This is because in partially labeled transductive learning, there is a probability that some of the supervisions or labels of the test patterns are selected randomly during training. KMs trained in partially labeled setting tend to perform better than before and standard CNNs only slightly outperform the KMs by a thin margin. The integration of prior knowledge and the implicit inference mechanism triggered by the partial labels during training is still beneficial to CNNs both when restricting the analysis to the final classes and also when considering the intermediate ones.

Similar to the previous simulation, Table 9 shows the classification and collective classification results with Weak Lukasiewicz t-norm in partially labeled setting for constant and vanilla SBR with different subsets of training patterns.

The comparative results of all the different t-norms are given in Table 55 and Table 56 in Appendix A. This experimental simulation also confirms the same findings as demonstrated in the experimental simulation I. Thus all the factors that actually contribute to the performance gains are:

Table 8: Accuracy (Acc) for the 7 final classes and F1 for all classes in Partially Labeled Transductive setting using shallow and deep classifiers compared with and without prior knowledge. KM and KM+R : kernel machines trained without and with rules respectively. CNN, CNN+R, CNN+R+CC: baseline neural network, neural network with rules and collective classification on neural networks with rules.

(32×32 Images)								
	#Pat	KM	KM+R	CNN	CNN+R		CNN+R+CC	
					I	I + II	I	I + II
Acc	3325	87.6	91.8	92.8	93.3	94.2	95.7	96.1
F1		44.6	48.2	48.3	48.8	52.5	55.1	56.7
Acc	2450	82.8	84.9	85.0	91.0	91.2	91.9	92.1
F1		44.4	47.1	47.3	48.8	50.0	49.7	51.4
Acc	1076	78.7	80.9	83.1	83.9	84.2	84.1	85.0
F1		40.0	42.8	43.2	43.7	44.9	44.3	45.1
Acc	500	77.1	79.5	80.6	81.2	81.9	82.4	82.8
F1		32.7	32.9	33.1	33.3	34.6	33.8	35.0
Acc	100	65.1	67.1	68.2	68.8	69.2	69.6	70.4
F1		23.9	25.0	25.5	25.8	26.0	26.1	26.9

(64×64 Images)								
	#Pat	KM	KM+R	CNN	CNN+R		CNN+R+CC	
					I	I + II	I	I + II
Acc	3325	90.9	95.4	95.9	96.1	97.7	97.4	98.9
F1		51.7	57.2	57.4	59.8	60.0	61.4	62.6
Acc	2450	83.9	85.9	86.1	92.1	92.9	93.2	94.1
F1		48.4	49.1	51.3	54.7	55.7	55.0	58.4
Acc	1076	80.7	81.9	83.5	84.9	85.1	85.2	85.8
F1		43.0	45.9	46.7	46.9	49.1	49.9	50.5
Acc	500	78.0	79.9	81.2	81.7	81.9	82.3	83.0
F1		38.7	39.9	40.8	41.3	41.8	42.6	43.0
Acc	100	70.1	72.1	73.2	73.8	74.6	75.2	75.4
F1		31.9	32.0	32.5	32.8	33.1	33.0	33.9

Table 9: Comparison of the classification and collective classification accuracies for the 7 final classes for Partially Labeled Transductive setting with 32×32 and 64×64 images. WL, CC-WL: classification with rules and the collective classification error rates using Weak Lukasiewicz t-norm in constant SBR. WL(λ_l^k), CC-WL(λ_l^k): classification with rules and the collective classification with Weak Lukasiewicz t-norm in variable SBR.

#Pat	WL	CC-WL	WL(λ_l^k)	CC-WL(λ_l^k)
(32×32 pixel images)				
3325	94.2	96.1	96.3	**96.8**
2450	91.2	92.1	92.2	92.4
1076	84.2	85.0	85.3	85.6
500	81.9	82.8	82.9	83.0
100	69.2	70.4	70.5	70.7
(64×64 pixel images)				
3325	97.7	98.9	99.0	**99.2**
2450	92.9	94.1	94.5	94.9
1076	85.1	85.8	85.9	85.9
500	81.9	83.0	83.2	83.4
100	74.6	75.4	75.5	75.6

- Using Deep CNN learners as the learning machinery in SBR because of their ability to better learn representations from the input images.
- Performing collective classification during inference to enforce logical rules on the test predictions externally that is not applied by the trained model.
- Addition of constraints using optimization strategies that enables the usage of a larger set of domain knowledge than being limited to only the convex set.
- Using predicate dependent regularizers in vanilla setting that helps the learning model to regularize better.

Experimental Simulation III :

This experimental simulation represents a non transductive setting where the training data consists of only the training patterns and does not in-

Table 10: Accuracy of 7 final classes and F1 for all classes of 32×32 images using non transductive CNNs compared with and without prior knowledge. CNN and CNN+R+CC: baseline neural network and collective classification on neural networks with rules. I and II refers to the ConvR and ConcR set of rules.

	#Pat	CNN	CNN+R+CC	
			I	I + II
Acc	3325	92.2	92.4	92.9
F1		48.0	55.2	56.4
Acc	2450	83.6	87.9	88.1
F1		47.3	53.1	54.2
Acc	1076	83.0	83.3	83.5
F1		41.9	40.9	42.1
Acc	500	80.5	80.9	81.2
F1		31.9	30.9	31.8
Acc	100	67.2	67.5	68.0
F1		20.0	21.0	21.5

Table 11: Accuracy of the 7 final classes and F1 score for all classes in Non Transductive setting with 64×64 using CNNs compared with and without prior knowledge. CNN and CNN+R+CC: baseline neural network and collective classification on neural networks with rules. I and II refers to the ConvR and ConcR set of rules.

	#Pat	CNN	CNN+R+CC	
			I	I + II
Acc	3325	95.7	96.2	96.5
F1		58.1	60.4	61.1
Acc	2450	85.2	91.0	92.0
F1		50.3	52.2	56.4
Acc	1076	83.4	83.9	84.7
F1		43.4	43.9	44.0
Acc	500	81.7	81.9	82.1
F1		34.2	34.1	34.9
Acc	100	68.4	68.7	69.4
F1		25.3	25.0	25.5

clude the validation and test patterns. The supervisions are also provided only for the training patterns. Non transductive experiments are performed with image inputs of 32×32 pixels and 64×64 pixels. The sim-

Table 12: Comparison of the collective classification accuracies for the 7 final classes in Non Transductive setting with 32×32 and 64×64 images of Winston benchmark. CC-WL and CC-P: collective classification error rates with Weak Lukasiewicz and Product t-norms in constant and variable SBR. The bold numbers in the table refers to the highest accuracies with 3325 supervisions on two different image resolution using Weak Lukasiewicz t-norm.

#Pat	CC-WL	CC-WL(λ_l^k)	CC-P	CC-P(λ_l^k)
(32×32 pixel Images)				
3325	92.9	**93.9**	91.5	91.6
2450	88.1	90.1	87.9	88.0
1076	83.5	83.6	81.5	82.8
500	81.2	81.3	80.0	81.5
100	68.0	68.1	67.0	67.5
(64×64 pixel Images)				
3325	96.5	**96.6**	95.3	96.5
2450	92.0	93.1	91.9	92.5
1076	84.7	84.9	84.4	84.7
500	82.1	82.3	81.9	82.3
100	69.4	69.6	69.3	69.6

ulation results with constant weight based SBR using Weak-Lukasiewicz t-norm are reported in Table 10 and Table 11 respectively.

In the non transductive setting, all the patterns in the training set are fully supervised, therefore, using the logic rules during joint training simply replicate information conveyed by the supervisions. Hence, to see any boost in performance using logic constraints, collective classification needs to be performed. However, as stated earlier for real world applications, non transductive learning can also benefit from constraints without collective classification when the fully supervised examples have noisy or poorly annotated labels. But for this dataset, the labels are clean and therefore, the improvement of accuracy using rules is seen only when collective classification is applied.

To create a semi-supervised setting for the non transductive learning in SBR, ablation studies are performed with different subsets of data examples varying between 3325 to 100 patterns. These ablation studies will enable to see the benefits of using constraints in neural networks even in non transductive models as they would have a mix of supervised and unsupervised datapoints.

Table 13: Comparison of the Train, Prediction and Collective classification time with different subsets of training patterns in transductive and non transductive learning. CNN-T, CNN-P and CNN-CC: Train time, Prediction time (without SBR) and Collective classification time respectively for the baseline CNN network. CNNT-T, CNNT-P and CNNT-CC: Train time, Prediction time and Collective classification time of the neural network when in transductive learning mode in SBR. All the times are in seconds.

#Pat	CNN-T	CNN-P	CNN-CC	CNNT-T	CNNT-P	CNNT-CC
(32×32 pixel Images)						
3325	420	20	35	530	23	38
2450	260	20	35	460	23	38
1076	105	18	36	300	23	38
500	74	20	36	133	23	37
100	60	20	35	92	23	37
(64×64 pixel Images)						
3325	740	46	50	960	50	56
2450	455	45	50	350	49	56
1076	266	46	50	300	59	56
500	135	46	50	212	49	56
100	104	46	50	130	50	56

Comparisons are also made among Weak-Lukasiewicz and Product t-norms with different image resolutions in constant and vanilla non transductive SBR in Table 12. The comparisons with all other t-norms (including Weak Lukasiewicz and Product) are reported in Table 57 of Section A.0.1. Similar to the observations in transductive learning, the experimental simulation of non transductive learning also shows the benefits of the Weak Lukasiewicz fragment over other fuzzy logics and the benefits of using predicate dependent regularizers over constant regularizers.

5.2.5 Discussion

Through the extensive experimental evaluations on this dataset, it can be concluded that the CNNs in general outperform kernel machines on this image classification task, even when KMs are boosted using prior knowledge via SBR because of the fact that CNNs can learn better input representations than KM. This conclusion justifies the motivation of integrating deep neural networks in SBR and makes a significant contribution in the field of deep learning research. All the three experimental

simulations also confirms that integration of rules into deep learning is more beneficial for tasks with scarcity of data examples. The other important addition is the further boost of the accuracy by applying the rules via collective classification during the test phase. The vanilla SBR with Weak Lukasiewicz t-norm in a partially labeled transductive setting performs the best on this dataset. It obtains an image classification accuracy of **99.2%** which is approximately 4.3% higher than the CNN baseline and almost 11% higher than KM baseline for 64×64 pixel images. The best performing results on a 32×32 image is **96.8%** which also outperforms a KM by approximately 9% and a deep CNN baseline by almost 6.3%. Through this research work, the state-of-the-art results are achieved on this dataset thus providing solid evidence of the usefulness of applying domain knowledge. There are also different novel concepts that are implemented in these simulations like the vanilla setting in SBR and the optimization strategies while joint training of the CNNs.

It is also important to note that the performance gain using rules in SBR is obtained with negligible overhead of time. Table 13 demonstrates that the time taken to train the 6-layered neural network on this dataset is about 530 seconds using 3325 training patterns, 710 validation and 910 test patterns in the transductive mode and approximately 420 seconds when the baseline network is trained in the non transductive mode using 32×32 pixel input images. The 6.3% increase in the accuracy obtained over the baseline architecture using this model through collective classification uses only 38 extra seconds which is a negligible overhead if other alternatives (for example training the network with more supervised examples etc.) are to be considered. Also it is seen that the runtime without SBR and with SBR differs in about $3 - 4$ seconds which indicates that when the network is trained with rules, it still remains time efficient. Similar observation is also seen as reported in Table 13 when 64×64 pixel images are used.

5.3 Experimental Analysis on CIFAR-10 Dataset

This image classification task is based on the benchmark and popular dataset of the AI community known as CIFAR-10[4] (Canadian Institute For Advanced Research). CIFAR-10 is composed of $50,000$ training and $10,000$ test images. Each pattern in the dataset is a 32×32 pixel RGB natural image belonging to one of the 10 classes: *AIRPLANE*, *AUTOMOBILE*, *BIRD*, *CAT*, *DEER*, *DOG*, *FROG*, *HORSE*, *SHIP*, and *TRUCK*.

Using WordNet[5] and additional common knowledge, a total of 15 classes have been recursively mapped into the hypernomy (a more general concept) obtaining the hierarchical taxonomical structure as shown in Figure 22. Using logic constraints all these 15 classes are integrated into learning in SBR.

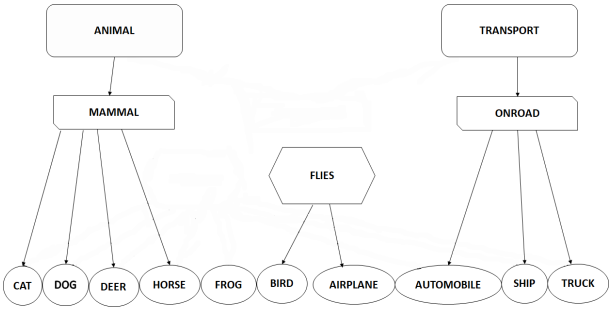


Figure 22: CIFAR-10 taxonomical hierarchy representing the intermediate and the final classes.

5.3.1 Knowledge Domain

The prior knowledge used for this task is shown in the Table 14. The knowledge base correlates the 10 output predicates (final classes of the dataset), each corresponding to a class to predict (for example the predicate DEER(x) will predict whether the pattern x is of the class *deer*), with 5 new predicates derived from the hierarchical information of WordNet and added to the knowledge base. These extra predicates therefore, form the intermediate classes in the classification task. They are learnt by the neural networks as extra labels on each pattern and therefore additional output layers with either a softmax or sigmoidal activation are used to classify these labels.

For example, one classifier layer predicts the predicates ANIMAL or TRANSPORT, one additional layer for the FLIES predicate and, finally, one output layer for the predicates ONROAD and MAMMAL. Since, each of the final classes either belongs to one of the intermediate classes among ANIMAL or TRANSPORT, therefore, these two classes are mu-

[5]https://wordnet.princeton.edu
[5]https://www.cs.toronto.edu/~kriz/cifar.html

Table 14: Prior knowledge used for the experiment simulations on the CIFAR-10 dataset.

$\forall x$ ANIMAL$(x) \vee$ TRANSPORT(x)
$\forall x$ TRANSPORT$(x) \Rightarrow$ AIRPLANE$(x) \vee$ SHIP$(x) \vee$ ONROAD(x)
$\forall x$ ONROAD$(x) \Rightarrow$ AUTOMOBILE$(x) \vee$ TRUCK(x)
$\forall x$ AIRPLANE$(x) \Rightarrow$ TRANSPORT(x)
$\forall x$ SHIP$(x) \Rightarrow$ TRANSPORT(x)
$\forall x$ ONROAD$(x) \Rightarrow$ TRANSPORT(x)
$\forall x$ AUTOMOBILE$(x) \Rightarrow$ ONROAD(x)
$\forall x$ TRUCK$(x) \Rightarrow$ ONROAD(x)
$\forall x$ ANIMAL$(x) \Rightarrow$ BIRD$(x) \vee$ FROG$(x) \vee$ MAMMAL(x)
$\forall x$ BIRD$(x) \Rightarrow$ ANIMAL(x)
$\forall x$ FROG$(x) \Rightarrow$ ANIMAL(x)
$\forall x$ MAMMAL$(x) \Rightarrow$ CAT$(x) \vee$ DEER$(x) \vee$ DOG$(x) \vee$ HORSE(x)
$\forall x$ CAT$(x) \Rightarrow$ MAMMAL(x)
$\forall x$ DOG$(x) \Rightarrow$ MAMMAL(x)
$\forall x$ DEER$(x) \Rightarrow$ MAMMAL(x)
$\forall x$ HORSE$(x) \Rightarrow$ MAMMAL(x)(x)
$\forall x$ OTHERS$(x) \Rightarrow$ BIRD$(x) \vee$ FROG$(x) \vee$ SHIP(x) AIRPLANE(x)
$\forall x$ BIRD$(x) \Rightarrow$ OTHERS(x)
$\forall x$ FROG$(x) \Rightarrow$ OTHERS(x)
$\forall x$ SHIP$(x) \Rightarrow$ OTHERS(x)
$\forall x$ AIRPLANE$(x) \Rightarrow$ OTHERS(x)
$\forall x$ FLIES$(x) \Rightarrow$ BIRD$(x) \vee$ AIRPLANE(x)
$\forall x \neg$ FLIES$(x) \Rightarrow$ CAT$(x) \vee$ DOG$(x) \vee$ HORSE$(x) \vee$ DEER$(x) \vee$ TRUCK(x)
$\vee$ SHIP$(x) \vee$ AUTOMOBILE$(x) \vee$ FROG(x)
$\forall x$ CAT$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ DOG$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ HORSE$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ DEER$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ TRUCK$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ SHIP$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ AUTOMOBILE$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ FROG$(x) \Rightarrow \neg$ FLIES(x)
$\forall x$ BIRD$(x) \Rightarrow$ FLIES(x)
$\forall x$ AIRPLANE$(x) \Rightarrow$ FLIES(x)

tually exclusive. Hence the classifier layer that classifies each pattern as either an ANIMAL or a TRANSPORT, has a softmax activation on it.

Thus the prior knowledge is enforced and integrated in deep learners by structing this as a multi-label classification problem in the SBR frame-

Table 15: Deep CNN architectures and the data pre-processing techniques used in CIFAR-10

CNN	PR+AUG
NIN-50	ZCA + Global Contrast Normalization
Resnet-20 Resnet-32 Resnet-44 Resnet-56 Resnet-110 Resnet-164 Resnet-1202	4-Pixels Padding + Random Crop + Horizontal Flip
PreResnet-110 PreResnet-164 PreResnet-1202	4-Pixels Padding + Random Crop + Horizontal Flip

work. The knowledge enforces the semantic consistency among the trained classifiers. In the above knowledge base most of the rules have a residua operator and hence the proportion of convex constraints generated are significantly greater. Also, using the Lukasiewicz and Weak Lukasiewicz convex fragment, a lot of convex functional constraints are also generated.

5.3.2 CNN Models

The experimental settings used for this classification task are a subset of the settings described in Section 5.2.2 of the Winston Animal classification problem. In transductive fully labeled, the neural network learners are jointly trained from scratch along with the rules in SBR which provides a strong integration of the logic constraints with supervised examples.

Whereas in non transductive setting, pre-trained neural networks from their original works are retrained only for the additional classifier layers and a collective classification is performed. In order to exactly reproduce the baseline results of different deep neural networks, their publicly available implementations are trained with no change in their architecture and training procedure except for the fully connected output layers. The deep neural networks used on CIFAR-10 are listed below as:

- *Network in Network (NIN)* model (LCY13) with 50 layers.
- *Residual Networks* (HZRS15a) with $20, 32, 44, 56, 110, 164$ and 1202 layers.
- *Pre-activated Residual* networks (HZRS16b) with $110, 164$ and 1202 layers.
- *Pyramidal Residual* networks (HKK16) with $110, 164, 200, 236$ and 272 layers.

All these networks have been trained from scratch for 150 or 300 epochs (depending on the network size) on 15 classes of CIFAR-10 using an initial learning rate of 0.1. A weight decay of 0.001, a momentum of 0.9 and also a batch normalization are used. No dropouts are used except for the NIN network, to be consistent with the original works. Simple data augmentation techniques including translation and 4 pixels padding on each side followed by a 32×32 random crop of the padded image and its horizontal flip as described in Table 15. In Table 15, PR + AUG is abbreviated for preprocessing and augmentation.

The NIN model have been trained using the same configuration as used in the seminar paper (LCY13), including dropout, global contrast normalization and ZCA whitening. All other parameters are same as in the original works of Resnets (HZRS15a; HZRS16b; HKK16).

The parameters of λ_l and λ_l^k of collective classification of a constant weight based SBR and vanilla SBR respectively, are selected via cross validation using the procedure described in Section 3.4.2. In vanilla SBR, the value of the λ_l^k for each predicate is chosen heuristically based on a simple assumption that if the target class works well and have a high classification accuracy from the CNNs, fewer corrections are required using prior knowledge, as the deviations from the actual targets is minimal, therefore, a high regularizer (λ_l^k value greater than 0.5) value is chosen and vice versa. For example, the target classes like AIRPLANE, BIRD and FROG have low classification accuracies using Resnets and PreResnets, therefore, low regularizers ($\lambda_l^k = 0.2$) are chosen for these classes, whereas SHIP has the highest classification accuracy and therefore $\lambda_l^k = 0.9$ is chosen.

During collective classification, the rules in the knowledge base represented in Table 14 are converted into constraints using different t-norms depending on the experimental setting. Collective classification is performed for 300 iterations with a learning rate of 0.01, an Adam optimizer

and a value of λ_h as 0.1. Also as described in Section 3.4.2, optimization heuristics are applied by adding the convex constraints at the beginning of the training followed by the other constraints.

5.3.3 Results

The main goal of the experimental simulations for this dataset remains the same as of the previous one, but here the comparisons are done between the logic integrated CNN networks and their corresponding baseline networks and not with kernel machines. In the previous works of SBR integrated with kernel machines, only small datasets like Winston Animal consisting of few thousand images were explored. For the first time, this research work does exhaustive experimental simulations with $50,000$ datapoints and also handle deeper networks. Several deep CNN architectures like Resnets, PreResnets etc. are evaluated against each other using different fuzzy logics. The contributions of variable regularizers and collective classification is also emphasized through these simulations for all the CNN networks used on this dataset however, their needs and roles remain identical as described for the Winston Animal dataset. The results of the two different experimental simulations using different t-norm settings: Non Transductive and Transductive Fully Labeled with collective classification are discussed below.

Experimental Simulation I :

For CIFAR-10 dataset, the first experimental simulation is a non transductive setup with collective classification. Out of the $50,000$ supervised datapoints, $45,000$ samples are used for training and the remaining 5000 datapoints are used in the validation set. This division is a standard while using this benchmark dataset.

In a non transductive setting the neural networks are trained with supervised examples and then rules are applied in collective classification during inference. Although, the intermediate predicates are used in the rules during collective classification, the accuracies are always reported on the final 10 classes.

Table 16 compares the collective classification error rates obtained on the test set from the trained CNN models. The comparison of the collective classification error rates using all the t-norms that are used in SBR is reported in Table 58 in Appendix A. The naming conventions like CC-WL and CC-P represents the collective classification errors for two

Table 16: Error rate for the 10 final classes for different deep architectures in non transductive learning with collective classification over the network outputs using different selection of t-norms with constant (λ_l) and predicate dependent metaparameters, λ_l^k. CC-WL and CC-P: the collective classification error rates using Weak Lukasiewicz and Product t-norms. The bold numbers in the tables refer to the best performing network or the ones where the improvement is highly remarkable.

Models	CNN	CC-WL	CC-WL(λ_l^k)	CC-P	CC-P(λ_l^k)
NIN-50	8.8	8.7	8.7	8.8	8.7
Res-20	8.8	8.0	7.9	8.5	8.0
Res-32	7.5	7.1	6.9	7.1	6.9
Res-44	7.2	6.6	6.3	6.8	6.5
Res-56	6.9	6.4	6.0	6.5	6.3
Res-110	6.6	6.0	5.5	6.3	6.0
Res-164	5.9	5.8	5.5	6.0	5.8
Res-1202	**7.9**	5.2	**5.1**	5.2	5.1
PRes-110	6.4	6.2	6.1	6.2	6.1
PRes-164	5.5	5.2	5.2	5.3	5.3
PRes-1202	**6.9**	6.1	**6.0**	6.1	6.1
APN-110, α=48	4.6	4.6	4.4	4.6	4.5
APN-110, α=84	4.3	4.2	4.1	4.2	4.2
APN-110, α=270	3.7	3.6	3.6	3.7	3.7
APN-164, α=48	4.2	4.0	4.0	4.1	4.0
APN-164, α=84	4.0	4.0	3.7	4.0	3.7
APN-164, α=270	3.9	3.4	3.4	3.5	3.4
APN-200, α=240	3.5	3.4	3.4	3.4	3.4
APN-236, α=220	3.4	3.4	3.3	3.4	3.5
APN-272, α=200	3.3	3.2	**3.1**	3.3	3.2
MPN-110, α=8	4.5	4.4	4.3	4.4	4.4
MPN-110, α=27	4.1	3.9	3.8	4.0	3.8

different selection of t-norms namely Weak Lukasiewicz and Product respectively. The CNN column refers to the baseline models where inference is performed without any prior knowledge. The accuracy values of these models are replicated from the original works of the corresponding CNN architectures. λ_l and λ_l^k represents the regularizer parameters for constant and vanilla SBR respectively.

The name of the deep CNN architectures are abbreviated to fit in the Table 16 as : NIN abbreviated for Network in Network model, Res

for Resnets, PRes for Pre-activated resnets, APN and MPN for Additive Pyramid Resnets and Multiplicative Pyramid Resnets respectively.

When these neural networks are trained from scratch, depending on the size and the complexity of the networks, they take around $1-4$ hours for training with 45000 data examples. However, collective classification for 300 iterations takes only about $3-15$ minutes during inference on test patterns which is a negligible overhead. The detailed comparison of the training time, prediction time and collective classification time for each network configuration of the different experimental simulations is given in Table 20. Collective classification always pushes the error rates considerably below their individual counterparts thus reconfirming its usefulness.

From the results of different CNN architectures in the Table 16, it is seen that the Weak Lukasiewicz t-norm outperforms the Product t-norm (other t-norms are reported in Table 58 in Section A.0.2), confirming the theoretical results presented in (GDGM17) which describes the optimization advantages of Weak-Lukasiewicz t-norm.

Using predicate dependent regularizers also consistently improves the results in all tested architectures. Improvements are especially large for very deep networks like the 1202 layered Resnet and PreResnet which showed significant performance degradation in the original works due to overfitting. These are architecturally large networks and only 45000 examples are not sufficient for training, and therefore they generalize poorly during inference time. Logic constraints suffice for the misclassifications made during the test time by these very deep trained CNN models. However, the proposed methodology brings a remarkable improvement as the error rate is dropped from 7.9% to 5.1% in Resnet-1202 network and from 6.9% to 6.0% in PreResnet-1202 network using Weak-Lukasiewicz t-norm.

The Additive Pyramidal Resnet with 272 layers is the best performing architecture among all others and it achieves an error rate of only 3.1% on CIFAR-10, which to our best knowledge was state-of-the art when the experiments were performed and published in the works of Roychowdhury et al. (RDG18).

An ablation study is performed using scarce data examples during training. 10%, 20% and 50% of the training samples are used during training to see the effect of logical constraints in semi-supervised setting. The remaining training data is used as unsupervised data during training. Table 17 shows the classification error rates with a few selected deep neural network models using variable percentage of training data using

Table 17: Comparison of the collective classification error rates of the 10 final classes for two different selection of t-norms using scarce training data in non transductive mode.% Data: percentage of supervisions used in each deep network. CNN, CC-WL and CC-P: neural network baseline, neural network outputs improved with collective classification using Weak Lukasiewicz and Product t-norms respectively.

Models	%Data	CNN	CC-WL	CC-WL(λ_l^k)	CC-P	CC-P(λ_l^k)
NIN-50	10	27.9	27.1	26.7	27.8	26.7
	20	21.2	20.4	20.4	20.5	20.5
	50	13.1	13.0	12.9	13.0	13.0
Res-20	10	27.1	26.4	26.2	26.5	26.3
	20	20.7	19.2	19.0	19.5	19.2
	50	12.5	12.3	12.1	12.4	12.1
Res-32	10	26.6	25.5	25.1	25.6	25.1
	20	18.6	17.9	17.7	17.9	17.8
	50	12.1	12.0	11.9	12.1	12.0
Res-44	10	25.4	25.1	25.0	25.2	25.2
	20	19.0	17.9	17.4	17.9	17.5
	50	12.1	11.9	11.8	12.0	11.9
Res-56	10	25.2	25.0	24.9	25.0	24.9
	20	16.4	15.9	15.9	16.0	15.9
	50	11.9	11.4	11.1	11.5	11.2
Res-110	10	24.8	24.7	24.1	24.7	24.1
	20	15.1	15.0	14.8	15.1	14.9
	50	10.1	10.0	9.9	10.0	10.0
PRes-110	10	24.0	23.8	23.6	23.8	23.7
	20	14.8	14.3	13.9	14.4	14.1
	50	10.0	9.9	9.8	9.9	9.8
APN-164, $\alpha = 270$	10	22.9	22.8	22.7	22.8	22.7
	20	13.6	13.4	13.3	13.4	13.3
	50	8.8	8.6	8.2	8.6	8.3

Weak-Lukasiewicz (CC-WL) t-norm and Product (CC-P) t-norm in constant and vanilla SBR. The comparison with all other t-norms is given in Table 59 in Section A.0.2.

It emerges that the reduction of the collective classification error rate is larger when the training data is scarce. For example: In Resnet-32, when only 10% of the supervised examples with logical rules are used, the error rate is reduced by 1.5% whereas using all the training samples and the rules reduces the error rate by 0.6% from its counterpart without logical rules. This experimental simulation on CIFAR-10 uses

Table 18: Error rate for the 10 final classes on CIFAR-10 for different deep architectures and applying collective classification over the network outputs using different selection of t-norms with constant and predicate dependent metaparameters (λ_l^k) in transductive mode. CNN, WL and CC-WL: neural network baseline, neural network outputs with rules and collective classification outputs using Weak Lukasiewicz t-norm respectively. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	WL	WL(λ_l^k)	CC-WL	CC-WL(λ_l^k)
NIN-50	8.8	8.2	8.2	8.1	8.1
Res-20	8.8	7.7	7.5	7.4	7.1
Res-32	7.5	6.8	6.7	6.6	6.5
Res-44	7.2	6.2	6.1	6.0	5.9
Res-56	6.9	6.2	6.1	5.9	5.8
Res-110	6.6	5.5	5.3	5.1	4.9
Res-164	5.9	5.2	5.1	4.9	4.7
Res-1202	7.9	4.9	4.8	4.7	4.5
PRes-110	6.4	5.8	5.8	5.8	5.7
PRes-164	5.5	5.0	5.0	4.9	4.9
PRes-1202	6.9	5.8	5.8	5.7	5.6
APN-110, α=48	4.6	4.0	3.9	3.8	3.7
APN-110, α=84	4.3	3.6	3.6	3.6	3.5
APN-110, α=270	3.7	3.0	3.0	2.9	2.8
APN-164, α=48	4.2	3.6	3.6	3.5	3.4
APN-164, α=84	4.0	3.1	3.0	2.8	2.7
APN-164, α=270	3.9	2.9	2.9	2.8	2.7
APN-200, α=240	3.4	2.8	2.8	2.7	2.6
APN-236, α=220	3.4	2.8	2.8	2.7	2.6
APN-272, α=200	3.3	2.7	2.7	2.6	**2.5**
MPN-110, α=8	4.5	3.9	3.8	3.7	3.6
MPN-110, α=27	4.1	3.4	3.4	3.4	3.3

deep CNN networks that are harder to train than the one used above in the Winston Animal classification problem, but the power of SBR is still pronounced when supervised examples are used in combination with a large set of unsupervised images. Thus it can be concluded that logical constraints can contribute to a greater extent in a scarcity of supervised examples and therefore it reconfirms that SBR is a powerful tool for semi-supervised learning irrespective of the size of the dataset or the depth of

the CNN networks.

Experimental Simulation II:

This simulation represents semi-supervised transductive learning such that all the patterns in the dataset (i.e $60,000$ datapoints) are available for training but only the supervisions of the training patterns ($45,000$ supervisions available) are used in training while the other supervisions are used for validation and test purposes.

Table 18 compares the classification and collective classification error rates for different selection of CNN models using Weak Lukasiewicz t-norm. The comparison results of all the t-norms is detailed in Table 60 in Section A.0.2. λ_l and λ_l^k represents the constant and vanilla SBR settings for the given selection of t-norms. It is observed that Weak-Lukasiewicz t-norm in vanilla SBR performs better even in transductive setting confirming the conclusions drawn from the findings in the non transductive mode and from the Winston Animal simulations.

Transductive learning uses all the validation and test patterns during training as unsupervised data in combination with the supervised training patterns and in this way it creates a semi-supervised learning environment as stated earlier. Here, the logical constraints play a role in the training of the model and therefore gains provided by the logical constraints can be seen even without collective classification. The columns WL and P shows the effect of using prior knowledge before collective classification, and these gains are obtained by just using the rules in joint learning. Collective classification and predicate based regularizers are used to further boost the performance gains over the gains obtained from joint learning.

Even in the transductive setting, Additive Pyramidal Residual network with 272 layers is the best performing architecture that reduces the error rate by 0.7% from its baseline using collective classification in vanilla SBR. It is also noticed that the error rate is dropped between 0.6% to 1.1% in many other deep CNN architectures. Thus, once again this confirms the power of prior knowledge in semi-supervised settings. When using scarce datapoints in the transductive learning mode, it is also seen the same behavior as noticed in the non transductive mode. The observations are given in Table 19 demonstrating that the reduction of the error rate is larger when the training data is scarce. For example: In transductive learning, Resnet-32, using 10% of the supervised examples and logical rules, the rules have a huge impact during training as well as

Table 19: Transductive classification and collective classification error rates of the 10 final classes for Weak Lukasiewicz t-norm using scarce training data. % Data: percentage of supervisions used in each deep network. CNN, WL, WL(λ_l^k), CC-WL and CC-WL(λ_l^k): neural network baseline, classification outputs for transductive training, collective classification output using Weak Lukasiewicz with constant and predicate dependent regularizers respectively.

Models	%Data	CNN	WL	WL(λ_l^k)	CC-WL	CC-WL(λ_l^k)
NIN-50	10	27.9	26.9	26.8	26.7	25.8
	20	21.2	20.1	20.0	19.8	19.2
	50	13.1	12.8	12.6	12.0	11.8
Res-20	10	27.1	25.9	25.8	25.5	25.0
	20	20.7	19.0	18.9	18.5	18.2
	50	12.5	12.0	11.8	11.6	11.3
Res-32	10	26.6	25.0	24.8	24.6	24.1
	20	18.6	17.5	17.4	17.1	16.8
	50	12.1	11.8	11.7	11.4	11.1
Res-44	10	25.4	24.7	24.6	24.2	24.0
	20	19.0	17.5	17.4	17.0	16.9
	50	12.1	11.5	11.4	11.2	10.9
Res-56	10	25.2	24.6	24.5	24.3	24.0
	20	16.4	15.3	15.2	15.0	14.9
	50	11.9	11.0	10.8	10.5	10.2
Res-110	10	24.8	24.1	24.0	23.7	23.1
	20	15.1	14.7	14.5	14.2	14.1
	50	10.1	9.9	9.8	9.5	9.1
PRes-110	10	24.0	23.2	23.1	23.0	22.7
	20	14.8	14.0	14.0	13.9	13.4
	50	10.0	9.8	9.8	9.7	9.2
APN-164, $\alpha = 270$	10	22.9	22.5	22.4	22.2	21.8
	20	13.6	13.1	13.1	13.0	12.4
	50	8.8	8.4	8.2	8.1	8.0

with collective classification during inference. The collective classification error rate is reduced by 2.1% whereas using all the training samples and the rules, the error rate is reduced by 1% from its counterpart without logical rules.

Table 20: Comparison of the Train, Prediction and Collective classification time with different deep CNN networks in transductive and non transductive learning. NN-T, NN-P and NN-CC: Train time, Prediction time and Collective classification time respectively for the baseline neural network. NNT-T, NNT-P and NNT-CC: Train time, Prediction time and Collective classification time of the respective neural network when in transductive learning mode in SBR. All the times are in hours.

Models	NN-T	NN-P	NN-CC	NNT-T	NNT-P	NNT-CC
NIN-50	1.50	0.04	0.05	2.10	0.05	0.05
Res-20	0.83	0.01	0.03	1.05	0.02	0.04
Res-32	1.05	0.01	0.02	1.10	0.02	0.04
Res-44	1.20	0.02	0.04	1.33	0.03	0.06
Res-56	1.37	0.03	0.05	1.83	0.04	0.07
Res-110	2.5	0.08	0.12	2.80	0.09	0.15
PRes-110	2.45	0.07	0.10	2.80	0.07	0.12
APN-164, $\alpha = 270$	3.65	0.12	0.17	3.93	0.14	0.25

5.3.4 Discussion

To summarize the main contributions of this research work on CIFAR-10, it is observed that any deep CNN model with logical constraints outperforms the baseline counterparts in all conditions and especially when there is a scarcity of supervised data. Collective classification also shows its benefits and further amplifies the classification accuracies during inference in both transductive and non transductive modes. The collective classification time is a very small overhead considering the benefits it provides at the test time. The train, the prediction and the collective classification time for the different CNN architectures used for the above simulations are listed in Table 20. Such a thorough study of using logical constraints with several deep architectures, using collective classification in transductive and non transductive learning on CIFAR-10 is done for the first time through this research work.

The experimental results show that Weak Lukasiewicz t-norm having a lot of positive implications during optimization performs better than other t-norms. The function based regularizers in vanilla SBR with Weak Lukasiewicz t-norm in a transductive setting is the best performer on this dataset. It obtains an image classification error rate of **2.5%** which is approximately 0.7% lower than the CNN baseline in transductive mode and almost 0.8% lower than the CNN baseline in non transductive mode.

As described earlier, transductive learning is very useful for several real world situations like image tagging in image hosting sites, anomaly detection or detection of operating modes in industrial equipments, therefore the performance improvement obtained in SBR during transductive learning is also a great benefit of this work for practical problems.

The ablation study performed using $10\%, 20\%$ and 50% further confirms the benefits of applying domain knowledge in the scarcity of data examples and supports the hypothesis of this research work that SBR is an effective semi-supervised learner irrespective of the CNN model used as its backbone. The other notable observation is when the networks are very deep, consisting of more than 1000 layers, they tend to overfit and in these situations addition of background knowledge becomes particularly useful. This observation reveals an important aspect for training very deep convolutional neural nets. Through, this research work for the first time, the neural networks above 1000 layers are trained efficiently and this opens new pathways of success for the deep learning community.

Addition of more carefully handcrafted rules is expected to further improve the performance. Although the classification accuracy for the best performing network is significantly high (approximately 97.5%) and there is less scope of improvement in this dataset but more constraints can definitely help in further reducing the misclassification errors. The knowledge base used in CIFAR-10 experimental simulations, contains only few non-convex constraints, but this still confirms the hypothesis that whenever the set of non-convex constraints increases, the heuristics used in optimization can have a great impact on the performance.

5.4 Experimental Analysis on CIFAR-100 Dataset

The CIFAR-100 dataset is composed of 100 classes containing 600 images for each class (Kri09)[6]. The images are split into training and test set, such that there are 500 training and 100 test images per class, respectively. The validation set is built with 5000 random datapoints from the entire training set. In CIFAR-100, the number of images in each class is only one-tenth of that in CIFAR-10 dataset but the total number of images are equal. The 100 classes in this dataset are grouped into 20 superclasses determined using WordNet[7] hierarchy. Each image comes with a

[6] https://www.cs.toronto.edu/~kriz/cifar.html
[7] https://wordnet.princeton.edu

Table 21: The Super and the Fine classes in the CIFAR-100 dataset.

SUPER CLASSES	FINE CLASSES
AQUATIC MAMMALS	BEAVER, DOLPHIN, OTTER, SEAL, WHALE
FISH	AQUARIUM FISH, FLATFISH, RAY, SHARK, TROUT
FLOWERS	ORCHIDS, POPPIES, ROSES, SUNFLOWERS, TULIPS
FOOD CONTAINERS	BOTTLES, BOWLS, CANS, CUPS, PLATES
FRUITS AND VEGETABLES	APPLES, MUSHROOMS, ORANGES, PEARS, PEPPERS
HOUSEHOLD ELECTRICAL	CLOCK, KEYBOARD, LAMP, PHONE, TELEVISION
HOUSEHOLD FURNITURE	BED, CHAIR, COUCH, TABLE, WARDROBE
INSECTS	BEE, BEETLE, BUTTERFLY, CATERPILLAR, COCKROACH
LARGE CARNIVORES	BEAR, LEOPARD, LION, TIGER, WOLF
MAN-MADE OUTDOOR	BRIDGE, CASTLE, HOUSE, ROAD, SKYSCRAPER
NATURAL OUTDOOR SCENES	CLOUD, FOREST, MOUNTAIN, PLAIN, SEA
OMNIVORES AND HERBIVORES	CAMEL, CATTLE, CHIMPANZEE, ELEPHANT, KANGAROO
MEDIUM MAMMALS	FOX, PORCUPINE, POSSUM, RACCOON, SKUNK
INVERTEBRATES	CRAB, LOBSTER, SNAIL, SPIDER, WORM
PEOPLE	BABY, BOY, GIRL, MAN, WOMAN
REPTILE	CROCODILE, DINOSAUR, LIZARD, SNAKE, TURTLE
SMALL MAMMALS	HAMSTER, MOUSE, RABBIT, SHREW, SQUIRREL
TREE	MAPLE, OAK, PALM, PINE, WILLOW
VEHICLE 1	BIKE, BUS, MOTORBIKE, PICKUP, TRAIN
VEHICLE 2	LAWN-MOWER, ROCKET, CAR, TANK, TRACTOR

fine label (the class to which it belongs) and a *coarse* label (the superclass to which it belongs). CIFAR-100 is a comparatively larger dataset with respect to the number of classes to be learned for an image classification problem. The aspect that makes this dataset very useful in the study of SBR, is number of annotated images present in each class is comparatively very low to train any deep neural network, therefore, the need to exploit prior knowledge is much higher than in Winston Animal or CIFAR-10 datasets.

Each pattern in CIFAR-100 is a RGB image composed of 32×32 pixels and are natural images like in CIFAR-10. Each image is an one object

image with the object in different poses, angles and magnifications occupying different positions in the image.

5.4.1 Knowledge Domain

The prior knowledge used for this task is divided into two groups: The first group consists of 150 logic rules that correlates the fine classes with the coarse/super classes in CIFAR-100. There are 100 fine predicates and 20 coarse predicates in the taxonomy. All these classes/predicates in both the levels are mutually exclusive and hence, two output softmax classifier layers are used in the CNNs while predicting these classes: one classifying the super classes and the other separating the 100 fine classes. The list of this class hierarchy is tabulated in Table 21. The second group of knowledge predicates includes an extra 5 predicates that expresses some additional semantic knowledge like whether an object occurs indoor or outdoor, whether it should have wheels, etc. The 5 new classes are: *MAMMALS, HOUSEHOLD, OUTDOOR, TRANSPORT, HAS WHEELS*.

Therefore, any CNN network that expresses all these relationships uses three sets of outputs: one with 100 outputs for the fine classes, one having 20 outputs for the super classes and other one that expresses these 5 handcrafted additional classes. A small sample of the knowledge base used in these experiments is reported in Table 22. The full set of rules is given in Table 61 in Section A.0.3.

5.4.2 CNN models

CNN models are integrated in SBR in the same manner as described in the experimental simulations of CIFAR-10 in Section 5.3.2.

Different configurations of deep CNN models are integrated in the Tensorflow implementation of SBR following their architectural properties as in their original works (HZRS15a; HZRS16b; HKK16; ZK16; ZZK$^+$17; DT17).

$32, 110, 164$ layered Resnets, 164 layered Preactivated Resnets, 110, 164, 200, 236 and 272 layered Pyramidal Resnets, $16, 28$ and 40 layered Wide Residual networks with the widening factor of 4, 8 and 10 are used to test the image classification performance on CIFAR-100 using SBR. 28 layered WideResnets (widening factor of 10) are also tested with images regularized through random erasing and cut-out techniques as reported

Table 22: Small sample of the 200 rules used for CIFAR-100 experiments.The rules are divided into two groups: sample of the 150 rules expressing the class taxonomy, and sample of 50 hand-crafted rules to express additional semantic information.

$\forall x$ AQUATIC MAMMALS$(x) \Rightarrow$ BEAVER$(x) \vee$DOLPHIN$(x) \vee$OTTER$(x) \vee$SEAL(x)
$\vee$WHALE(x)
$\forall x$ BEAVER$(x) \Rightarrow$ AQUATIC MAMMALS(x)
$\forall x$ DOLPHIN$(x) \Rightarrow$ AQUATIC MAMMALS(x)
$\forall x$ OTTER$(x) \Rightarrow$ AQUATIC MAMMALS(x)
$\forall x$ SEAL$(x) \Rightarrow$ AQUATIC MAMMALS(x)
$\forall x$ WHALE$(x) \Rightarrow$ AQUATIC MAMMALS(x)

$\forall x$ VEHICLE 1$(x) \Rightarrow$ BIKE$(x) \vee$ BUS$(x) \vee$ MOTORBIKE$(x) \vee$PICKUP_TRUCK$(x) \vee$ TRAIN(x)
$\forall x$ VEHICLES 2$(x) \Rightarrow$ LAWN MOWER$(x) \vee$ ROCKET$(x) \vee$ STREETCAR$(x) \vee$ TANK(x)
$\vee$ TRACTOR(x)
$\forall x$ BIKE$(x) \Rightarrow$ VEHICLE 1(x)
$\forall x$ BUS$(x) \Rightarrow$ VEHICLE 1(x)
$\forall x$ MOTORBIKE$(x) \Rightarrow$ VEHICLE 1(x)
$\forall x$ PICKUP$(x) \Rightarrow$ VEHICLE 1(x)
$\forall x$ TRAIN$(x) \Rightarrow$ VEHICLES 1(x)
$\forall x$ LAWN MOWER$(x) \Rightarrow$ VEHICLES 2(x)
$\forall x$ ROCKET$(x) \Rightarrow$ VEHICLES 2(x)
$\forall x$ STREETCAR$(x) \Rightarrow$ VEHICLES 2(x)
$\forall x$ TANK$(x) \Rightarrow$ VEHICLES 2(x)
$\forall x$ TRACTOR$(x) \Rightarrow$ VEHICLES 2(x)

$\forall x$ MAMMALS $(x) \Rightarrow$ LARGE CARNIVORES$(x) \vee$ OMNIVORES AND HERBIVORES(x)
$\vee$ MEDIUM MAMMALS$(x) \vee$ SMALL MAMMALS$(x) \vee$ AQUATIC MAMMALS(x)
$\forall x$ LARGE CARNIVORES$(x) \Rightarrow$ MAMMALS(x)
$\forall x$ OMNIVORES AND HERBIVORES$(x) \Rightarrow$ MAMMALS(x)
$\forall x$ MEDIUM MAMMALS$(x) \Rightarrow$ MAMMALS(x)
$\forall x$ SMALL MAMMALS$(x) \Rightarrow$ MAMMALS(x)
$\forall x$ AQUATIC MAMMALS$(x) \Rightarrow$ MAMMALS(x)

$\forall x$ HAS WHEELS$(x) \Rightarrow$ BICYCLE$(x) \vee$ BUS$(x) \vee$ MOTORCYCLE$(x) \vee$ PICKUP TRUCK$(x) \vee$
TRAIN$(x) \vee$ LAWN MOWER$(x) \vee$ STREETCAR$(x) \vee$ TANK$(x) \vee$ TRACTOR(x)
$\forall x$ BICYCLE$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ BUS$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ MOTORCYCLE$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ PICKUP TRUCK$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ TRAIN$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ LAWN MOWER$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ STREETCAR$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ TANK$(x) \Rightarrow$ HAS WHEELS(x)
$\forall x$ TRACTOR$(x) \Rightarrow$ HAS WHEELS(x)

on the original paper (ZZK$^+$17; DT17). These techniques are complemented during training with other simple data augmentation like horizontal flips and random crops. During training with these regularization techniques, dropout was not used since it is believed that random erasing and cut outs both are actually contiguous dropouts occurring at the

input level rather than at the neuron level.

All the models are initialized from the same weight initialization and trained for 150 to 300 epochs (depending on the network size) similar to the CIFAR-10 experiments. All the hyper parameter settings and other data augmentation techniques are used consistently as in the original works.

The hyperparameters of collective classification used for this dataset are also similar to the CIFAR-10 experiments. The classification accuracies are always reported on the final 100 classes. In constant weight based SBR, the λ_l values for all predicates is set to 1. Different settings of rule injections are used for the experiments with different t-norms. When variable or predicate dependent λ_l^k are chosen, the classes like BABY, BOY, LIZARD etc. that have low classification accuracies with CNN networks, are traded with low regularizer values between 0 to 0.49 empirically, whereas the classes like APPLE, BOTTLE, CAN etc. have high regularizers between 0.51 to 1.

In order to follow the optimization plans, different empirical analysis are performed, like assigning the non-convex constraints a second priority or giving the non-convex constraints a lower weight (i.e lower λ_h^k) values.

5.4.3 Results

Four different experimental simulations are performed on this image classification task as listed below.

- Non-Transductive simulation with collective classification using different t-norms and different optimization strategies, expressing the hierarchical relationship between the super classes and the fine classes. Semi-supervised setting is artificially created by training on a small percentage of supervised examples to take advantage of logical constraints during joint training.

- Transductive simulation with collective classification using different t-norms and optimization strategies, expressing the hierarchical relationship between 120 super and fine classes.

- Non-Transductive simulation with collective classification using different t-norms and different optimization strategies, using the logical rules of the super and fine classes along with the hand crafted rules for the additional classes. Semi-supervised setting

is also artificially created in this simulation by training on small percentages of supervised examples.

- Transductive simulation with collective classification using different t-norms and optimization strategies, expressing the same relationship between 125 classes: super, fine and the additional classes.

The main objective of these simulations is to study extensively the effects of incrementally adding constraints in the SBR framework. The previous works of SBR didnot use optimization strategies and therefore adding of non convex constraints was not feasible. However, these experiments for the first time empirically show how the number and the nature of the domain rules affect the quality of the trained model. It also demonstrates the benefits of adding constraints at different stages during the learning and understanding how the simple domain knowledge of the class hierarchy (i.e. it is not necessary to design complex rules in order to improve classification accuracies) can be utilized to improve the classification outputs of the deep neural networks. Like the previous simulations, these experiments emphasizes on the differences between transductive and non transductive learning in SBR and on the importance of collective classification as the magnitude of the number of final classes increase 10 times than CIFAR-10.

In these experimental simulations, a combination of different fuzzy logics are used with different proportions of rules during a single training. This setting is known as Combination setting and it is hypothesized to perform better than using an individual fuzzy logic like Weak Lukasiewicz or Product for all the rules during a single training. Sometimes, few rules of the knowledge base is assigned lower priorities compared to the others, and this setting is called as the Half-Weight Combination. The following experimental evaluations make an extensive study about different approaches of using fuzzy logics during training in different CNN networks.

Experimental Simulation I :

Non transductive setup using 45000 supervised training samples drawn randomly from 100 classes, the remaining 5000 examples used for validation and the $10,000$ datapoints for the test set.

Table 23 and Table 24 compares the collective classification error rates in a constant weight based SBR and in a vanilla SBR respectively using

Table 23: Error rate for the 100 final classes for different deep architectures in non transductive mode with collective classification over the network outputs using different selection of t-norms with constant λ_l. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network using different selection of t-norms.

Models	CNN	CC-WL	CC-P	CC-HW
Res-32	29.5	28.2	28.7	28.4
Res-110	25.2	23.5	23.6	23.5
Res-164	25.2	23.8	24.0	23.8
PRes-164	24.3	22.4	22.7	22.4
APN-110,α=84	20.2	19.9	19.9	19.8
APN-110, α=270	18.3	18.0	18.2	18.0
APN-164, α=84	18.3	18.1	18.3	18.1
APN-164, α=270	17.0	16.9	16.9	16.9
APN-236, α=220	16.4	16.2	16.3	16.2
APN-272, α=200	16.4	16.0	16.2	16.0
MPN-110, α=27	18.8	18.5	18.6	18.5
WRes-16, γ=8	22.1	21.8	21.9	21.8
WRes-28, γ=10	20.5	20.3	20.4	20.3
WRes-40, γ=4	22.9	21.5	21.8	21.4
WRes-D-28, γ=10	20.0	19.9	20.0	19.9
WRes-D-RE-28, γ=10	17.7	16.4	16.5	16.4
WRes-D-C-28, γ=10	15.2	**14.9**	15.0	**14.9**

different t-norms with 150 rules created with the heirarchical information of the super and the fine classes. Res, PRes, APN, MPN, WRes, WRes-D, WRes-D-RE and WRes-D-C refers to Resnets, Pre-Activated Resnets, Additive Pyramidal Resnets, Multiplicative Pyramidal Resnets, Wide Resnets, Wide Resnets with dropout, Wide Resnets with dropout and random erasing and Wide Resnets with dropout and cutouts respectively (HZRS15a; HZRS16b; HKK16; ZK16; ZZK$^+$17; DT17). Weak Lukasiewicz, Product t-norms and Half-Weight Combination are used to inject logical constraints into the CNN networks integrated in the SBR. These are abbreviated as CC-WL, CC-P and CC-HW respectively in the results Table 23 and Table 24.

As mentioned earlier different combinations of Lukasiewicz, Product and Minimum t-norms are applied to different groups of rules in the rule

Table 24: Error rate for the 100 final classes for different deep architectures in non transductive mode with collective classification over the network outputs using different selection of t-norms with variable λ_l^k. CNN, CC-WL(λ_l^k), CC-P(λ_l^k) and CC-HW(λ_l^k): convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network using different selection of t-norms.

Models	CNN	CC-WL(λ_l^k)	CC-P(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.5	28.0	28.6	28.0
Res-110	25.2	22.8	22.9	22.8
Res-164	25.2	23.5	23.9	23.7
PRes-164	24.3	22.0	22.3	22.0
APN-110, α=84	20.2	19.8	19.9	19.8
APN-110, α=270	18.3	18.0	18.2	18.0
APN-164, α=84	18.3	18.1	18.2	18.1
APN-164, α=270	17.0	16.7	16.9	16.7
APN-236, α=220	16.4	16.0	16.3	16.1
APN-272, α=200	16.4	16.0	16.2	16.0
MPN-110, α=27	18.8	18.4	18.5	18.5
WRes-16, γ=8	22.1	21.7	21.8	21.7
WRes-28, γ=10	20.5	19.9	20.0	19.9
WRes-40, γ=4	22.9	21.4	21.7	21.4
WRes-D-28, γ=10	20.0	19.8	19.9	19.8
WRes-D-RE-28,γ=10	17.7	16.4	16.5	16.4
WRes-D-C-28, γ=10	15.2	**14.8**	15.0	**14.7**

set. For example, about 75 rules in the knowledge base are converted to constraints using Lukasiewicz t-norm, some 40 rules are converted using Minimum t-norm and the remaining rules use Product t-norms. Thus the group of constraints built from the combination of different t-norms is referred to as Combination (C) setting. Some of the rules in the knowledge base are also assigned lower priorities which means that they will be added in the later stage of learning and these are the ones that have the possibility of becoming non-convex (for example: the constraints created from rules with the $\vee$ operator). Using different t-norms for different groups of rules and then assigning lower priority (priority of 0.5) to the concave rules is referred to as the Half-Weight Combination (HW) setting. Comparison of all the t-norms for constant weight based SBR and vanilla SBR of this simulation are detailed in Table 62 and in Table 63 in

Table 25: Comparison of the collective classification error rate for the 100 final classes using scarce training data in non transductive mode with constant weight based SBR. % Data: different amounts of supervisions used for training each network. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network using different selection of t-norms.

Models	%Data	CNN	CC-WL	CC-P	CC-HW
Res-32	10	43.4	41.6	41.7	41.6
	20	39.3	38.2	38.2	38.2
	50	34.6	33.9	33.9	33.9
PRes-164	10	40.4	40.1	40.2	40.1
	20	37.6	37.5	37.5	37.5
	50	31.3	31.3	31.3	31.3
APN-236,$\alpha = 220$	10	31.9	30.6	30.8	30.7
	20	28.2	27.4	27.4	27.4
	50	20.2	19.6	19.6	19.5
APN-272,$\alpha = 200$	10	30.9	29.8	29.8	29.8
	20	27.6	26.4	26.4	26.4
	50	19.8	18.6	18.6	18.5
WRes-28,$\gamma = 10$	10	35.9	34.8	34.8	34.8
	20	33.6	32.4	32.4	32.4
	50	24.8	23.6	23.6	23.5
WRes-D-28,$\gamma = 10$	10	33.5	32.3	32.3	32.3
	20	31.6	31.4	31.4	31.4
	50	23.8	22.6	22.6	22.6
WRes-D-RE-28,$\gamma = 10$	10	29.3	27.2	27.2	27.2
	20	27.2	26.1	26.2	26.2
	50	19.4	18.2	18.3	18.3
WRes-D-C-28,$\gamma = 10$	10	28.8	26.8	26.8	26.8
	20	26.2	25.5	25.5	25.5
	50	18.8	**17.2**	17.3	**17.1**

Section A.0.3.

Weak Lukasiewicz t-norm and Half-Weight Combination setting are found to be the ones that perform the best with most of the tested CNN architectures in constant and vanilla SBR. A 28-layered wide residual network with dropout and cutouts is one of the best performer, that reduces

the collective classification error rates from 15.2% to 14.9% in a constant SBR and by further 0.2% in vanilla SBR. Weak Lukasiewicz fragment has optimization advantages over other t-norms and Half-Weight Combination helps to use the different t-norms very tactfully with lower weights to the rules that can introduce concavity, therefore facilitating faster training and better generalization capabilities of the CNNs. The Half-Weight Combination setting helps to empirically demonstrate the advantages of optimization heuristics. It supports the theoretical development made in this research work by addition of the optimization plans to the SBR framework thus making it easier to train better classification models. It is also seen that vanilla SBR regulates the predicates based on the classification outputs of the CNNs, and therefore provides performance gains even during collective classification in non transductive learning.

An ablation study is performed by selecting a few deep CNN architectures and using different t-norms with variable percentage of supervised data points. Table 25 and Table 26 shows the effect of logical constraints when 10%, 20% and 50% of supervised examples are used for training with Weak-Lukasiewicz, Product t-norms and Half-Weight combination setting in constant and vanilla SBR.

The collective classification error rate with constraints in constant SBR is reduced by 1.5% from the baseline CNN when only 50% of the supervisions are available during training compared to 0.24% reduction when 100% of supervisions are available. In vanilla SBR, the collective classification error rate is reduced by 2.17% with only 50% supervised data compared to 0.4% reduction when 100% of supervised examples are used. This confirms the advantages of the SBR logical framework in semi-supervised learning also for classification datasets with comparatively large number of classes (100 to 125 classes). For the comparisons with other t-norms in fuzzy logics, a detailed analysis is presented in Table 64 and in Table 65 in Section A.0.3. All these results for non transductive learning aligns with the conclusions drawn for the multi-label image classification on the Winston Animal dataset and CIFAR-10.

Experimental Simulation II :

Transductive simulation where training of the CNN models is performed using all $110,000$ patterns out of which only $45,000$ supervisions are used for training and the remaining supervisions are used for validation and test operations.

Table 27 and Table 28 compares the collective classification error rates

Table 26: Comparison of the collective classification error rate for the 100 final classes using scarce training data in non transductive mode with vanilla SBR. % Data refers to the different amounts of supervisions used for training each network. CNN, CC-WL(λ_l^k), CC-P(λ_l^k) and CC-HW(λ_l^k): convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	%Data	CNN	CC-WL(λ_l^k)	CC-P(λ_l^k)	CC-HW(λ_l^k)
Res-32	10	43.4	40.1	40.2	40.1
	20	39.3	37.4	37.4	37.5
	50	34.6	31.0	31.0	30.9
PRes-164	10	40.4	38.1	38.1	38.2
	20	37.6	36.0	36.0	35.9
	50	31.3	29.9	30.0	29.9
APN-236, $\alpha = 220$	10	31.9	29.8	29.8	29.8
	20	28.2	26.4	26.4	26.4
	50	20.2	18.6	18.6	18.5
APN-272, $\alpha = 200$	10	30.9	27.8	27.8	27.8
	20	27.6	23.4	23.4	23.3
	50	19.8	17.6	17.6	17.5
WRes-28, $\gamma = 10$	10	35.9	33.8	33.8	33.8
	20	33.6	31.4	31.4	31.4
	50	24.8	22.2	22.2	22.1
WRes-D-28, $\gamma = 10$	10	33.5	29.7	29.7	29.7
	20	31.6	28.4	28.5	28.4
	50	23.8	21.5	21.5	21.5
WRes-D-RE-28, $\gamma = 10$	10	29.3	26.9	26.9	26.9
	20	27.2	25.4	25.4	25.4
	50	19.4	17.6	17.6	17.6
WRes-D-C-28, $\gamma = 10$	10	28.8	26.9	26.9	26.9
	20	26.2	23.7	23.7	23.7
	50	18.8	16.6	16.7	**16.6**

in a constant weight based SBR and vanilla SBR respectively with the 150 rules that exploits the coarse and fine class relationship as in the previous simulation. The naming conventions in the tables remain the same.

It is observed that Weak Lukasiewicz t-norm performs well for most of the deep CNN architectures even in transductive learning both in constant and vanilla SBR. Using a wise and careful combination of different fuzzy logics and assigning lower weights to the rules that can lead

Table 27: Classification and Collective classification error rate for the 100 final classes test set for different deep architectures using different selection of t-norms with constant λ_l in transductive mode. CNN, WL, CC-WL, HW and CC-HW: convolutional neural network baseline, classification error in transductive learning, collective classification on neural network outputs using Weak Lukasiewicz and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network using the three selection of t-norms.

Models	CNN	WL	CC-WL	HW	CC-HW
Res-32	29.5	26.2	26.0	26.1	26.0
Res-110	25.2	23.4	22.9	23.2	22.9
Res-164	25.2	22.8	22.6	22.8	22.5
PRes-164	24.3	21.8	21.4	21.6	21.4
APN-110,α=84	20.2	19.1	18.8	19.0	18.9
APN-110, α=270	18.3	17.0	16.7	17.0	16.7
APN-164, α=84	18.3	16.3	16.0	16.2	16.0
APN-164, α=270	17.0	16.1	15.8	16.0	15.8
APN-236, α=220	16.4	15.2	15.0	15.2	14.9
APN-272, α=200	16.4	14.7	14.5	14.6	14.5
MPN-110, α=27	18.8	17.1	16.8	17.0	16.8
WRes-16, γ=8	22.1	21.0	20.9	20.9	20.9
WRes-28, γ=10	20.5	19.2	19.0	19.2	19.0
WRes-40, γ=4	22.9	20.6	20.4	20.6	20.4
WRes-D-28, γ=10	20.0	19.2	19.0	19.1	18.9
WRes-D-RE-28, γ=10	17.7	15.2	15.1	15.2	15.1
WRes-D-C-28, γ=10	15.2	13.9	**13.6**	13.8	**13.6**

to concavity, helps in building an optimal set of constraints. In transductive learning mode, on CIFAR-100, the wide residual network with 28 layers having dropout at the neuron level and cut outs at the input level achieves the highest performance gain. It reduces the classification error rates from 15.2% to 13.8% in a constant SBR and by further 0.7% in vanilla SBR. Collective classification further pushes down the error rate by $0.3 - 0.4\%$. All the factors that contribute to this significant performance gain of 2.5% are : using a combination of t-norms and optimization heuristics (i.e introducing convex constraints at a earlier stage of learning compared to the non convex constraints) during joint training, using predicate dependent regularizers on CNN outputs and using collective classification at the test time. The flexibility of using differ-

Table 28: Classification and Collective classification error rate for the 100 final classes on CIFAR-100 test set for different deep architectures using different selection of t-norms with variable λ_l^k in transductive mode. CNN, WL(λ_l^k), CC-WL(λ_l^k), HW(λ_l^k) and CC-HW(λ_l^k): convolutional neural network baseline, transductive classification error rates and collective classification on neural network outputs using Weak Lukasiewicz and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network using different selection of t-norms.

Models	CNN	WL(λ_l^k)	CC-WL(λ_l^k)	HW(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.5	25.9	25.1	25.9	25.1
Res-110	25.2	21.9	21.7	21.8	21.7
Res-164	25.2	21.6	21.0	21.6	21.0
PRes-164	24.3	20.7	20.0	20.7	20.0
APN-110, α=84	20.2	18.1	17.8	18.1	17.8
APN-110, α=270	18.3	16.5	16.0	16.5	16.0
APN-164, α=84	18.3	15.9	15.1	15.9	15.1
APN-164, α=270	17.0	15.1	14.8	15.1	14.7
APN-236, α=220	16.4	13.8	13.0	13.8	13.0
APN-272, α=200	16.4	13.1	12.9	13.1	12.9
MPN-110, α=27	18.8	15.9	15.4	15.9	15.4
WRes-16, γ=8	22.1	20.9	20.8	20.9	20.7
WRes-28, γ=10	20.5	18.1	17.9	18.1	17.9
WRes-40, γ=4	22.9	19.8	19.4	19.8	19.4
WRes-D-28, γ=10	20.0	19.0	18.8	19.1	18.9
WRes-D-RE-28, γ=10	17.7	15.1	15.0	15.1	15.0
WRes-D-C-28, γ=10	15.2	13.2	**12.8**	13.1	**12.7**

ent strategies to improve the classification results in the SBR framework, makes it not only a powerful architecture but it also becomes an unified tool that can be used in variety of real world applications targeted to solve different kinds of problems. Detailed comparisons of all these scenarios with all other t-norms are given in Table 66, Table 67, Table 68 and Table 69 in Section A.0.3.

The improvement in transductive learning is 2.5% using the same rule set as compared to only 0.5% improvement over the baselines in non transductive learning. During transductive learning, the validation and the test patterns are also used during training without supervisions,

therefore, creating a semi-supervised learning environment in which the rules have a greater effect that further reduces the error rates in classification. These benefits can be used in real life applications like searching for matching images by photographers from a huge database of noisy images. For example :the photographers can use a model trained (training is done in transductive mode with few supervised examples and millions of unsupervised datapoints) to classify 100 categories and search for a matching category in a few seconds.

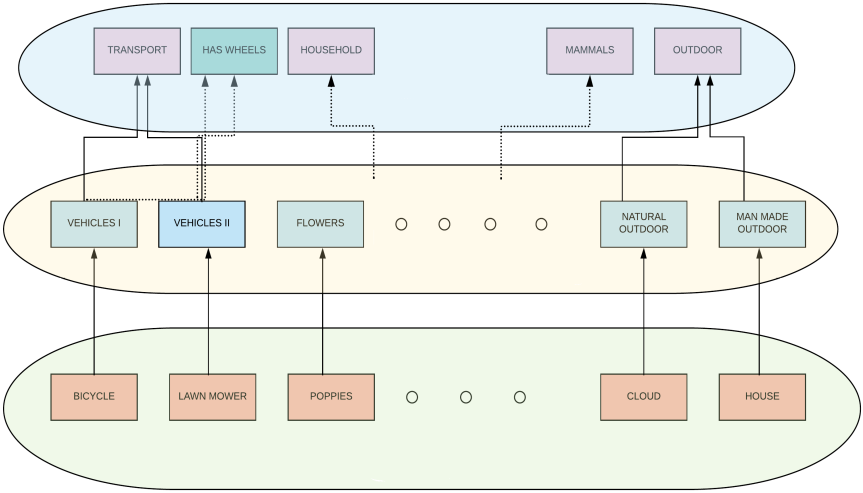


Figure 23: Three hierarchical levels including the fine, coarse classes and the new 5 additional classes.

Experimental Simulation III :

This shows a non transductive setup with 5 additional predicates. However, the number of supervised examples and the split of the training, validation and test set remains exactly the same as in the experimental simulation I of CIFAR-100. With the addition of new predicates, the CNNs are trained with additional classification layers. Hence instead of the two class hierarchy that was present previously with the coarse and fine classes, now another additional upper hierarchical level gets added. This taxonomical structure is represented in the Figure 23.

Table 29 and Table 30 compares the collective classification error rates in a constant weight based SBR and in a vanilla SBR respectively using different t-norms with all the available rules created using 125 predicates. A new set of 50 rules is created with the addition of the 5 new predicates. All the naming conventions in the results table remain the same as used in the previous experiments and the accuracies are always calculated over the final 100 classes. Comparison with all the t-norms are detailed in Table 70 and in Table 71 in Section A.0.3.

Half-Weight Combination setting (where combination of different fuzzy logics are used for different set of rules and also priorities are assigned to rules following some heuristics), performs very well in most of the tested CNN architectures in constant and vanilla SBR. With the addition of new rules, the number of non-convex constraints increases and hence, Half-Weight Combination setting which optimally inserts the non convex constraints into the learning, outperforms even the Weak-Lukasiewicz t-norm which by itself has several optimization advantages.

The 28-layered wide residual network with dropout and cut outs is now seen to have a collective classification error rate of 14.1% in a constant SBR and 14.0% in vanilla SBR with Half-Weight Combination setting which is 1.2% lower than the deep neural network baseline model trained without rules and about 0.7% lower than with a smaller subset of rules. Thus incremental addition of more rules has shown its benefits and this gain in performance is achieved. Therefore, taking a baby step at a time, like adding more rules during joint learning, using predicate based regularizers, applying collective classification have all cumulatively helped in achieving a accuracy gain with a promising margin. These performance gain results empirically reconfirm that each of these additions make SBR a complete unified approach for a semi-supervised logic integrated deep learning based framework which is much more advanced than its previous version in the earlier works (DGS16).

Experimental Simulation IV :

Transductive simulation with 5 additional predicates/classes where all $110,000$ data patterns are used for training of the CNN models, out of which only $45,000$ patterns are supervised during training and the remaining supervisions are used for validation and test operations. The rule set is exactly the same as the rules used in the experimental simulation III.

Table 31 and Table 32 compares the collective classification error rates

Table 29: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures over 125 network outputs using different selection of t-norms with constant λ_l in non transductive mode. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	CC-WL	CC-P	CC-HW
Res-32	29.5	27.5	28.2	27.4
Res-110	25.2	22.8	23.0	22.8
Res-164	25.2	22.8	23.9	22.8
PRes-164	24.3	21.7	21.8	22.6
APN-110,α=84	20.2	19.0	19.1	19.1
APN-110, α=270	18.3	17.2	17.5	17.2
APN-164, α=84	18.3	17.3	17.7	17.3
APN-164, α=270	17.0	16.1	16.2	16.0
APN-236, α=220	16.4	15.4	15.5	15.4
APN-272, α=200	16.4	15.4	15.4	15.3
MPN-110, α=27	18.8	17.7	17.8	17.7
WRes-16, γ=8	22.1	21.0	21.0	21.0
WRes-28, γ=10	20.5	19.6	19.8	19.7
WRes-40, γ=4	22.9	20.7	21.0	20.7
WRes-D-28, γ=10	20.0	19.2	19.2	19.1
WRes-D-RE-28, γ=10	17.7	15.7	15.7	15.6
WRes-D-C-28, γ=10	15.2	14.2	14.3	**14.1**

in a constant weight based SBR and vanilla SBR respectively in this transductive set up. Detailed analysis on all the t-norms is found in Table 72, Table 73, Table 74 and Table 75 in appendix A.

With the addition of the new predicates, resulting into more constraints, and using optimization heuristics in the Half-Weight Combination setting is seen to be the best performer. During collective classification, the wide residual network with 28 layers, in a constant SBR lowers the error rate by 0.6% from its counterpart that uses only hierarchical information of 120 predicates and by 1.2% from its baseline architecture which is not trained with any logical constraints. However in a vanilla SBR, the reduction is observed to be 0.7% and 3.2% respectively. Transductive mode already provides a semi-supervised environment for learning, therefore the benefits of rules are quite predominant in this type

Table 30: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures over 125 network outputs using different selection of t-norms with variable λ_l^k in non transductive mode. CNN, CC-WL(λ_l^k), CC-P(λ_l^k) and CC-HW(λ_l^k): convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	CC-WL(λ_l^k)	CC-P(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.5	27.2	28.0	27.0
Res-110	25.2	22.7	22.9	22.6
Res-164	25.2	22.6	23.0	22.6
PRes-164	24.3	21.2	21.6	21.2
APN-110, α=84	20.2	19.0	19.0	19.0
APN-110, α=270	18.3	17.1	17.2	17.1
APN-164, α=84	18.3	17.2	17.3	17.2
APN-164, α=270	17.0	16.0	16.0	15.9
APN-236, α=220	16.4	15.2	15.5	15.2
APN-272, α=200	16.4	15.1	15.9	15.0
MPN-110, α=27	18.8	17.4	17.6	17.3
WRes-16, γ=8	22.1	20.9	21.0	20.9
WRes-28, γ=10	20.5	19.2	19.4	19.2
WRes-40, γ=4	22.9	20.7	20.8	20.5
WRes-D-28, γ=10	20.0	19.0	19.1	19.0
WRes-D-RE-28,γ=10	17.7	15.5	15.7	15.5
WRes-D-C-28, γ=10	15.2	14.0	14.1	**14.0**

of learning.

5.4.4 Discussion

In the image classification task on CIFAR-100, the CNN models like Resnets, Pyramidal Resnets and Wide Resnets used in combination with different regularization techniques at the input level when integrated in SBR and learned jointly with rules, the classification error rate is reduced by approximately $3\% - 4\%$ in small networks to 5% in large networks. As demonstrated in CIFAR-10 experiments, when the size of the network increases, the importance of adding rules also increases because the rules help in overcoming the problem faced due to overfitting. Con-

Table 31: Classification and collective classification error rate for the 100 final classes with 125 outputs from different deep architectures using different selection of t-norms with constant λ_l in transductive mode. CNN, WL, CC-WL, HW and CC-HW: convolutional neural network baseline, transductive classification outputs with rules, collective classification on neural network outputs using Weak Lukasiewicz and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	WL	CC-WL	HW	CC-HW
Res-32	29.5	25.8	25.4	25.7	25.2
Res-110	25.2	22.7	22.1	22.6	22.0
Res-164	25.2	22.3	22.1	22.0	21.9
PRes-164	24.3	21.5	21.0	21.4	21.0
APN-110,α=84	20.2	18.9	18.6	18.8	18.6
APN-110, α=270	18.3	16.5	16.1	16.4	16.1
APN-164, α=84	18.3	16.0	15.6	16.0	15.6
APN-164, α=270	17.0	15.8	15.4	15.7	15.4
APN-236, α=220	16.4	14.9	14.7	14.8	14.7
APN-272, α=200	16.4	14.4	14.1	14.3	14.1
MPN-110, α=27	18.8	16.7	16.3	16.6	16.2
WRes-16, γ=8	22.1	20.8	20.3	20.7	20.3
WRes-28, γ=10	20.5	19.0	18.7	18.9	18.8
WRes-40, γ=4	22.9	20.4	20.1	20.2	20.1
WRes-D-28, γ=10	20.0	18.6	18.1	18.5	18.1
WRes-D-RE-28, γ=10	17.7	15.0	14.9	14.9	14.9
WRes-D-C-28, γ=10	15.2	13.5	13.1	13.4	**13.0**

sidering less than 500 supervisions are available for each class (few of them are randomly selected and used in the validation process), during training, the improvements happen only by the application of 200 logic rules and some optimization heuristics with negligible computational overhead. Apart from all the benefits of SBR observed in CIFAR-10, the additional perspective obtained through the experimental evaluations in CIFAR-100 is that the increase in the number of logic rules is beneficial and results in creating better classification models. Therefore, this leads to the finding that addition of more domain knowledge can better suffice for the scarcity of supervisions.

Constructing the rules is also easy as they can be directly obtained from the hierarchical information available in the WordNet taxonomy.

Table 32: Classification and collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with variable λ_l^k in transductive mode. CNN, WL, CC-WL, HW and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	WL(λ_l^k)	CC-WL(λ_l^k)	HW(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.5	25.2	24.9	25.3	24.8
Res-110	25.2	21.6	21.2	21.7	21.2
Res-164	25.2	21.0	20.6	21.0	20.5
PRes-164	24.3	20.3	19.9	20.3	19.8
APN-110, α=84	20.2	17.9	17.2	17.9	17.1
APN-110, α=270	18.3	15.8	15.4	15.9	15.4
APN-164, α=84	18.3	15.4	14.8	15.4	14.8
APN-164, α=270	17.0	15.0	14.3	15.1	14.2
APN-236, α=220	16.4	13.3	12.7	13.4	12.6
APN-272, α=200	16.4	13.1	12.1	13.1	12.0
MPN-110, α=27	18.8	15.6	14.4	15.5	14.4
WRes-16, γ=8	22.1	20.6	20.4	20.6	20.3
WRes-28, γ=10	20.5	17.8	17.1	17.9	17.1
WRes-40, γ=4	22.9	19.8	19.1	19.9	19.1
WRes-D-28, γ=10	20.0	18.5	18.3	18.5	18.2
WRes-D-RE-28, γ=10	17.7	14.9	14.8	14.9	14.8
WRes-D-C-28, γ=10	15.2	12.9	12.1	13.0	**12.0**

The other conclusions obtained through the simulations on this dataset remain the same as the previous ones: Using of techniques like predicate dependent regularizers and collective classification, helps to further improve the rule based learning strategies on CIFAR-100, the transductive simulations and the ablation studies with scarcity of supervisions, demonstrate SBR as a powerful framework for semi-supervised learning. All these techniques are used for the first time in the deep learning community with a rich variety of deep neural networks on CIFAR-100 and this makes an important addition in the field of deep learning research as a whole. Also, the experimental evaluations illustrates a simple but novel methodology of using logical rules by combining different types of

Table 33: Comparison of the Train, Prediction and Collective classification time with different deep CNN networks in transductive and non transductive learning with two levels of hierarchy. NN-T, NN-P and NN-CC: Train time, Prediction time and Collective classification time respectively for the baseline neural network. NNT-T, NNT-P and NNT-CC: Train time, Prediction time and Collective classification time of the respective neural network when in transductive learning mode in SBR. All the times are in hours.

Models	NN-T	NN-P	NN-CC	NNT-T	NNT-P	NNT-CC
Res-32	8.50	0.34	0.35	9.15	0.35	0.35
Res-110	32.50	1.50	1.54	32.83	1.83	1.87
Res-164	33.75	2.14	2.20	34.10	2.22	2.24
PRes-164	33.70	2.12	2.18	34.0	2.20	2.24
APN-110, α=84	49.59	3.25	3.28	50.50	3.30	3.36
APN-110, α=270	54.83	4.35	4.40	55.13	4.40	4.46
APN-164, α=84	53.83	3.27	3.30	54.13	3.32	3.38
APN-164, α=270	56.13	4.40	4.45	56.83	4.45	4.50
APN-236, α=220	65.83	5.45	5.52	66.13	5.54	5.58
APN-272, α=200	67.43	5.62	5.70	68.20	5.74	5.83
MPN-110, α=27	55.23	4.55	4.60	56.43	4.58	4.70
WRes-16, γ=8	33.50	1.83	2.14	33.83	2.13	2.87
WRes-28, γ=10	35.50	2.13	2.85	35.83	2.83	3.17
WRes-40, γ=4	36.50	2.83	3.15	36.83	3.50	3.47
WRes-D-28, γ=10	35.0	2.0	2.15	35.13	2.13	2.50
WRes-D-RE-28, γ=10	35.2	2.0	2.18	35.23	2.13	2.50
WRes-D-C-28, γ=10	35.13	2.0	2.16	35.20	2.53	2.50

fuzzy logics during training. The use of Combination and Half-Weight Combination setting has proved to be beneficial with almost all different deep CNN architectures experimented in the above simulations. This finding paves new pathways of research using SBR for the future works. It would be interesting to study which combination of logical rules help in image classification in which way, what would be best values for the weights assigned to the rules (i.e. how should the rules be penalized during the learning process), how can the weights (λ_h as described in Equation 3.1) of these rules be learned in more principled way.

As described for the Winston Animal and CIFAR-10 datasets, it is important to note that the performance gain using rules and through collec-

Table 34: Comparison of the Train, Prediction and Collective classification time with different deep CNN networks in transductive and non transductive learning with three levels of hierarchy. NN-T, NN-P and NN-CC: Train time, Prediction time and Collective classification time respectively for the baseline neural network. NNT-T, NNT-P and NNT-CC: Train time, Prediction time and Collective classification time of the respective neural network when in transductive learning mode in SBR. All the times are in hours.

Models	NN-T	NN-P	NN-CC	NNT-T	NNT-P	NNT-CC
Res-32	8.56	0.36	0.38	9.54	0.38	0.40
Res-110	32.54	1.60	1.64	33.10	2.10	2.87
Res-164	33.83	2.24	2.30	34.50	2.32	2.54
PRes-164	33.80	2.20	2.28	34.23	2.50	2.54
APN-110, α=84	50.59	3.65	3.68	51.50	3.80	3.88
APN-110, α=270	55.11	4.65	4.70	55.56	4.70	4.76
APN-164, α=84	54.50	3.67	3.70	54.83	3.72	3.78
APN-164, α=270	56.43	4.80	4.85	57.10	4.85	4.90
APN-236, α=220	65.92	5.60	5.62	66.43	6.14	6.18
APN-272, α=200	67.82	6.12	6.40	68.40	6.50	6.83
MPN-110, α=27	56.0	4.61	4.64	57.43	5.08	5.17
WRes-16, γ=8	33.80	2.14	2.32	34.13	2.53	2.87
WRes-28, γ=10	35.60	2.21	2.91	36.10	3.12	3.18
WRes-40, γ=4	36.80	2.87	3.25	36.85	3.56	3.67
WRes-D-28, γ=10	35.23	2.12	2.18	35.41	2.20	2.52
WRes-D-RE-28, γ=10	35.33	2.10	2.28	35.42	2.15	2.52
WRes-D-C-28, γ=10	35.25	2.11	2.26	35.50	2.51	2.53

tive classification in SBR is obtained with negligible overhead of time in both transductive and non transductive learning. Table 33 and Table 34 describes the time taken to train the different deep CNNs on this dataset with two and three levels of hierarchy respectively. With three levels of hierarchy, the training and the inference time increases by a small margin with the same networks when trained with two levels of hierarchy. The addition of rules and collective classification benefits become more predominant as the size of the dataset and the number of categories increase to 100. Here, training of the baseline network from scratch takes approximately $10 - 70$ hours but the collective classification takes only $0.5 - 6$ hours depending on the depth of the neural network. Therefore,

using rules in SBR with lesser amount of supervised examples and using collective classification during the inference time, approximately $50-100$ hours of training time can be saved which would otherwise be required if more data patterns are used during training. Thus, the efficiency in terms of accuracy and time makes the deep learning based SBR framework versatile for different types of practical applications.

5.5 Experimental Analysis on ImageNet Dataset

This image classification task using logical constraints is performed on the most popular image dataset known as ImageNet. ImageNet is an image database organized according to the WordNet hierarchy composed of 1000 classes (Kri09)[8] used for ILSVRC challenges since 2010. For this task, ILSVRC 2012 challenge dataset is used. This challenge has approximately 1.3 million training images, 50000 validation images and 100K test images.

Each of the images are mostly high resolution single object RGB image. The input images used by different CNN architectures are resized and cropped to 224×224 pixels. The images in this database are not only arranged in 1000 categories but they also possess a number of non overlapping nodes and internodes. This indicates that each object category also has several levels of hierarchy in the word sysnets. For example: the category DOG have intertwining super classes like HUNTING DOG$\rightarrow$ CARNIVORE $\rightarrow$ MAMMAL $\rightarrow$ ANIMAL. These semantic and hierarchical relationships of the ImageNet categories are explored to enforce the logical constraints and establish semantic consistency among the inputs.

5.5.1 Knowledge Domain

The prior knowledge base used for this image classification task is created using all the sysnet hierarchical information present in the WordNet hierarchy of this database. The knowledge base correlates the 1000 output predicates with different predicates that are additionally added through hand crafted rules directly derived from this taxonomical architecture of the sysnets. Due to limitation of the resources, some of the sysnet information is simplified for this research work. A subset of the rules present in the knowledge base is shown in Table 35.

[8] `https://http://image-net.org/challenges/LSVRC/`

Table 35: Subset of ImageNet rules handcrafted to exploit the sysnet information.

$\forall x$ PERSON$(x) \vee$ FOOD$(x) \vee$ ARTEFACT$(x) \vee$ ANIMAL$(x) \vee$ PLANT(x)

$\forall x$ ANIMAL$(x) \Rightarrow$ AMPHIBIAN$(x) \vee$ REPTILE $(x) \vee$ FISH $(x) \vee$ BIRD $(x) \vee$ MAMMAL $(x) \vee$ INVERTEBRATE(x)

$\forall x$ AMPHIBIAN$(x) \Rightarrow$ ANIMAL(x)
$\forall x$ REPTILE$(x) \Rightarrow$ ANIMAL(x)
$\forall x$ AMPHIBIAN$(x) \Rightarrow$ BULLFROG$(x) \vee$ TREE_FROG$(x) \vee$ TAILED_FROG$(x) \vee$ EUROPEAN_FIRE_SALAMANDER(x)
$\vee$COMMON_NEWT$(x) \vee$ EFT $(x) \vee$ SPOTTED_SALAMANDER $(x) \vee$ AXOLOTL(x)
$\forall x$ BULLFROG $(x) \Rightarrow$ AMPHIBIAN (x)
$\forall x$ TREE_FROG$(x) \Rightarrow$ AMPHIBIAN(x)
$\forall x$ TAILED_FROG$(x) \Rightarrow$AMPHIBIAN(x)
$\forall x$ EUROPEAN_FIRE_SALAMANDER $(x) \Rightarrow$ AMPHIBIAN (x)
$\forall x$ COMMON_NEWT $(x) \Rightarrow$ AMPHIBIAN(x)
$\forall x$ EFT $(x) \Rightarrow$ AMPHIBIAN(x)
$\forall x$ SPOTTED_SALAMANDER $(x) \Rightarrow$ AMPHIBIAN(x)
$\forall x$ AXOLOTL $(x) \Rightarrow$ SALAMANDER(x)

$\forall x$ REPTILE $(x) \Rightarrow$ SNAKE$(x) \vee$ CROCODILIAN $(x) \vee$ DINOSAUR $(x) \vee$ SAURIAN $(x) \vee$ TURTLE(x)
$\forall x$ SNAKE $(x) \Rightarrow$ REPTILE(x)
$\forall x$ CROCODILIAN $(x) \Rightarrow$ REPTILE (x)
$\forall x$ DINOSAUR $(x) \Rightarrow$ REPTILE (x)
$\forall x$ SAURIAN $(x) \Rightarrow$ REPTILE(x)
$\forall x$ TURTLE $(x) \Rightarrow$ REPTILE(x)
$\forall x$ SNAKE $(x) \Rightarrow$ VIPER$(x) \vee$ SEA_SNAKE $(x) \vee \forall x$ ELAPID$(x) \vee$ CONSTRICTOR $(x) \vee$ COLUBRID_SNAKE (x)
$\forall x$ VIPER $(x) \Rightarrow$ SNAKE (x)
$\forall x$ SEA_SNAKE $(x) \Rightarrow$ SNAKE(x)
$\forall x$ ELAPID $(x) \Rightarrow$ SNAKE (x)
$\forall x$ CONSTRICTOR $(x) \Rightarrow$ SNAKE(x)
$\forall x$ COLUBRID_SNAKE $(x) \Rightarrow$ SNAKE(x)
$\forall x$ VIPER $(x) \Rightarrow$ HORNED_VIPER $(x) \vee$ RATTLE_SNAKE(x)
$\forall x$ HORNED_VIPER $(x) \Rightarrow$ VIPER(x)
$\forall x$ RATTLE_SNAKE $(x) \Rightarrow$ VIPER(x)
$\forall x$ RATTLE_SNAKE $(x) \Rightarrow$ DIAMONDBACK $(x) \vee$ SIDEWINDER(x)
$\forall x$ DIAMONDBACK $(x) \Rightarrow$ RATTLE_SNAKE(x)
$\forall x$ SIDEWINDER $(x) \Rightarrow$ RATTLE_SNAKE(x)

$\forall x$ PERSON$(x) \Rightarrow$ PLAYER$(x) \vee$ RELATIVE(x)
$\forall x$ PLAYER$(x) \Rightarrow$ PERSON(x)
$\forall x$ RELATIVE$(x) \Rightarrow$ PERSON(x)
$\forall x$ PLAYER$(x) \Rightarrow$ BALL_PLAYER$(x) \vee$ SCUBA_DIVER(x)
$\forall x$ RELATIVE$(x) \Rightarrow$ GROOM(x)

$\forall x$ FOOD$(x) \Rightarrow$ ITEM_LIST$(x) \vee$ SOLID$(x) \vee$ SEMI_SOLID$(x) \vee$ LIQUID(x)
$\forall x$ ITEMLIST$(x) \Rightarrow$ FOOD(x)
$\forall x$ SOLID$(x) \Rightarrow$ FOOD(x)
$\forall x$ SEMI_SOLID$(x) \Rightarrow$ FOOD(x)
$\forall x$ LIQUID$(x) \Rightarrow$ FOOD(x)
$\forall x$ ITEMLIST$(x) \Rightarrow$ MENU(x)
$\forall x$ MENU$(x) \Rightarrow$ ITEMLIST(x)
$\forall x$ SOLID$(x) \Rightarrow$ ICE_CREAM$(x) \vee$ ICE_LOLLY$(x) \vee$MEAT$(x) \vee$ BAKERY(x)
$\forall x$ MEAT$(x) \Rightarrow$ HOTDOG$(x) \vee$ POTPIE
$\forall x$ HOTDOG$(x) \Rightarrow$ MEAT
$\forall x$ POTPIE$(x) \Rightarrow$ MEAT
$\forall x$ BAKERY$(x) \Rightarrow$ PIZZA$(x) \vee$ MEAT_LOAF$(x) \vee$ FRENCH_LOAF$(x) \vee$ PRETZEL$(x) \vee$ DOUGH$(x) \vee$ POTPIE (x)
$\vee$ BURRITO $(x) \vee$ CHEESE_BURGER $(x) \vee$ BAGEL $(x) \vee$ CAKE
$\forall x$ PIZZA $(x) \Rightarrow$ BAKERY(x)
$\forall x$ MEAT_LOAF $(x) \Rightarrow$ BAKERY(x)
$\forall x$ FRENCH_LOAF $(x) \Rightarrow$ BAKERY(x)
$\forall x$ PRETZEL $(x) \Rightarrow$ BAKERY(x)
$\forall x$ DOUGH $(x) \Rightarrow$ BAKERY(x)
$\forall x$ POTPIE $(x) \Rightarrow$ BAKERY(x)
$\forall x$ BURRITO $(x) \Rightarrow$ BAKERY(x)
$\forall x$ CHEESE_BURGER $(x) \Rightarrow$ BAKERY(x)
$\forall x$ BAGEL $(x) \Rightarrow$ BAKERY(x)
$\forall x$ CAKE $(x) \Rightarrow$ BAKERY(x)
$\forall x$ CAKE $(x) \Rightarrow$ TRIFLE(x)
$\forall x$ TRIFLE $(x) \Rightarrow$ CAKE(x)
$\forall x$ SEMI_SOLID $(x) \Rightarrow$ MASHED_POTATO$(x) \vee$ GUACAMOLE $(x) \vee$ SAUCE(x)
$\forall x$ SAUCE $(x) \Rightarrow$ CARBONARA$(x) \vee$ CHOCOLATE_SAUCE(x)
$\forall x$ MASHED_POTATO $(x) \Rightarrow$ SEMI_SOLID(x)
$\forall x$ GUACAMOLE $(x) \Rightarrow$ SEMI_SOLID(x)
$\forall x$ SAUCE $(x) \Rightarrow$ SEMI_SOLID(x)
$\forall x$ CARBONARA $(x) \Rightarrow$ SAUCE (x)
$\forall x$ CHOCOLATE_SAUCE $(x) \Rightarrow$ SAUCE(x)
$\forall x$ LIQUID $(x) \Rightarrow$ CONSOMME $(x) \vee$ EGGNOG $(x) \vee$ WINE(x)
$\forall x$ CONSOMME $(x) \Rightarrow$ LIQUID(x)
$\forall x$ EGGNOG $(x) \Rightarrow$ LIQUID(x)
$\forall x$ WINE $(x) \Rightarrow$ LIQUID(x)
$\forall x$ WINE $(x) \Rightarrow$ RED_WINE(x)

5.5.2 CNN Models

Each hierarchical level is represented through an additional classifier layer while training any deep CNN architecture. These classifier layers either have a softmax activation if the considered classes in that layer are mutually exclusive or a sigmoid activation if the considered classes are interdependent. Therefore, architecturally the deep CNNs integrated in the SBR for ImageNet resembles the CNNs used in CIFAR-10 and CIFAR-100 except with minor differences in the number of output layers, training parameters etc. ImageNet is one of the largest annotated image dataset available to the deep learning research community with also large amount of sysnet information associated with it.

The integration of the baseline CNN models with SBR is also similar to the previous experimental simulations in Section 5.2 of this chapter. As ImageNet is a very popular image classification dataset, there are several pretrained deep CNN models publicly available. Pretrained Residual networks, Pre-activated Residual networks and Additive Pyramidal Residual networks are retrained for several iterations with additional classifier layers and logical constraints to obtain the trained models with additional classes. Collective classification is then performed on the test set using these models.

Resnets with 50, 101 and 152 layers, Pre-activated Resnets with 152 and 200 layers and Additive Pyramid Resnets with 200 layers are used to test the image classification performance on ImageNet using SBR. All these networks are trained with images resized along its shorter side and randomly sampled from the range $[256, 480]$ for scale augmentation (HZRS15a; HZRS16b; HKK16). A 224×224 crop is then randomly sampled from an image or its horizontal flip, with per-pixel mean separated as in the original works (HZRS15a).

Batch normalization is used after each convolution layer in these deep networks. During fine tuning these CNN networks with additional classifier layers, a mini-batch size of 64 images are used. The retraining is continued for 50 epochs with an initial learning rate of 0.001, a weight decay of 0.0001 and a momentum of 0.9. The hyperparameters of collective classification are identical to the ones used for the CIFAR experiments.

5.5.3 Results

A large dataset like ImageNet has never been tested before using domain knowledge integrated frameworks. Therefore, the empirical results

is a strength of this research work and a great addition to the research community. Thorough experimental evaluations are conducted to exploit several levels of sysnet hierarchy and to take the advantage of these hierarchical relationships. Experimental simulations on this dataset are shown only for the non transductive setting because of the limitation of resources. In non transductive learning, the patterns are fully supervised during training and hence the rules constructed from hierarchical information are implicitly satisfied during training. However, the advantages provided by the rules can be gained by using the rules for collective classification during the test time. The other scenario where rules can be beneficial in non transductive learning is when the labels are poor and noisy. The addition of logical rules into the deep learners in such cases help in designing robust noise-free easily generalizable classification models. Although, this aspect is not focused in this work but the same SBR framework can be used to deal with such situation without any need of theoretical modifications.

There are three experimental simulations implemented using this dataset, exploiting different levels of hierarchical information. The main goal of these experimental simulations is to emulate the scenario of incremental addition of constraints where there are large number of classes so as to demonstrate the benefit of increasing the number of constraints on very large datasets, the benefit of using optimization heuristics for concave constraints and the improvement that can be obtained by using only collective classification on the baseline network predictions during inference. The experiments on ImageNet using the SBR framework is performed to reconfirm all the findings from Winston Animal, CIFAR-10 and CIFAR-100 datasets on a larger scale. This is just to prove that all the advantages and capabilities of the SBR framework discussed earlier is scalable and usable for real world scenarios in the era of Big data.

Experimental Simulation I :

This experimental simulation is performed with 1.3 million supervised training examples belonging to 1000 classes. In this simulation only three hierarchical levels have been considered. It includes 5 super classes, 21 intermediate classes and 1000 leaf classes. During this simulation, all the classes present in each hierarchical level are mutually exclusive and therefore, softmax activation is used in each of the output layers. The 5 super classes or the top hierarchical level classes are listed as:

Table 36: Second hierarchical level of ImageNet used in the Experimental Simulation I.

TOP HIERARCHICAL LEVEL	SECOND HIERARCHICAL LEVEL
PERSON	PLAYER, RELATIVE
FOOD	ITEMLIST, SEMI SOLID, SOLID, LIQUID
ANIMAL	AMPHIBIAN, REPTILE, FISH, BIRD, MAMMAL, INVERTEBRATE
ARTEFACT	DECORATION, INSTRUMENTATION, STRUCTURE, COVERING, HAY, FABRIC
PLANT	FLOWER, FRUIT, FUNGUS

- PERSON
- FOOD
- ANIMAL
- ARTEFACT
- PLANT

Each of the super classes are divided into intermediate hierarchical level described in the Table 36. And then the second hierarchical level is further divided into 1000 leaf classes.

Table 37 and Table 38 compares the collective classification error rates in a constant weight based SBR and in a vanilla SBR respectively using different t-norms with all the taxonomical rules created with 1026 predicates. The naming conventions used for the t-norms are the same as in CIFAR-10 and CIFAR-100 simulations.

Weak Lukasiewicz, Product t-norms and Half-Weight Combination setting are compared for different CNN architectures. For the remaining types of t-norms, comparison studies are given in Table 76 and in Table 77 in Section A.0.4.

Table 37: Classification error rate for 1000 classes for different deep architectures in a constant weight based SBR using 26 intermediate classes. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting.The bold numbers in the table refer to the best performing network with three selections of t-norms.

Models	CNN	CC-WL	CC-P	CC-HW
Res-50	24.7	24.5	24.6	24.5
Res-101	23.6	23.6	23.6	23.5
Res-152	23.0	22.8	22.9	22.8
PRes-152	22.2	22.0	22.1	22.0
PRes-200	21.9	21.8	21.8	21.8
APN-200 $\alpha=300$	20.5	20.1	20.2	20.2
APN-200 $\alpha=450$	20.1	**20.0**	20.1	**19.9**

Table 38: Classification error rate for 1000 classes on ImageNet for different deep architectures in a vanilla SBR using 26 intermediate classes. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network using the best performing t-norm.

Models	CNN	CC-WL (λ_l^k)	CC-P (λ_l^k)	CC-HW (λ_l^k)
Res-50	24.7	24.4	24.5	24.4
Res-101	23.6	23.5	23.6	23.5
Res-152	23.0	22.8	22.8	22.7
PRes-152	22.2	22.0	22.1	22.0
PRes-200	21.9	21.7	21.8	21.7
APN-200 $\alpha=300$	20.5	20.0	20.2	20.1
APN-200 $\alpha=450$	20.1	**19.9**	20.0	**19.8**

With three levels of hierarchical relationship that have been exploited in this simulation, the collective classification error rate is reduced from 20.1% to 19.9% in Additive Pyramidal Resnets (the deep CNN that gives the lowest error rate) with constant SBR and from 20.1% to 19.8% in vanilla SBR using the Half-Weight Combination setting. In Half-Weight Combination setting, a large set of rules use Weak Lukasiewicz convex fragment and therefore, has optimization advantages and also Half-Weight Combination setting has the advantage of using low weights for non convex constraints at the later stage of learning for faster con-

Table 39: Comparison of the collective classification error rate for 1000 final classes of ImageNet using a subset of data and 26 intermediate predicates in non transductive mode with constant λ_l values. % Data: percentage of supervisions used during training. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network with 50% supervisions using two selections of t-norms.

Models	%Data	CNN	CC-WL	CC-P	CC-HW
Res-50	10	36.1	35.2	35.3	35.2
	20	34.2	33.0	33.1	33.0
	50	30.1	28.6	28.9	28.5
Res-101	10	35.1	34.0	33.9	33.9
	20	33.7	31.9	31.9	31.9
	50	28.5	25.8	25.9	25.8
Res-152	10	34.6	33.1	33.2	33.2
	20	32.6	31.2	31.2	31.2
	50	26.1	26.0	26.0	25.9
PRes-152	10	33.4	32.1	32.2	32.1
	20	31.0	30.1	31.0	30.1
	50	26.1	25.0	25.0	24.9
PRes-200	10	33.2	32.8	32.8	31.8
	20	30.4	29.3	29.3	29.3
	50	25.9	25.1	25.2	25.1
APN-200, $\alpha = 300$	10	32.9	32.8	32.8	32.7
	20	29.6	29.1	29.2	29.1
	50	25.8	24.9	24.9	24.9
APN-200, $\alpha = 450$	10	32.8	31.2	31.2	31.1
	20	29.2	28.0	28.0	28.0
	50	25.0	**24.2**	24.2	**24.1**

vergence. Predicate based regularizers in most CNN architectures help in improving the collective classification on the test set as they effectively regulate the effect of logical constraints based on the individual outputs from the CNN. For this hierarchy, ANIMAL, DINOSAUR, PERSON, HOTDOG etc. classes have high classification outputs and hence they use high regularizer values whereas classes like TRUFLE, DALMATIANS, TABBY CATS have low classification probabilities and hence are regulated with low regularizers chosen through cross validation. There-

fore, the benefits of using predicate regularizers in SBR even for large datasets with numerous output classes is empirically proven through the simulation.

Since in non transductive learning, all the training patterns are supervised during training and because all the patterns are cleanly labeled, the logical rules are implicitly satisfied and hence the small improvement in accuracies using logical constraints is only seen when collective classification is used during inference. However, when a small subset of labels are used during learning, leaving the rest of the training patterns as unsupervised data, the effect of prior knowledge can be demonstrated on a semi-supervised setting. Table 39 and Table 40 shows the collective classification error rates when using variable percentage of supervisions in constant and vanilla SBR respectively.

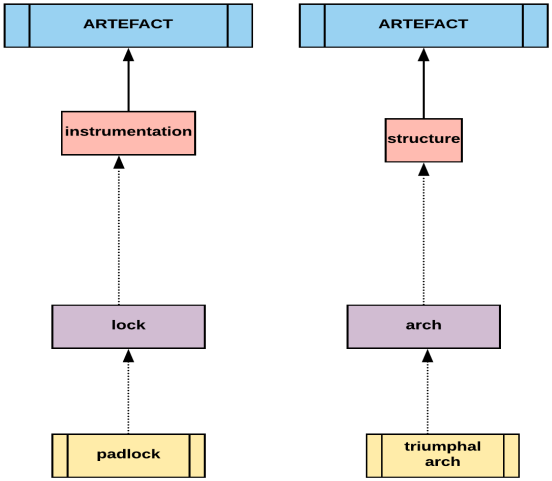


Figure 24: Four levels of hierarchical relationship exploited in ImageNet dataset.

Detailed study with all the other t-norms is given in Table 78 and Table 79 in Section A.0.4. With only 50% of the supervisions, the collective classification error reduces by 0.8% whereas using 100% of the supervisions, reduces the error by 0.14%. Also, the difference in error rate between using 100% supervisions and 50% supervisions is approximately 3.4%. This shows when there are missing labels in a large dataset or when it is very expensive and tedious to label all the datapoints, exploit-

Table 40: Comparison of the collective classification error rate for 1000 final classes using scarce training data and 26 intermediate predicates in non transductive mode with variable λ_l values. CNN, CC-WL, CC-P and CC-HW: convolutional neural network baseline, collective classification on neural network outputs using Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the table refer to the best performing network with 50% supervisions for different selection of t-norms.

Models	%Data	CNN	CC-WL	CC-P	CC-HW
Res-50	10	36.1	34.8	34.8	34.7
	20	34.2	32.4	32.5	32.4
	50	30.1	28.0	28.0	28.0
Res-101	10	35.1	33.4	33.5	33.4
	20	33.7	31.2	31.3	31.2
	50	28.5	25.3	25.4	25.3
Res-152	10	34.6	32.5	32.6	32.5
	20	32.6	30.9	30.9	30.8
	50	26.1	25.0	25.0	24.9
PRes-152	10	33.4	32.1	32.2	32.1
	20	31.0	31.0	31.0	30.9
	50	26.1	24.9	25.0	24.9
PRes-200	10	33.2	32.0	32.0	31.9
	20	30.4	29.0	29.0	28.9
	50	25.9	24.4	24.5	24.4
APN-200, $\alpha = 300$	10	32.9	31.8	31.8	31.7
	20	29.6	28.4	28.4	28.4
	50	25.8	24.2	24.2	24.2
APN-200, $\alpha = 450$	10	32.8	30.8	30.8	30.8
	20	29.2	27.4	27.4	27.4
	50	25.0	**23.6**	23.6	**23.5**

ing the background information available for the task in hand, logic rules can be easily formulated and using these rules can help in providing huge benefits in terms of resources used for labeling. There are several businesses that can make enormous profits by building deep learning applications like image recognition or video classification products without investing countless hours for manual sorting and labeling of data.

Experimental Simulation II :

This experimental simulation includes more taxonomical information than the experimental simulation I, resulting into more predicates and more rules. With more levels of hierarchy, greater amount of semantic information is being exploited. It includes 5 super classes, 576 intermediate classes and 1000 leaf classes. Some of extended hierarchical structures are given in Figure 24. The two examples below show the four levels of hierarchical structure exploited in this simulation.

Depending on the hierarchical relationship between the classes, softmax activations are used if the classes are mutually exclusive or else sigmoid activations are used in the classification layers. The different t-norms, the CNN architectures and the split of training, validation and test sets remain the same as in experimental simulation I in Section 5.5.3.

Table 41 and Table 42 compares the collective classification error rates in a constant weight based SBR and in a vanilla SBR respectively using different t-norms with all the taxonomical rules created with 1581 predicates. With the addition of more rules, the collective classification accuracies are improved and also the importance of optimization heuristics is better observed empirically.

With 4 levels of hierarchy and with about 580 intermediate predicates, the collective classification error rates are further reduced from the previous experimental simulation (with 26 intermediate predicates) by 0.4% in constant SBR and by 0.7% in vanilla SBR. It shows that using more rules help in increasing the accuracy of classification in the trained models but further future studies are required to understand which kind of rules help more and vice versa. And the results of this simulation shows the path towards future works in this direction.

The Additive Pyramidal Resnets which is the best performing CNN baseline with 4 levels of hierarchy reduces the error rates from 20.1% to 19.2% with the Half-Weight Combination setting. Comparisons with all other t-norms are given in Table 80 and Table 81 in Section A.0.4.

When three levels of hierarchy are exploited, the difference of collective classification error rate between using 50% supervisions and 100% supervisions is 3.4%, but when four levels of hierarchy are used, this gap is further reduced to only 2% in constant SBR and 1% in vanilla SBR. This shows that integration with logic and using of more rules, reduces the need to provide high number of supervised examples. Datasets like ImageNet have several levels of taxonomical information, and using this as background knowledge helps in creating a large knowledge base auto-

Table 41: Classification error rate for 1000 classes for different deep architectures in a constant weight based SBR using 581 intermediate classes. CNN refers to the baseline network, CC-WL, CC-P and CC-HW: collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	CC-WL	CC-P	CC-HW
Res-50	24.7	24.4	24.5	24.4
Res-101	23.6	23.5	23.5	23.5
Res-152	23.0	22.7	22.8	22.6
PRes-152	22.2	21.9	21.8	21.8
PRes-200	21.9	21.5	21.5	21.4
APN-200 $\alpha=300$	20.5	20.0	20.1	20.0
APN-200 $\alpha=450$	20.1	19.6	19.8	**19.6**

matically without much expertise from the WordNet. The results of limited number of supervisions for a constant SBR is shown in Table 43 for three different settings of t-norms. For all other t-norms and for vanilla SBR, the results are detailed in Section A.0.4 in Table 82 and Table 83 respectively.

Experimental Simulation III :

This experimental simulation further adds more predicates exploring all the nodes and internodes in the WordNet hierarchy. For some of the categories, there is a maximum of four levels of hierarchy to reach from the super class to the leaf class as shown in Figure 24, but for some of the leaf classes, there are a lot of intertwining nodes like: *ANIMAL* $\rightarrow$ *REPTILE* $\rightarrow$ *SAURIAN* $\rightarrow$ *LIZARD* $\rightarrow$ *GREEN_LIZARD*.

In this experimental simulation, all the intermediate classes are used to gain the full advantage of the taxonomical hierarchy in ImageNet. This almost triples the number of intermediate predicates than the experimental simulation II and also expands the knowledge base to a great extent. This simulation includes 5 super classes, 1600 intermediate classes and 1000 leaf classes. There are multiple classification output layers and depending on the hierarchical relationship between the classes, softmax as well as sigmoid activation is used in each of these output layers (softmax for exclusive classes and sigmoid for interdependent classes). Due to the space limitations, the diagrammatic representation of all the class

Table 42: Classification error rate for 1000 classes on ImageNet for different deep architectures in a vanilla SBR (λ_l^k) using 581 intermediate classes. CNN: the baseline network, CC-WL, CC-P and CC-HW: collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	CC-WL (λ_l^k)	CC-P (λ_l^k)	CC-HW (λ_l^k)
Res-50	24.7	24.2	24.3	24.2
Res-101	23.6	23.3	23.4	23.2
Res-152	23.0	22.2	22.2	22.2
PRes-152	22.2	21.4	21.4	21.3
PRes-200	21.9	21.2	21.3	21.2
APN-200 $\alpha=300$	20.5	19.7	19.8	19.7
APN-200 $\alpha=450$	20.1	19.4	19.5	**19.2**

hierarchy is not provided.

Table 44 and Table 45 compares the collective classification error rates in Weak-Lukasiewicz, Product t-norms and Half-Weight Combination setting in a constant weight based SBR and in vanilla SBR respectively using different t-norms. For the detailed comparison with all other t-norms, Table 84 and Table 85 are given in Section A.0.4 for constant SBR and vanilla SBR respectively.

With multiple levels of hierarchy and with about 1600 intermediate predicates, the collective classification error rates are reduced further by 1.2% in constant SBR and by 2.0% in vanilla SBR. It is also seen that in non transductive learning, if only 50% of the supervisions in a deep CNN model is used for a large dataset like ImageNet, in combination with the logical rules, the collective classification accuracy is as high as using 100% of the supervisions without logic rules in the same model. For example, the Additive Pyramid Resnet-200 with 1.3 million supervised data examples but no logic rules, has a classification error rate of 20.1%, whereas with only half of the supervised examples and using logical constraints, the error rate obtained through collective classification at the inference time is 20.9% in constant weight based SBR as shown in Table 46 and it is 20.7% in vanilla SBR as shown in Table 47. With half of the labeled data, the performance of the trained network is almost equivalent to the network trained with millions of examples. This helps in achieving benefits also in terms of computational overheads incurred in training these huge networks. Therefore, this final experimental simulation not only

Table 43: Comparison of the collective classification error rate for 1000 final classes using scarce training data with four levels of hierarchical information with 581 intermediate predicates in non transductive mode with constant λ_l values. CNN: baseline network. CC-WL, CC-P and CC-HW: collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination setting of t-norms. The bold numbers in the table refer to the best performing network with 50% supervisions using different selection of t-norms.

Models	%Data	CNN	CC-WL	CC-P	CC-HW
Res-50	10	36.1	34.9	34.9	34.7
	20	34.2	32.4	32.5	32.4
	50	30.1	27.6	27.6	27.5
Res-101	10	35.1	33.0	33.0	32.9
	20	33.7	31.0	31.0	30.9
	50	28.5	24.4	24.5	24.3
Res-152	10	34.6	31.9	32.0	31.9
	20	32.6	29.2	29.2	29.2
	50	26.1	24.4	24.5	24.3
PRes-152	10	33.4	31.4	31.2	31.1
	20	31.0	28.8	29.0	28.9
	50	26.1	24.1	24.2	24.1
PRes-200	10	33.2	31.1	31.2	31.1
	20	30.4	28.9	28.9	28.8
	50	25.9	23.1	23.2	23.1
APN-200, $\alpha = 300$	10	32.9	30.8	30.8	30.8
	20	29.6	27.0	27.0	27.0
	50	25.8	23.9	23.9	23.9
APN-200, $\alpha = 450$	10	32.8	29.7	29.8	29.8
	20	29.2	26.0	26.0	26.0
	50	25.0	**21.8**	21.8	**21.7**

proves SBR as a scalable semi-supervised framework, but also shows the efficiency of the novel backpropagation schema and optimization strategies succesfully implemented in this research work for a large dataset and a large knowledge base. The detailed results with different t-norms settings and scarce supervisions are given in Table 86 in Section A.0.4.

Table 44: Classification error rate for 1000 classes for different deep architectures in a constant weight based SBR using 1605 intermediate classes. CC-WL, CC-P and CC-HW: collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold number in the table refer to the best performing network using the best performing t-norm.

Models	CNN	CC-WL	CC-P	CC-HW
Res-50	24.7	24.1	24.2	24.1
Res-101	23.6	23.1	23.1	23.1
Res-152	23.0	21.2	22.1	21.2
PRes-152	22.2	21.0	20.9	20.9
PRes-200	21.9	20.9	20.9	20.8
APN-200 $\alpha = 300$	20.5	19.2	19.3	19.2
APN-200 $\alpha = 450$	20.1	18.9	19.0	**18.9**

Table 45: Classification error rate for 1000 classes for different deep architectures in a vanilla SBR (λ_l^k) using 1605 intermediate classes. CNN: the base line network, CC-WL, CC-P and CC-HW: collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination. The bold number in the refer to the best performing network with half weight combination setting.

Models	CNN	CC-WL (λ_l^k)	CC-P (λ_l^k)	CC-HW (λ_l^k)
Res-50	24.7	23.8	23.9	23.8
Res-101	23.6	22.9	23.0	22.9
Res-152	23.0	21.3	21.3	21.3
PRes-152	22.2	20.9	20.9	20.9
PRes-200	21.9	20.3	20.4	20.3
APN-200 $\alpha = 300$	20.5	18.9	19.0	18.9
APN-200 $\alpha = 450$	20.1	18.1	18.2	**18.1**

5.5.4 Discussion

To summarize, CNN models like Resnets, Pre-activated Resnets and Additive Pyramidal Resnets integrated with prior knowledge in the SBR framework when trained with limited number of supervised examples in non transductive mode, is seen to perform significantly better than the individual baseline CNNs. It is also seen that collective classification during inference helps to further improve the classification outputs of several classes of the ImageNet dataset. The best performing network is the

Table 46: Comparison of the collective classification error rate for 1000 final classes using scarce training data with 1605 intermediate predicates in non transductive mode with constant λ_l values. % Data: percentage of supervision used in each case. CNN: baseline network, CC-WL, CC-P and CC-HW: collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the Table refer to the best performing network with 50% supervisions for different selection of t-norms.

Models	%Data	CNN	CC-WL	CC-P	CC-HW
	10	36.1	33.9	34.0	33.7
Res-50	20	34.2	31.2	31.2	31.1
	50	30.1	26.5	26.8	26.4
	10	35.1	32.0	32.1	32.0
Res-101	20	33.7	30.1	30.1	30.0
	50	28.5	24.1	24.1	24.0
	10	34.6	31.0	31.0	31.0
Res-152	20	32.6	28.9	28.9	28.8
	50	26.1	24.0	24.0	24.0
	10	33.4	30.9	30.9	30.9
PRes-152	20	31.0	28.1	28.1	28.0
	50	26.1	24.0	24.0	23.9
	10	33.2	30.9	31.0	30.9
PRes-200	20	30.4	27.9	27.9	27.9
	50	25.9	22.2	22.3	22.2
	10	32.8	29.8	29.9	29.8
APN-200, $\alpha = 300$	20	29.6	26.8	26.8	26.8
	50	25.8	23.1	23.2	23.2
	10	32.8	28.2	28.2	28.1
APN-200, $\alpha = 450$	20	29.2	25.5	25.5	25.5
	50	25.0	**20.9**	21.0	**20.9**

Additive Pyramid Residual network with 200 layers that has a collective classification error rate of 18.1% which is 2% lower than the baseline. Although, this is not the present state-of-the-art, but the simulation results prove that any network used with SBR can achieve a lower error rate with the help of logical constraints and collective classification. Efficient Net (TL19) is the present state-of-the-art deep CNN, results published in the proceedings of ICML,2019, which has an error rate of 15.6% that is almost 2.5% lower than what is achieved in this research work. As a part of

Table 47: Comparison of the collective classification error rate for 1000 final classes of ImageNet using scarce training data and with 1605 intermediate predicates in non transductive mode with variable λ_l^k values. $\%$Data refers to the percentage of supervision used in each case. CNN refers to the baseline network, CC-WL, CC-P and CC-HW refers to the collective classification error rates for Weak Lukasiewicz, Product and Half-Weight Combination setting. The bold numbers in the Table refer to the best performing network with 50% supervisions for different selection of t-norms.

Models	%Data	CNN	CC-WL	CC-P	CC-HW
	10	36.1	33.7	33.8	33.7
Res-50	20	34.2	31.8	31.9	31.8
	50	30.1	26.9	27.0	26.9
	10	35.1	32.8	32.9	32.8
Res-101	20	33.7	30.9	31.0	30.9
	50	28.5	23.9	24.0	23.9
	10	34.6	30.5	30.6	30.5
Res-152	20	32.6	28.2	28.3	28.2
	50	26.1	23.7	23.7	23.7
	10	33.4	31.1	31.1	31.1
PRes-152	20	31.0	28.0	28.0	27.9
	50	26.1	23.9	24.0	23.9
	10	33.2	31.0	31.0	31.0
PRes-200	20	30.4	28.1	28.1	28.0
	50	25.9	22.4	22.5	22.4
	10	32.9	30.0	30.1	30.0
APN-200, $\alpha = 300$	20	29.6	26.8	26.84	26.8
	50	25.8	23.2	23.2	23.2
	10	32.8	29.0	29.0	29.0
APN-200, $\alpha = 450$	20	29.2	25.9	25.9	25.9
	50	25.0	**20.7**	20.7	**20.7**

the future work, it can be easily shown that using the same set of experimental conditions as used in this work, a higher classification accuracy can be obtained also with Efficient Net with ease. The other architectures that have better classification results on ImageNet than Additive Pyramid Nets like NasNet (ZVSL17), Dual Path Networks (DPN) (CLX$^+$17), AmoebaNet (RAHL18) etc. can be easily plugged as the learning machinery in SBR and the baseline results can be further boosted through logical rules and collective classification exploiting the class hierarchy

information.

Significant accuracy gains are observed with Weak Lukasiewicz t-norms and Half-Weight Combination setting, even when a very small knowledge base is used, as they have optimization advantages over other t-norms. As the knowledge base is expanded gradually, exploiting more sysnet information with more intertwined classes and increasing the number of logical rules, it not only improves the collective classification performance, but also reduces the need of labeled examples. However, with the increase of the number of non-convex constraints in the knowledge base, the need of carefully inducing the optimization strategies becomes greater, and if this is not done, it negatively effects the performance of the models. Labeling of huge datasets is not only an expensive task but often the data is poorly labeled and using this data for training makes the training input very noisy. Therefore, using prior knowledge helps in designing better learning systems. Scaling several logic integrated deep learners in the SBR framework for a large dataset like ImageNet justifies SBR as a powerful tool for real world classification applications in any field having a scarcity of labeled data and a huge database of unlabeled data. This study also shows the benefits of predicate based regularizers on the classification performance and makes a detailed comparison of constant regularizers and variable regularizers for each of the experimental simulations.

Similar, to the CIFAR-10 and CIFAR-100 experiments, the time overhead for performing collective classification is only a small fraction (approximately $\frac{1}{8} - \frac{1}{10}$) for each of these deep CNN networks compared to their training time when trained from scratch.

5.6 Summary

To conclude, this chapter demonstrates the capabilities of SBR when integrated with deep learners on small and large scale image classification problems. It empirically shows the advantage of each of the novel techniques described in Chapter 3. The chapter starts with justifying the benefits of replacing shallow networks like kernel machines with deep neural networks in SBR through experimental simulations. It then continues to further reconfirm the hypothesis that SBR is designed to work with scarce data examples and is able to reduce the dependency of deep networks on labeled data. The variety and richness of the CNN networks tested in this chapter also proves the learner agnostic nature of

SBR. Then finally the chapter adds on to explain the practical benefits of the optimization plans, the ability to add non convex constraints outlined in Section 3.4.2 through the detailed experimental analysis on different sized image datasets. These dimensions and capabilities of SBR have never been explored before in the literature. Therefore, presence of different features like using any type of learner effortlessly, training very deep networks with logical rules using efficient backpropagation, using variable regularizers, using constraints obtained through different fuzzy logics, reducing the need of large amount of supervisions and the ability to integrate concave constraints makes SBR an unified framework for any kind of image classification tasks.

Chapter 6

Video Classification

This chapter demonstrates a video classification and automatic tool annotation process for cataracts surgery (on publicly released CATARACTS dataset) using prior information in a probabilistic framework with deep learners (Convolutional Neural Networks and Markov Random Fields). Here, an ensemble of Convolutional Neural Networks is integrated with a Markov Random Field (CNN-MRF) (HLC$^+$19). This framework is also compared with the Semantic Based Regularization framework described in Chapter 3 and Chapter 5 using the same ensemble of CNN networks as DResSys as the underlying learners. Therefore, this chapter also emphasizes the power of SBR in solving multi-label video classification problem by using prior knowledge in the form of constraints during test time. It provides empirical evidence for the flexibility of SBR to be applied in other domains like video classification or video sequencing. The general domain knowledge about the cataracts surgery is exploited along with the temporal coherence among consecutive video frames to design the knowledge base (KB) in SBR. Extending deep learning techniques independently to each video frame often leads to temporal inconsistencies in the predictions. For example: a tool appearing in one video frame is likely to be present in its prior and posterior video frame as well, if this doesnot happen, it is most probably inconsistent. This phenomenon is known as temporal coherence or temporal consistency. This property of videos is exploited through rules in SBR and these rules aim to improve the CNN ensemble predictions through collective classification during inference. With the help of collective classification, SBR aims to compete with well established traditional temporal smoothing techniques

like median filtering and random field optimizations. Application of SBR on video classification problem is explored for the first time in this research work.

6.1 Dataset and Problem Description

The CATARACTS dataset was released as a part of the CATARACTS GRAND CHALLENGE [1] (The Challenge on Automatic Tool Annotation for CATARACT Surgery), organized in 2017 for designing efficient solutions for surgical workflow analysis, with potential applications in report generation, surgical training and even real-time decision support. 14 teams participating in the challenge, submitted different deep learning based approaches but DResSys, was the winning entry in the CATARACTS challenge, designed as a part of this research work in collaboration with D-Wave Systems (HLC$^+$19). The framework is able to precisely locate surgical tools in the video frames, and also indicate which tools are being used by the surgeon at each instant of the surgery. The tool annotation performed using DResSys in some cases are better than human interpretations or atleast comparable to the human annotations. The motivation of this challenge from the perspective of the benefits provided to the medical community as well as the previous works related to this field are discussed in more details in the works of Hajj et al. (HLC$^+$19). In this research work, the design and the motivation of DResSys is outlined and it is also shown that when the CNNs of the DResSys are integrated in the SBR framework and prior knowledge is exploited, the final classification results are comparable to the performance of DResSys. This is a real world application of SBR and it lays the stones of new research that can be conducted on different problems of video classification like logo recognition in videos, activity recognition in sports videos etc.

6.2 Experimental Analysis

The dataset for this challenge consists of 50 videos of phaco-emulsification cataract surgeries performed in Brest University Hospital between January 22, 2015 and September 10, 2015. The surgeries had to be conducted because of several reasons which included age related cataracts,

[1] https://cataracts.grand-challenge.org/

traumatic cataracts and refractive errors. The patients involved in the surgery were aged 61 years on an average (minimum: 23, maximum: 83, standard deviation: 10) including 38 females and 12 males. Consent was obtained from all the patients before using their data for this challenge. Surgeries had been performed by three surgeons: A renowned expert who had an experience of 48 surgeries, an one-year experienced surgeon with only 1 surgery experience and an intern also with 1 surgery experience. Surgeries were performed under an OPMI Lumera T-microscope. Videos were recorded with a 180I camera (Toshiba, Tokyo, Japan) and a MediCap USB200 recorder.

Each recorded video frame is 1920×1080 pixels and the frame rate of recording is approximately 30 frames per second (fps). Each video has been recorded for a duration of approximately 10 minutes and 56 seconds on average and in total, more than 9 hours of surgery videos are present. For the purpose of the challenge, the dataset is evenly divided into a training set and a test set of 25 videos each. There is a total of 21 surgical and non-surgical tools as shown in Figure 25 that have been used during these surgeries. They are listed in *Tool* column of Table 48. These 50 videos are divided into training and test sets in such a way that each tool appears in the same number of videos for both the sets. No exclusive validation set is provided, during the challenge.

To annotate the ground truth for the training and test sets, an elaborative procedure has been followed. All surgical tools in the videos were first enumerated and labeled by the surgeons to specify the name of each tool. Then, the usage of each tool in the videos was annotated independently by two non-clinical experts. A tool was considered to be in use whenever it was in contact with the eyeball and a timestamp was recorded by both the experts at that very instant, and also when it stopped touching the eyeball. Up to three tools could be used simultaneously: two by the surgeon performing the surgery and sometimes one by an assistant. Annotations were performed at the level of the video frames, using a web interface connected to a SQL database. Finally, annotations from both the non-clinical experts were adjudicated. Adjudication was a process undertaken when the two experts didn't agree on the name of the tool. During such times, the two experts watched the video together and jointly determined the actual tool usage and from this a probabilistic reference standard was obtained:

- 0 is the value of assigned when both experts agree that the tool is not being used.

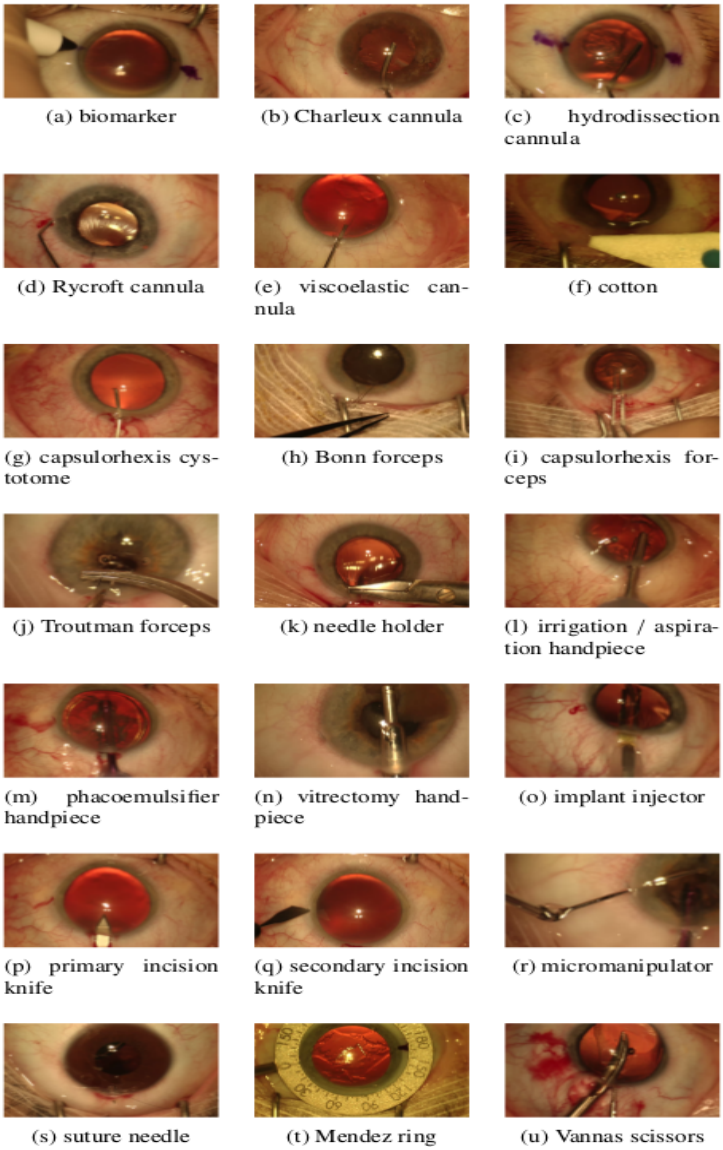


Figure 25: Tools used in the cataracts surgery (HLC$^+$19).

Table 48: Statistics about tool usage annotation in the CATARACTS dataset. The first two columns indicate inter-rate agreement (Cohens kappa), before and after adjudication (Adj. is the abbreviation for adjudication); the largest changes are in bold. The last column indicates the prevalence of each tool in the training subset (percentage of training frames), ignoring the frames where the experts disagree about the usage of that tool, even after adjudication.

Tool	Before Adj.	After Adj.	% of frames
Biomarker	0.835	0.835	0.02
Charleux cannula	0.949	0.963	1.79
Hydrodissection cannula	**0.868**	**0.982**	2.43
Rycroft cannula	**0.882**	**0.919**	3.18
Viscoelastic cannula	**0.860**	**0.975**	2.54
Cotton	0.947	0.947	0.75
Capsulorhexis cystotome	0.994	0.995	4.42
Bonn forceps	0.793	0.798	1.10
Capsulorhexis forceps	0.836	0.849	1.62
Troutman forceps	0.764	0.764	0.26
Needle holder	0.630	0.630	0.08
Aspiration handpiece	0.995	0.995	14.20
Phacoemulsifier handpiece	0.996	0.997	15.30
Vitrectomy handpiece	0.998	0.998	2.76
Implant injector	0.980	0.980	1.41
Primary incision knife	0.959	0.961	0.70
Secondary incision knife	0.846	0.852	0.52
Micromanipulator	0.990	0.995	17.60
Suture needle	0.893	0.893	0.22
Mendez ring	0.941	0.953	0.10
Vannas scissors	0.823	0.823	0.04

- 1 is assigned when both experts agree that the tool is being used.
- 0.5 is assigned when the experts disagree among each other.

Inter-rate agreement, before and after adjudication, has been measured using Cohen's Kappa (as described in Section A.0.5) for each tool across all the video frames and have been reported in Table 48. A chord diagram illustrating the co-occurrence of tools in training video frames is reported in Figure 26. Tool usage predictions on the test videos are evaluated against the annotation of the ground truth using the following

procedure : For each tool label T, the annotation performance for tool T is defined as the area $A_z(T)$ under the receiver-operating characteristic (AUC-ROC) curve. Frames associated with a disagreement between experts are ignored when computing the ROC curve. Then, a mean $A_z(T)$ value over all tool labels T is defined.

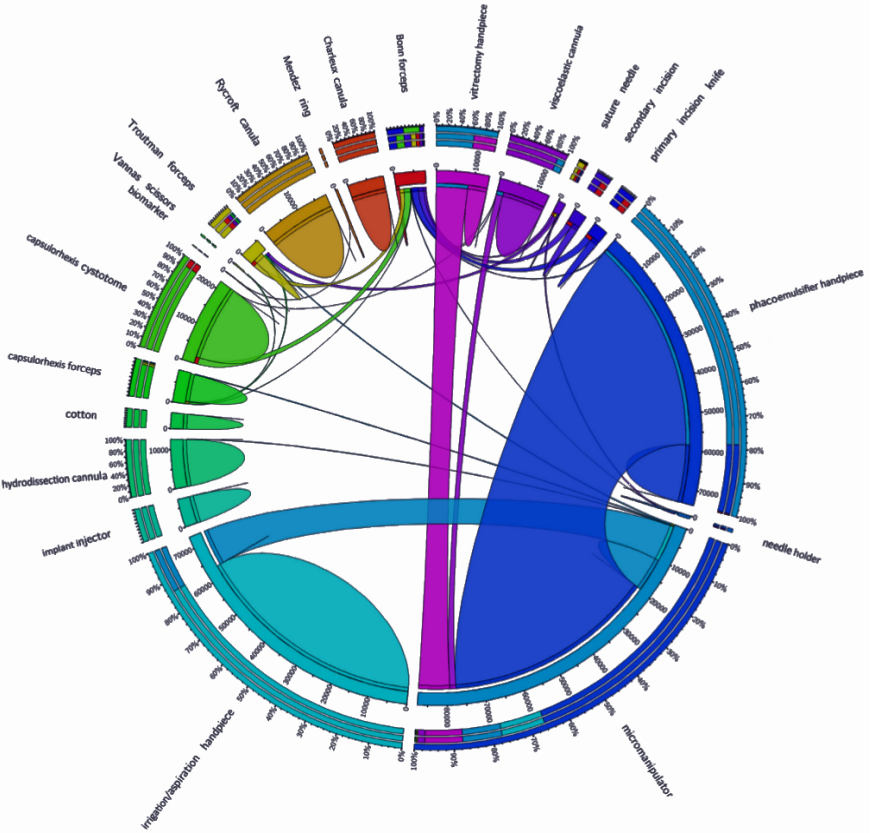


Figure 26: Chord diagram illustrating tool co-occurrence in training video frames (HLC$^+$19).

6.2.1 Overview

DResSys uses an ensemble of different deep CNN architectures (Resnet-50 (HZRS15a), Inception-$v4$ (SVI$^+$16) and two NASNET-A (ZVSL17) networks with different input sizes) to make predictions on individual video frames. Since there is always a temporal connectivity or correlation among the adjacent frames, the predictions of CNN ensemble are further smoothed with different procedures described below to get the final predictions:

- Using a simple median filtering technique.
- Using markov random fields.
- Using logical rules and collective classification in the SBR framework.

6.2.2 Data Pre-processing

Distributions of the labels in each of the 25 videos of the training set is plotted (similar to Figure 26), that reflects the frequency of each annotated tool in each video. Based on this result, the training data is divided into two subsets: the training set and the validation set approximately in the ratio of 5:1. It is necessary to have all the tools in the training set to get the trained model that can be used for prediction, the division is made in the best possible way so that most of the tools have the same distribution in the training and the validation set but if a tool is present in only one of the videos, that video is placed in the training set. Validation set therefore, consists of the videos numbered *train04*, *train12* and *train21* while the remaining videos are placed in the training set. Images are extracted from these videos using *ffmpeg*. The training videos are extracted following two different procedures:

- 6 frames per second are extracted at an uniform rate from the training set. This generates a total of approximately 70K training images.
- The tool co-occurrence as highlighted in Figure 26, the class imbalance problem exists amongst the tools, therefore, a selective measure of frame extraction is followed. All the frames within videos containing the rare tools (for example, *Biomarker*, *Vannas scissors*)

are selected, but in parts of the videos with common tools, the sample rate is maintained at 6 frames/sec. Further, 40K frames at uniform rate are selected from amongst training frames that have no tools. This selective frame sampling method provides a total of approximately 100K training images.

In both of the above scenarios, the validation set is constructed by extracting 6 frames per second uniformly from the validation videos. However for the test set, again two procedures are followed:

- 3 frames are extracted per second from the test videos. The trained model performs the predictions on these frames and for the remaining frames, the predictions are obtained through a simple interpolation technique.
- The trained model performs predictions on all the frames of the test videos. But this is computationally heavier and more time consuming.

6.2.3 CNN Models

DResSys is a Tensorflow and Python based framework. The network architecture used for training is a CNN ensemble of a 50-layer Residual Network, Inception-$v4$ and two NASNet-A networks. Each of these networks are trained individually and then the ensemble is built by taking a weighted geometric mean. The Resnet and the Inception-$v4$ architectures, pretrained on ImageNet models are trained on Cataracts images of 540×960 pixels. NASNet-A is a much larger network requiring more GPU memory, so two NASNet-A networks are trained with smaller image inputs of 270×480 pixels and 337×600 pixels. The training data is augmented by random horizontal flipping and cropping of images. Each of the networks are trained on a GTX 1080Ti GPU.

Residual nets, Inception nets and NasNet-A nets are trained with the Adam optimizer (KB14) using a weighted sigmoid cross entropy loss on the tool labels to further help with the class imbalances. The weights assigned to the different tool labels is chosen through cross validation. Some of the rarest tools are not present in the videos of the validation set, therefore for these tools, the weight is taken as 1, the highest possible weight from the range of $[0, 1]$. Training ran for at most 13 epochs with a batch size of 4 and a validation batch size of 7. The initial learning

rate for each network is 0.003 and it is decayed after every 6 epochs by a weight decay of 0.1.

6.2.4 Post Processing - Temporal smoothing

Several smoothing approaches are explored to capture the dependence of tool labels across consecutive frames in the videos. The goal of the post processing step is to improve the classification output of the deep CNN ensemble network by exploiting the temporal relationship that exists between video frames.

Median Filtering :

The first three experimental simulations as described in the next Section 6.2.5 are based on a simple median filtering method with a kernel size of 61. This technique tries to establish a temporal correlation by running a median filter through each video frame and replacing each frame with the median of labels of its neighboring video frames during inference.

Markov Random Field Based Smoothing Model :

The fourth experimental simulation, designs the DResSys that includes a Markov Random Field (MRF) based temporal smoothing model. The MRF model provides a probability distribution across the time dependent label space. Assume that $\mathbf{y} = y_1, y_2, \ldots, y_T$ represents the binary label vector for a given tool in the output predictions where $y_t = 1/0$ which indicates the presence/absence of the tool in a particular video frame. The proposed MRF model has a chain-like structure and defines a conditional probability distribution $p(\mathbf{y}|\mathbf{x}) \sim exp(E(\mathbf{y};\mathbf{x}))$ for the label vector $\mathbf{y}$ given the video $\mathbf{x}$ using an energy function $E(\mathbf{y};\mathbf{x})$

$$E(\mathbf{y};\mathbf{x}) = \sum_{t=1}^{T} a(s_t) y_t + \frac{w}{2} \sum_{t=1}^{T} \sum_{n \in N(t)} y_t y_n \tag{6.1}$$

where $N(t) = \{t-19, t-17, \ldots, t-1, t+1, \ldots, t+17, t+19\}$ represents the set of neighboring frames for the t^{th} video frame, and therefore, provides a long-range temporal connectivity. The total variation is a lag of 38 frames that corresponds to approximately 1 second of recorded video.

In Equation 6.1, $a(s_t)$ is the bias for the t^{th} frame's label which is computed by shifting and scaling the output of the CNN ensemble frame-level prediction score s_t of frame t. The scalar coupling parameter w in Equation 6.1 enforces the label agreement between the neighboring frames when $w > 0$.

The w parameter, the shift and scale parameters of the linear map $a(s_t)$ are all set by a grid search over the validation set that maximizes the AUC-ROC average score for all the 21 tools. The shift, scaling and w parameters are shared across all tools. Nevertheless, due to the shifting and scaling of the ensemble neural network output (which does differ across categories) a tool-specific MRF model is formed which is robust against overfitting. The MRF model, $p(\mathbf{y}|\mathbf{x})$, represents the joint probability distribution for all the labels in the temporal domain of a tool. Lastly, in order to process the videos efficiently the MRF model is formed in smaller segments of length ∼20,000 frames instead of the full length of the video.

Collective Classification using Logical Rules in SBR :

The MRF based temporal smoothing of DResSys is compared with the smoothing performed using the collective classification module of the SBR framework. Temporal correlations among the video frames are established through the logical rules. Some other specific prior knowledge obtained from the problem domain at hand is also expressed in the form of rules (For example: If a *Surture_needle* is present, a *Needle_holder* is also present in that particular video frame and vice versa). The list of logical rules used for this simulation is tabulated in Table 49. The temporal correlations are expressed using a new given predicate called as *Succ* (abbreviation of *succession*) that indicates the next frame. The grounding for this predicate is built offline and is given as input to the SBR. This predicate is associated with each of the 21 tool labels to indicate whether the given tool is present or absent in the next frame i.e the tool label in the next frame is 0 or 1 depending on the tool label of the current frame. Therefore, this predicate when grounded with any tool label predicate it helps to establish a consistency or a manifold. The tool labels in the preceding or the succeeding frames in the temporal domain should be related and maintain a regularized manifold. Therefore, these temporal rules along with the problem domain rules are collectively used for collective classification in SBR on the test set predictions obtained as the output of the CNN ensemble. This help to smooth the predictions of the

Table 49: Prior knowledge for CATARACTS dataset.

$\forall x$ Biomarker$(x)\vee$ Charleux_cannula$(x)\vee$ Hydrodissection_cannula(x)
$\vee$ Rycroft_cannula$(x)\vee$ Viscoelastic_cannula$(x)\vee$ Cotton$(x)\vee$
Capsulorhexis_cystotome(x) $\vee$ Bonn_Forceps$(x)\vee$ Capsulorhexis_forceps(x)
$\vee$ Troutman_forceps$(x)\vee$ Needle_holder$(x)\vee$ Aspiration_handpiece(x)
$\vee$ Phacoemulsifier_handpiece$(x)\vee$ Vitrectomy_handpiece$(x)\vee$ Implant_injector(x)
$\vee$ Primary_incision_knife$(x)\vee$ Secondary_incision_knife$(x)\vee$ Mendez_ring(x)
$\vee$ Micromanipulator(x) $\vee$ Suture_needle$(x)\vee$ Vannas_scissors(x)

$\forall x$ Biomarker$(x) \Rightarrow \neg$ Charleux_cannula$(x) \wedge \neg$ Hydrodissection_cannula$(x)\wedge$
$\neg$ Rycroft_cannula$(x)\wedge \neg$ Needle_holder$(x) \wedge \neg$ Cotton$(x) \wedge \neg$ Bonn_Forceps(x)
$\neg$ Capsulorhexis_cystotome$(x) \wedge \neg$ Capsulorhexis_forceps$(x) \wedge$
$\neg$ Troutman_forceps$(x) \wedge\neg$ Mendez_ring(x) Aspiration_handpiece$(x)\wedge$
$\neg$ Phacoemulsifier_handpiece$(x)\wedge \neg$ Primary_incision_knife$(x)\wedge$
$\neg$ Implant_injector$(x) \wedge \neg$ Vitrectomy_handpiece$(x) \wedge\neg$ Micromanipulator(x)
$\wedge\neg$ Viscoelastic_cannula(x) Secondary_incision_knife$(x)\wedge\neg$ Suture_needle(x)
$\wedge\neg$ Vannas_scissors(x)

$\forall x$ Needle_holder$(x) \Rightarrow$ Suture_needle(x)
$\forall x$ Suture_needle$(x) \Rightarrow$ Needle_holder(x)

$\forall x\forall y$ Succ$(x, y)\wedge$ Biomarker$(x) \Rightarrow$Biomarker(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Charleux_cannula$(x) \Rightarrow$ Charleux_cannula(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Hydrodissection_cannula$(x) \Rightarrow$ Hydrodissection_cannula(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Rycroft_cannula$(x) \Rightarrow$ Rycroft_cannula(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Viscoelastic_cannula$(x) \Rightarrow$ Viscoelastic_cannula(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Cotton$(x) \Rightarrow$ Cotton(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Capsulorhexis_cystotome$(x) \Rightarrow$ Capsulorhexis_cystotome(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Bonn_Forceps$(x) \Rightarrow$ Bonn_Forceps(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Capsulorhexis_forceps$(x) \Rightarrow$ Capsulorhexis_forceps(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Troutman_forceps$(x) \Rightarrow$Troutman_forceps(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Needle_holder$(x) \Rightarrow$ Needle_holder(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Aspiration_handpiece$(x) \Rightarrow$ Aspiration_handpiece(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Phacoemulsifier_handpiece$(x) \Rightarrow$ Phacoemulsifier_handpiece(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Vitrectomy_handpiece$(x) \Rightarrow$ Vitrectomy_handpiece(y)
$\forall x\forall y$ Succ$(x, y)\wedge$Implant_injector$(x) \Rightarrow$ Implant_injector(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Primary_incision_knife$(x) \Rightarrow$ Primary_incision_knife(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Secondary_incision_knife$(x) \Rightarrow$ Secondary_incision_knife(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Micromanipulator$(x) \Rightarrow$ Micromanipulator(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Suture_needle$(x) \Rightarrow$ Suture_needle(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Mendez_ring$(x) \Rightarrow$ Mendez_ring(y)
$\forall x\forall y$ Succ$(x, y)\wedge$ Vannas_scissors$(x) \Rightarrow$ Vannas_scissors(y)

ensemble deep neural network and obtain a more generalizable model.

6.2.5 Results

The prediction probability of each of the trained CNN is obtained on the validation set. Since 6 frames per second are sampled from each of the validation video, the prediction probability of the missing frames are interpolated using a $1D$ linear interpolation on the sampled frames. This interpolation technique is chosen because the consecutive frames in the videos always maintain the property of temporal similarity. The ROC score for each trained frame-level CNN model is then calculated. For the CNN ensemble, the ROC scores are aggregated using a weighted geometric mean. The hyperparameters of each individual CNN, the weights assigned to the aggregated CNN ensemble, the parameters of median filtering and the parameters of the MRF model are all set by performing a grid search over the validation set.

Although, two tools namely *Viterectomy_handpiece* and *Vannas_scissors* are not present in the validation set, therefore, there is a small risk in using these tuned hyper parameters on validation set during inference, but it is assumed that if the hyper-parameters work well for 19 tools, it should not fail in predicting the remaining two rare tools that are absent from the validation set. It is later seen that this assumption works well for the test set.

As already explained in the earlier sections, the main goal of this research work is to accurately annotate the tools in the video frames of the test set, however this raises several challenges that needs to be addressed during these experimental analysis. The best possible framework design needs to be determined that can be used for solving this video classification problem. Some of the important challenges faced are:

- Determining the correct sampling technique of the training frames to get the best trained CNN model.
- To effectively deal with the class imbalance problem of the tools in the training videos.
- Choosing the CNN model to be trained and to find the suitable technique of combining different CNN models if an ensemble is to be build.

- To determine the optimal input size of the images that helps the deep CNN networks to learn rich feature representations without making the training time overhead abnormally high.

- To determine an optimal post processing technique to preserve the temporal dependence among the consecutive video frames.

There are five different simulations performed under different conditions with different CNN networks, training parameters, post processing techniques etc. each aimed at meeting one or more or all of the above mentioned challenges.

Experimental Simulation I :

In the first experimental simulation, the training is performed on uniform sampled frames at 6 fps for each of the training videos. Three different deep CNN architectures are trained on them: a 50 layered Resnet, an Inception-$v4$ net with input images of size 540×960 pixels and one NasNet-A architecture with an input image size of 270×480 pixels. A CNN ensemble is constructed taking a weighted geometric mean of the predictions of the individual CNNs which is then smoothed using median filtering with a kernel size of 61. The average prediction of all the tools using each of the experimental simulations are aggregated and reported in Table 50. However, per tool predictions for this simulation is reported in Table 87 which shows the area under the AUC-ROC curve for each tool on the test set. It shows that some of the common tools like *Phacoemulsifier_handpiece* and *Micromanipulator* have a very high AUC-ROC score of approximately 0.999 whereas for the tools that are very rare like *Biomarker* and *Mendez_ring* have low AUC-ROC score approximately between 0.72 to 0.78. This actually motivated the usage of selective sampling of the training frames based on the presence of the tools in each of the videos. Median filtering seems to perform well in establishing the temporal coherence between the consecutive video frames. The average prediction of all the tools after application of median filtering is seen to increase by 1.5%. During inference, obtaining predictions on the subsampled frames at a rate of 3 fps and using linear interpolation on the remaining frames is not markedly different from obtaining predictions on all the test frames, therefore, the results are not reported separately.

Experimental Simulation II :

In this experimental simulation the training is performed on selective sampled frames based on the co-occurrence of tools as shown in Figure 26 but the trained CNN networks are identical to the experimental simulation I. This simulation also relied on median filtering to establish temporal correlation among the video frames. Table 88 reports the area under the ROC curve for each tool on the test set using this experimental setup. However, the average of all the tools and a comparison with other simulations is reported in Table 50. In this simulation, with selective sampling of the rare tools like *Biomarker*, the AUC-ROC score of the tool improves to 0.9 from 0.72 and hence the overall average also rises by 1.67% from the previous experimental simulation and by 3.2% from the baseline CNN ensemble without temporal smoothing. The average AUC-ROC score now becomes 0.970 from 0.954 in the experimental simulation I. Some of the tools however, has a AUC-ROC score slightly lower than the previous simulation but this is because with the new set of training frames, the deep CNN models are trained having different weights that leads to different predictions, but the increase/decrease in the AUC-ROC score of the common tools are in a tolerance of $0.2 - 0.8\%$. In this simulation, predictions are obtained on all the frames. Selective sampling actually helps to deal with the class imbalance problem very effectively and opens new doors to the real time usage of such a framework for inter-operative decision support during cataract surgeries.

Experimental Simulation III :

Since the experimental simulation II suggests an improvement on the overall performance of the AUC-ROC scores using selective sampling of the training frames, this experimental simulation also uses selective sampled frames. When the individual CNN architectures are evaluated on the validation set, it is seen that NasNet-A is the highest performing deep CNN network. Another NasNet-A model is therefore added to the CNN ensemble to further boost up the ROC scores of the individual tools. Thus this experimental simulation meets the first 3 challenges mentioned above. It makes a wise sampling of the video frames, tries to alleviate the class imbalance problem and also choose top performing CNN networks for building the CNN ensemble. The temporal smoothing is still performed using median filtering. Table 89 reports the area under the ROC curve for each tool on the test set using this simu-

Table 50: Area under ROC curve for an average of 21 tools in the four different experimental simulations. The column *Ensemble* refers to the CNN networks used, *Sample* refers to the sampling techniques of the video frames during training and *Smooth* refers to the temporal smoothing technique used. Exp # refers to the experimental simulation number. CC refers to collective classification in SBR with the rules given in Table 49.

Exp#	Ensemble	Sample	Smooth	ROC-AUC
				DResSys
	Res-50, Incep-$v4$, NasA-small	Random	No Smoothing	0.940
I	Res-50, Incep-$v4$, NasA-small	Random	Median Filtering	0.954
II	Res-50, Incep-$v4$, NasA-small	Weight Based	Median Filtering	0.970
	Res-50, Incep-$v4$, NasA-small, NasA-large	Weight Based	No Smoothing	0.972
III	Res-50, Incep-$v4$, NasA-small, NasA-large	Weight Based	Median Filtering	0.980
IV	Res-50, Incep-$v4$, NasA-small, NasA-large	Weight Based	MRF Filtering	0.997
V	Res-50, Incep-$v4$, NasA-small, NasA-large	Weight Based	SBR (CC)	0.995

lation. It shows that addition of another NasNet-A architecture with a larger input size of 337×600 pixels to the CNN ensemble improves the AUC-ROC of most of the tools by 0.04% to 0.08% and the overall average AUC-ROC of all the tools is reported in Table 50. During inference in this simulation, predictions are obtained on all the frames and this builds a more accurate annotation model. The AUC-ROC score of this ensemble of CNNs without median filtering is 0.972 which is already higher than the previous experimental simulation and therefore, the contribution of the additional NasNet-A architecture added to this ensemble can easily be understood. However, using of median filtering further boosts the AUC-ROC score to 0.980 with the same CNN ensemble.

Experimental Simulation IV :

This experimental simulation uses the CNN ensemble from the previous simulation. However in this analysis, the predictions on the test set are smoothed with a different temporal smoothing technique called as the MRF smoothing as defined earlier in this chapter. Using a MRF model, the average AUC-ROC scores of the tools have been boosted to a significant extent of 1.73% from the median filtered smoothed model. It is seen that almost all the tools have a score very close to 1. The detailed results on the test set is reported in Table 90 and the average is reported in Table 50. This system with four different CNN networks forming an ensemble with a combination of a MRF based smoothing model is called as DResSys. DResSys was built as the part of the CATARACTS challenge. The results of all other competing solutions for the CATRACTS challenge and the detailed comparison among them are given in the works of Haji et al. (HLC$^+$19). Although, it might seem that these solutions differ by a very thin margin, but during a surgical procedure, a small precision matters a lot. It has also been mentioned that the solutions were compared to the human level performance and DResSys was found to be very close to human grader results (the AUC-ROC average of the human grader is 0.998). Although it is not better than a human annotator in any of the tools except for Cotton where DResSys even beats a human expert in annotating it across the video frames in the test set. Thus DResSys proves to be an effective way of automating the tool annotations which not only saves a lot of time but also brings sensitivity almost equal to that of a professional human interpreter.

Experimental Simulation V :

This experimental simulation performs collective classification with the rules described in Table 49 in the SBR framework. Collective classification exploits the temporal correlations among the video frames to establish a regularized manifold between the preceding and the succeeding frames. The predictions of the CNN ensemble obtained on the test set in the experimental simulations III and IV are given as inputs during collective classification in SBR. Some of the logical rules used establish relationships among the 21 output tool categories whereas the others are used to express the temporal relationships using the *Succ* predicate. Using collective classification on the test set in SBR, the overall AUC-ROC scores of the tools have been boosted to a significant extent of 2.37% from

the corresponding baseline output of the CNN ensemble and by approximately 4.3% from the experimental simulation I. Like the MRF based smoothing model, SBR also brings the ROC scores of several tools very close to 1. The detailed results of the individual tools is reported in Table 91 and the average is reported in the bottom row of the Table 50. These results are slightly lower than the previous simulation, the winning entry of the CATARACTS challenge (DResSys). It varies by a very small margin of 0.002 in the AUC-ROC score value which probably is not very significant and also it is very close to human level performance. Hence it demonstrates the power of SBR in establishing temporal coherence or a regularized manifold in video classification problems.

The experimental simulation IV and V also addresses all the challenges that are mentioned in the beginning of Section 6.2.5. This research work compares the SBR framework with the winning entry DResSys (the state-of-the-art results in this problem) and achieves satisfying performance. Both of these frameworks use deep CNN ensemble and encode information from this ensemble into the post processing step. The experimental simulation V also demonstrates that the prior knowledge when used in collective classification mode in SBR can make significant contribution in solving video classification problems. The constraints applied during the test time, force the test patterns (test video frames in this task) to respect them and establish consistency or a manifold (temporal consistency in this task). Through, this task, it is reconfirmed that SBR is a powerful and versatile framework that can tightly integrate logical rules in deep learners effectively.

6.3 Summary

To summarize, the chapter describes an effective framework, DResSys that is designed to perform cataracts tool annotation and in a form it encodes prior knowledge from the CNN ensemble to establish temporal relationships. Although, it has not been tested for other applications, but still the framework is very flexible and has a lot of reusable modules that can be used for multi-label video sequencing problems using deep learners. There is a huge scope to use DResSys in the future for several practical applications like in activity or motion recognition in videos for different industries.

In this problem, DResSys achieves an average AUC-ROC score of 0.997 which is higher than any other competing solutions (HLC$^+$19) or

frameworks and therefore is the present state-of-the-art for this application. SBR framework however, provides a very tough competition to DResSys and its contribution to this video classification problem is the core part of this research work.

There has been a lot of challenges in the design of the deep learning machinery used in DResSys and SBR that have been imposed mostly by the dataset like the distribution of the tools have been highly unequal and tools are often visually similar to each other (like Cannule and Forceps). There have also been several other problems like uneven illumination, zoom level variations, partial tool occlusion, motion and out-of-focus blur in the input images. Most of these problems have been adequately handled in either by the deep CNNs themselves, or by using of data augmentation techniques, or using of re-sampling strategies, or designing of adequate cost functions and smoothing techniques etc. The temporal sequencer is built with the motivation of using the prior knowledge from the trained CNN ensemble in the MRF model to establish a temporal connectivity and correcting the output errors of the CNNs. This sequencer is seen to boost the performance of most of tools and develop a system that can be used in real time for the inter-operative decision support during cataract surgeries.

The collective classification framework of SBR on the other hand encodes prior information in the form of logical constraints and proves itself to be very beneficial for this kind of real world problems. The potential of SBR demonstrated in this work also paves new path of research for using the SBR framework in similar problems like annotating human movements in sports videos, gesture recognition in the gaming videos etc. Instead of using rules only during the inference time through collective classification, it would be interesting to see how logical rules help in such problems when joint learning takes place. This would help in exploring the full potential of SBR also for video classification problems as it was empirically demonstrated for image classification in Chapter 5.

Chapter 7

Contaminant Separation

This chapter consists of the second part of the thesis and is focused on image segmentation using deep convolutional neural networks. A few frameworks dedicated to segmentation problems in computer vision have been already discussed in Section 4.2. This chapter is more application oriented but it is closely related to the other part of the thesis as it tries to solve a vision task and uses deep learning methodologies to achieve the goal. In this chapter, the frameworks are adapted and compared against each other in solving a real world problem of separating contaminants from microscopy images for cleanliness analysis. However, the novel methodologies used in these frameworks are reusable and scalable to other image and video segmentation problems of the real world scenarios. This research work can also be viewed from the perspective of using domain specific image segmentation techniques (for example: watershed transform and connected component analysis algorithms) that are integrated with neural networks in a cascaded fashion, therefore building an end-to-end solutions for industry based contaminant separation problem. The contaminant separation problem is defined as locating, segmenting, analyzing and aiding the automatic removal of the random shaped particles from their low contrast background (filters) images using an end-to-end framework that can be integrated with optical microscopic acquisition systems. The analysis of the particles during the process of removal involves determining the size, shape and the count of the particles.

In this work, for the first time, cheap, accurate and time efficient deep learning frameworks are proposed for automatic segmentation and clas-

sification of contaminants, replacing the expensive slow systems existing in the automobile, aerospace or manufacturing industries. Apart from the novelty in terms of the application, there have also been new algorithmic techniques like region proposal selection criteria, different cost functions, optimization tweaks, cascaded post processing tricks etc. introduced as a part of this research work that makes improvement over the existing benchmark frameworks like MaskRCNN (HGDG17) and UNET (RFB15) if they would be used in their original forms to solve this problem.

7.1 Introduction

Automotive, industrial and aerospace manufacturers are under constant pressure to reduce their costs of production, improve operations, lower the selling prices and still have high profits. To accomplish these goals, manufacturers regularly seek new ways to improve efficiency of their production process and continue delivering high quality products at profitable prices. Maintaining specified levels of fluid and component cleanliness can have a major effect on the reliability of the components and systems involved, ultimately impacting the overall efficiency of the operation. The entire system is subject to failure if any part of the fuel system, braking system, hydraulic circuit, or electronics does not meet the cleanliness requirements.

Key to managing component cleanliness is the measurement and analysis of contaminants present in the production parts. Contamination analysis is therefore critical in understanding the source and the root cause of component failure. There are several basic types of contamination (Win) as listed below:

- *Particulate contamination :* Inanimate and inorganic particles such as the residue from a manufacturing process (for example: particles from abrasion or grinding, fibers from clothes or dust from the local surroundings), and also biological and organic, such as bacteria, fungi, spores, flakes of shed skin, or cell fragments. Example of residues are shown in Figure 27.

- *Molecular contamination :* Organic and inorganic films, such as the residue from additives during manufacturing, such as low temperature lubricants or preservatives, or finger prints etc. Example of films are shown in Figure 28

Figure 27: Particulate contamination: Residue from the manufacturing process (Win).

In any industry, the assessment of cleanliness of the machine parts, both before and after the product component undergoes a cleaning process, is done either by using a direct or an indirect method.

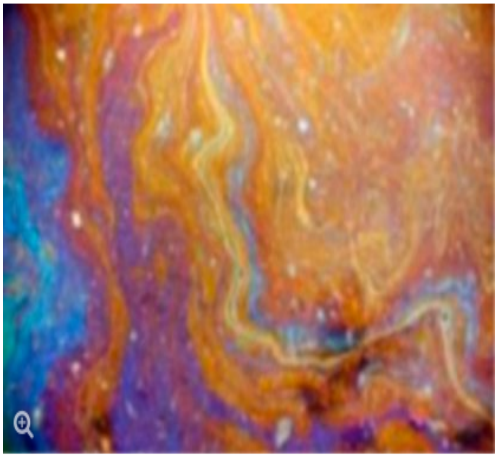


Figure 28: Molecular contamination: Organic and inorganic films during manufacturing (Win).

- A direct method involves the direct examination of the surfaces of the parts without extraction or transfer of particles. This is normally performed with optical or electron microscopy. The advantages of direct analysis involves no sampling losses and no extraction method is required. However, it is not a practical solution when the substrate (component parts) have complex shapes and sizes.

- An indirect method involves extraction or transfer of particles from the substrate to an analytic medium. This is normally done by using a liquid or gas solvent, stripping, or tape lift, followed by evaluation using a microscopic analysis system. An indirect process of component analysis is practical for components with complex geometries and the entire component can be examined, but the particle losses during the extraction process can take place, and higher costs are involved due to the extraction method required, and also very clean and controlled working conditions are required to avoid cross-contamination.

For most of the industries, according to the ISO standard 16232, the quantitative determination of the particulate contamination is done by removal of the particles from the surface using an indirect method such as sonication, pressure rinsing, functional test benches, agitation, and then transferred onto a membrane filter, via filtration of the extraction liquid. The total contamination is then deduced from the analysis.

Identifying particulate contamination, measuring them and helping in their removal automatically is an industrial problem that has been actively pursued since the year 2000. A particle analysis comprises an evaluation of the particle characteristics with respect to number, size and particle type, sometimes also the morphology of the particles. For the analysis of the particle contamination after removal from the production parts, the most popular automated technique is the imaging method.

Automated image analysis using simple optical electron microscopy with traditional analysis techniques like thresholding, color segmentation etc. are not sufficient enough to detect and analyze the particles for this problem. Especially, the metallic particles pose difficulties in the automated evaluation. A metallic particle shows reflecting and non-reflecting areas in optical images and contains corridors of similar brightness as that of the background filter. An automatic analysis system therefore runs the risk of not detecting such particles as a whole, or assigning

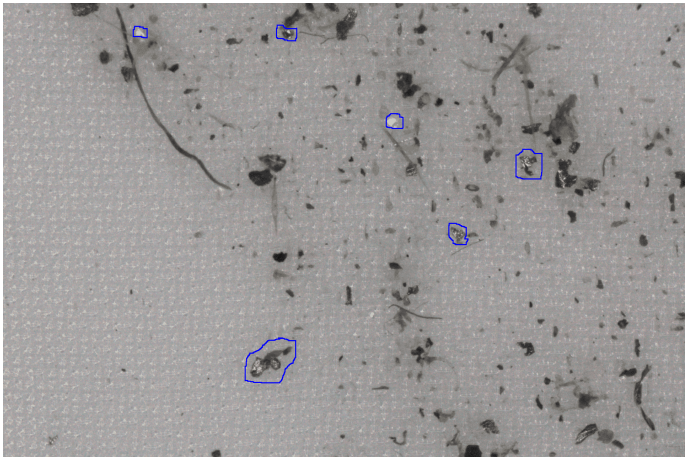


Figure 29: An example image of the contaminants on a wire meshed membrane filter.

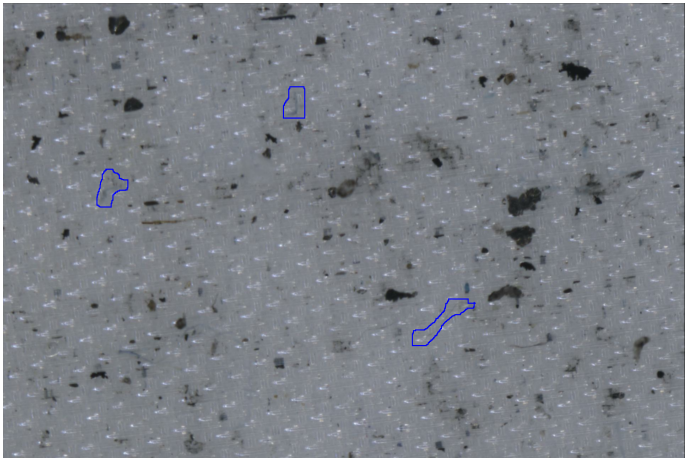


Figure 30: Another example image of the contaminants in the membrane filter with lot more particles.

them as several smaller particles. Sometimes also the very faint particles or stains on the background filters are not detected. Also, dedicated magnification of the camera can limit the cameras field of view and larger

particles might be split between two or more images. Two images of contaminants acquired with a simple optical microscope are given in Figure 29 and Figure 30. Blue lines in the images mark some of the metallic particles, very faint particles or very white particles that are difficult to detect using simple traditional analysis techniques. With the requirement of higher accuracy in contaminant particle detection, the detection ability of traditional object detection algorithms such as threshold segmentation, edge detection, contour detection and level set segmentation are unsatisfactory. Therefore, to fill this gap in research and to get better detection and segmentation results, more complex machine learning algorithms using deep CNNs are proposed for this problem.

At the present market, the best available detection and segmentation systems (Met10) use polarized light for the detection of highly reflective metallic particles and differentiate them from the dark, dull non metallic particles. Usual light sources emit non polarized light that contains oscillations in all directions. By passing through a linear polarizing foil (called as a polarizer) the light has one (linear) oscillation direction. The metal and the non metal particle scatters polarized light in different ways as shown in Figure 31. Light after passing through metal particles remains polarized whereas for non metallic particles, the polarized light is scattered in all possible directions and therefore using a second polarizer-analyzer system, it (Met10) is able to distinguish between metallic and non metallic particles.

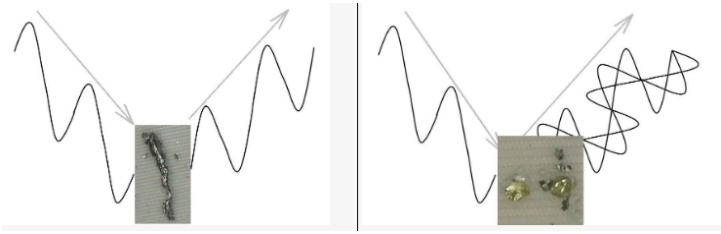


Figure 31: First part of the image shows a metal particle that reflects the incident light, like mirrors such that the incident and reflected light have the same oscillation direction whereas the second part shows a non metal particle that modifies or disturbs the incident light direction (mostly because light can intrude into the material), and therefore light scattered from non-metal particles is no more polarized (Met10).

The detection and contaminant separation capability of this system

is much higher than the ones using optical microscopy, however, the acquisition and analysis of the images using this system is more time consuming. Also the integrated software used in combination with the polarizers and analyzers to analyze the polarized images, are quite sophisticated and the overall system is very expensive. The total time for acquiring and analyzing about 200 images of 2048×2048 pixels is approximately 15 minutes and the cost of the system is approximately $50,000$ USD ($\$50,000$). The main objective in the automobile, aerospace or manufacturing industries is to get a detection system that is precise and accurate and also economical with a high speed of operation (acquisition and analysis to be completed is less than 5 minutes for an entire membrane filter that contains almost $180 - 200$ 2048×2048 images).

Therefore, as a part of this thesis, two different deep learning solutions are proposed for this contaminant separation problem. They are compared against each other for the segmentation quality and for the time required during inference. Each of the methods have its own strengths that can be exploited depending on the requirements of the specific industry. In contaminant separation, speed, accuracy and cost, all the three factors should be kept optimal. Therefore, the use of machine learning seems very ideal. These are the three main challenges that needs to be addressed and bridge the gap while designing any machine learning solution for this problem.

The industries need to measure the structural and the categorical properties of the particles, therefore, two end-to-end segmentation deep neural models are proposed. These models help to segment and classify the particles at the same time. The first approach uses a modified Mask RCNN (HGDG17) (the original framework is described in the Section 4.2 and the adaptations to the contaminants problem are explained in this chapter), an instance segmentation model. The second approach uses a modified UNET (RFB15) (the original work is outlined in Section 4.2 whereas the modifications are detailed in this chapter), a semantic segmentation model followed by image post processing. Each of the two end-to-end solution generates classified object instances at the output. Although both of these frameworks on their own solve two different problems of instance and semantic segmentation but their performance is not compared individually. But the performance of the pre-processing, prediction and the post processing framework (end-to-end solution) together delivers object instances for both of these approaches. Therefore, comparison of the end-to-end solution using Mask RCNN based framework and UNET based framework is quite meaningful in the context

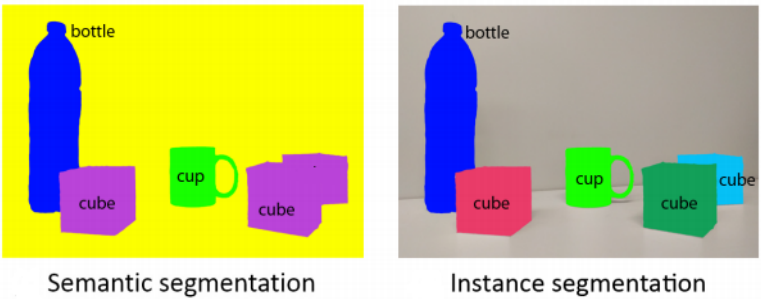


Figure 32: Semantic and Instance segmentation along with classified objects are shown in the example (GOO+17).

of this research work. Both of these solutions bring down the cost of the whole system to approximately $1/5$th of that of the polarized system. Also, the accuracy of the both the solutions are comparable and almost equal in terms of the segmentation and classification quality but the UNET based system is significantly faster than the Mask RCNN based system. The analysis time for 200 images with 4 GeForce-2080Ti GPUs for a Mask RCNN integrated system takes approximately 5 minutes whereas a UNET based system, can analyze an entire membrane filter (about 200 images) in approximately 1.15 minutes using only one GPU.

7.2 Related Works

Many applications need accurate and efficient segmentation mechanisms like autonomous driving (EMGG09; Gei12; COR+16), human-machine interaction (OWL15), image search engines (WWH+14), virtual or augmented reality systems etc. Semantic and instance segmentation are the high-level tasks (defined in Section 4.2) that pave the way towards complete scene understanding. An example to differentiate the two segmentation techniques is shown in Figure 32.

UNET and Mask RCNN models and their upgraded versions have been applied in literature for different applications. Some of them have very promising results and have helped in solving several real world problems. In the next section, some of these works in different fields are reviewed.

7.2.1 Semantic vs Instance segmentation models on natural images

The UNET (RFB15) is inspired by Fully Convolutional Network (FCN) (LSD14) architecture and is shaped like a U having a contracting part to compute features and an expanding part to spatially localize patterns in the image as explained in details in the Section 4.2. UNET is capable of learning from a relatively small training set which gives it a huge competitive advantage over many other semantic segmentation CNNs. UNET architecture in its various forms have been used in different types of segmentation tasks of natural images. One such example is given on the left side of Figure 32 which shows the semantic segmentation results and also tells us about the class of each object. UNET has been very successfully applied to different datasets containing natural images like PASCAL-VOC [1], COCO [2], Cityscapes [3] etc. TernausNet (IS18), a version of UNET with VGG11 encoder pretrained on ImageNet obtained the first place in the Carvana Image Masking Challenge [4] where it effectively segmented car images from the photo studio backgrounds. A linear combination of several UNET networks known as stacked UNETs (SUNETs) (SGDG18) have been developed by Shah et al. which iteratively combines features from different resolution scales. Stacked UNET can leverage the information globalization power of UNETs in a deeper network and is more capable of handling the complexity of natural images than plain UNET. They obtain the state-of-the-art results of natural image segmentation on PASCAL VOC challenge 2018, but they are more dense and computationally intensive.

The MaskRCNN (MRCNN) architecture (HGDG17) is an instance segmentation model as described in Section 4.2 which has a proposal network and deep CNNs to predict the bounding box coordinates, the associated classes of the objects and also the binary masks of the objects. Because of the presence of different modules, MRCNN is a more complex network compared to UNET. The right side of Figure 32 demonstrates an example of instance segmentation using MRCNN. Mask RCNN also obtained the state-of-the-art results in instance segmentation on COCO, PASCAL VOC and Cityscapes segmentation challenges in 2016. Yu et al. (YTRW18) performed human instance segmentations on the Person

[1] www.host.robots.ox.ac.uk/pascal/VOC/

[2] www.cocodataset.org/

[3] www.cityscapes-dataset.com/

[4] www.kaggle.com/c/carvana-image-masking-challenge

dataset (PIC) [5] at ECCV, 2018. In most of these tasks, MRCNN used residual networks or feature pyramid networks as the backbone architecture that are pretrained on ImageNet dataset.

7.2.2 Semantic vs Instance segmentation on medical or pathological images

UNET architecture (RFB15) was first applied on biological microscopy images. It showed remarkable semantic segmentation results on the neural structures. The geometry of the neural structures is somewhat similar to the long fiber contaminants on the filter images present in the images used in this thesis. The similarity in the structure of the particles is one of the motivation for using the UNET network. The UNET architecture was also seen to achieve state-of-the-art results on ISBI cell dataset (RFB15). Chang et al. proposed (CZY$^+$18) 3-D UNET with multiclass focal loss for brain tumor segmentation. Hybrid densely connected UNET (H-DenseUNET), have been proposed by Li et al. (LCQ$^+$17) for liver and tumor segmentation during liver cancers. H-DenseUNET consists of a 2-D DenseUNET for efficiently extracting intra-slice features and a 3-D (CZY$^+$18) counterpart for hierarchically aggregating volumetric contexts for the segmentation purpose. Zhuang et al. (Zhu18) proposed LadderNet, a multibranch convolutional neural network for semantic segmentation, which has more paths of information flow. LadderNet have been used in successfully segmenting lesions from the brains. UNET++ (ZMRSTL18), is another architecture used for medical image segmentation. It is a deeply-supervised encoder-decoder network where the encoder and the decoder sub-networks are connected through a series of nested, dense skip pathways. UNET++ outperformed UNET architecture across multiple medical image segmentation tasks like nodule segmentation in the low-dose CT scans of chest, nuclei segmentation in the microscopy images, liver segmentation in abdominal CT scans, and polyp segmentation in colonoscopy videos. Chen et al. (CZH$^+$18) proposed a bridging architecture between two U-nets where they connect each decoder layer of the first UNET with the corresponding encoder layer of the second UNET, that directly inputs the features of the previous layers into the latter layers. This process reduces the training cost and exhibits a better performance than using an ensemble of two UNETs or a stacked UNET or UNET++. The hybrid and modified UNETs are ar-

[5] www.picdataset.com/

chitecturally more complex than plain UNET and therefore require more inference time in dense prediction of pixels in high resolution images. Alom et al. (AHY+18) proposed a slightly different network combining UNET and recurrent residual networks to obtain a CNN model called as R2U-Net that gave superior segmentation results than plain UNET or residual UNET on blood vessel segmentation in retina images, skin cancer segmentation, and lung lesion segmentation problems.

MRCNN has not been used in the medical community extensively but still applied in some of the works outlined here. Johnson et al. (Joh18) and Xie et al. (XLZS18) have used Mask RCNN for automatic segmentation of the overlapping nuclei in a wide range of microscopic histopathological images, for a variety of cells acquired under different conditions. Xie et al. (XSNT18) also used an improved MRCNN in breast tumor segmentation and classification of ultrasound images. In the works of Zhao et al. (ZYZ+18), MaskRCNN have been used for doing 3D biomedical image segmentation and in the works of Liu et al. (LDD+18), MRCNN helped in segmenting lung nodules from CT images. Liu et al. (LL18) used an improved RPN network where ROI align mechanism is based on bilinear interpolation for segmentation on ultrasound images.

7.2.3 Semantic vs Instance segmentation on other problem specific images

UNET have been successfully applied in many other real world problems especially in satellite imaging as discussed here. Zhang et al. (ZLW17) proposed a deep residual UNET for road extraction from aerial images in the field of remote sensing image analysis. It combines the strengths of residual learning and UNET. In DSTL Satellite Imagery Feature Detection Challenge [6], and Mapping Challenge [7], Ternaus-$v4$ (IS18), a variant of UNET, detects buildings from satellite images and is one of the top performer. The DSTL dataset has scarce labeled examples and UNET performs well even in scarce data settings. Li et al. (LLY+18) designed a network called DeepUNET that helped in pixel level sea-land segmentation from remote satellite images.

In the works of Jaspreet et al. (SS18), Mask RCNN is used for automatic detection and classification of damage in roads that helped in maintenance and autonomous driving. Many remote sensing applica-

[6] www.kaggle.com/c/dstl-satellite-imagery-feature-detection
[7] www.crowdai.org/challenges/mapping-challenge

tions also require instance segmentation of roads and buildings in satellite images and MRCNN has been the top performer in many competitions using satellite images like DSTL and Mapping Challenge.

Therefore, all these varied applications suggest the immense power and flexibility of both of these architectures in segmentation of different types of images.

7.3 Experimental Methods and Evaluation

The main aim of this experimental analysis section is to develop an end-to-end deep learning solution to segment individual contaminant particles for cleanliness analysis that can beat the existing solutions in terms of accuracy and speed. The two solutions proposed are:

- Instance segmentation Mask RCNN model with refined regional proposal network to detect and segment the non-overlapping as well as the overlapping particles as separate instances through the process of instantiation. This model is then connected to a post processing module that implements panoptic segmentation which implies that every pixel of the image can belong to one and only one instance or object of that image. Since, Mask RCNN already outputs individual instances of objects, less amount of post processing is required in this approach while developing the end-to-end solution for contaminant separation.

- Semantic segmentation UNET based model with refined cost function for faster convergence and better generalization capability. This model does not help in the separation of the overlapping particles. Hence to separate the overlapping particles as different instances or objects, this model is cascaded with traditional computer vision modules like connected component analysis and watershed transform. In this framework, several domain specific post processing approaches are used to build the end-to-end solution of contaminant separation.

The description of the contaminants dataset, the modified evaluation metrics and the proposed solutions with Mask RCNN and UNET are described in the following sections.

7.3.1 Dataset

The contaminant image is created by raster scanning the membrane filters mounted on a casket in an optical microscope. These membrane filter samples have been collected from different industrial setups in the automobile sector. On scanning each filter, about $180 - 200$ images, each of 2048×2048 pixels are obtained. The total contaminants dataset used for this research work is a collection of 25 membrane filter images collected under varying lightning conditions resulting in a total of 5000 images, each of size 2048×2048 pixels. The images also vary greatly in magnification, and the type of background filter used. Some of the background filters are 5 micron wired mesh filters whereas others are $10/20$ micron wired meshes. The shapes and sizes of the particles present on each of the filter also varies greatly. The number of particles present in each image varies randomly from about 10 particles to approximately 2000 particles. On an average there are 340 particles per image. Some examples of the images have been shown earlier in Figure 29 and Figure 30. The particles in the dataset are divided into three classes :

- Metal
- Non-metal
- Fiber

The dataset is annotated by humans using GNU Image Manipulation Program (GIMP) as the annotation tool. Since, the objective of this problem is to accurately measure the size and shape of the particles, each image is annotated with a instance level segmentation mask. The particles in the segmentation masks of the ground truth annotations are represented in three different colors to represent the three classes. The metal class objects have red mask, the non metal class objects are annotated with blue mask and the fiber class objects are represented with green mask. Figure 29 is annotated in Figure 33.

The images are divided into 3500 training images, 500 validation images and 1000 test images. This is a relatively small training set for training an instance based model like Mask RCNN. Each image is split into 16 equal parts for memory efficiency during training. Different data augmentation techniques applied to input splits generate 168K images, each of 512×512 pixels. The imgaug library [8] is used for creating augmented

[8] www.imgaug.readthedocs.io/en/latest/

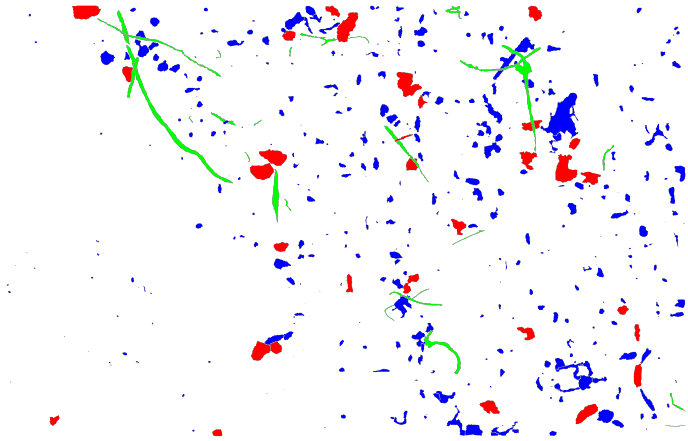


Figure 33: Instance segmentation mask and classified objects based on color of the image in Figure 29.

data. For example the different augmentations used are horizontal and vertical flipping, rotating, scaling, adding gaussian noise etc. Since this is a segmentation task, it is also necessary to perform the same augmentation on the mask as on the original image while feeding the input into any segmentation framework.

7.3.2 Evaluation Metrics

A new and meaningful set of evaluation metrics are defined for this instance segmentation task that is based on the concepts of Intersection over Union (IOU), True positives, False positives, True negatives, False negatives and F1 score.

- *Intersection over Union :* It is also known as the Jaccard index, which essentially is a method to quantify the percentage of overlap between the target/ground truth mask and the prediction mask. It is the metric used in semantic segmentation. The intersection of two masks A and B is given as $(A \cap B)$, comprised of the pixels found in both the masks, whereas the union $(A \cup B)$ is the sum of all pixels found either in the prediction or in the target mask. The IOU thresholds are ranged from 0.05 to 0.95 and the intersection over union mask is computed for each of the threshold value in

this research work.

$$IOU(A, B) = \frac{(A \cap B)}{(A \cup B)}$$

- *True Positives :* It is observed when a prediction-ground truth instance pair in the masks have an IOU score that exceeds a predefined IOU threshold. Total number of prediction instances that match the ground truth instances are called the true positives (TP).

- *False Positives :* It indicates when a predicted instance mask has no associated ground truth instance mask. Total number of false predicted instances that match no ground truth instances are called as false positives (FP).

- *False Negatives :* It indicates a ground truth instance mask has no associated predicted instance mask. Total number of ground truth instances that are missed and no corresponding prediction instances are present are referred to as false negatives (FN).

- *Segmentation Precision (SP) :* It is defined for instance segmentation and is effectively described as the number of correct instance segmentations relative to the total number of segmentations. Therefore, it is formulated as

$$SP = \frac{TP}{TP + FP}$$

- *Segmentation Recall (SR) :* It is effectively described as the completeness of the correct segmentations relative to the ground truth instances. Therefore, formulated as

$$SR = \frac{TP}{TP + FN}$$

- *F1 Score :* It is defined for the segmentation quality and it is the harmonic mean of the segmentation precision and segmentation recall for a given IOU threshold. Therefore, it is formulated as

$$F1 = \frac{2 \times SP \times SR}{SP + SR}$$

- *Classification Quotient (CQ) :* The segmentation precision is class agnostic but the classification quotient is a factor that takes into consideration the class performance. In case of instance segmentation the *True Precision (TPR)* is defined as:

$$TPR = \frac{TP_c}{TP + FP}$$

where TP_c is the number of true positives for the given class. Therefore, $TruePrecision$ of the instance segmentation model is the product of segmentation precision and the classification quotient.

$$TPR = SP \times CQ$$

. The classification quotient is defined as the number of true positives for a given class relative to the total number of true positives.

$$CQ = \frac{TP_c}{TP}$$

As mentioned earlier, the segmentation precision, segmentation recall, F1 score and classification quotient are all calculated for different IOU thresholds ranging from 0.05 to 0.95. With the increase in the IOU threshold during the inference time, the number of instances correctly predicted decreases. After comparing the segmentation outputs at different IOU thresholds, the IOU threshold 0.7 is considered to be the one which is most acceptable according to the required industry standard. The segmentation precision, segmentation recall and F1 score are made class agnostic because, sometimes, even when the particles are not correctly classified but are well segmented, it is good enough for the industries to get an estimation of the size and morphology of the particle although the category of the particle is not correctly identified. Since, the size of the contaminants poses a more dangerous risk than the class of the particles, as bigger sized particles result in more wear and tear of the machine parts, therefore, the solutions are pushed towards getting better segmented instances.

7.3.3 Modified Mask RCNN based end-to-end solution with improved RPN

Description

Mask RCNN is implemented based on its public implementation in Matterport [9] and adapted to the contaminant separation problem. The Mask RCNN is an intuitive extension of the Faster RCNN architecture with improved properties and additional mask head for instance segmentation (as described in Section 4.2) and the increase in computational overhead due to these modifications is very small.

The modified Mask RCNN used in this research work have some differences from the original works (HGDG17) as listed below:

1. in the backbone feature extraction network used,

2. in the selection criteria of anchors similar to the works of Liu et al. (LL18), therefore different types of region proposals are generated from the Region Proposal Network (RPN),

3. and also an end-to-end panoptic post processing in the mask head for removing the overlapping predictions of the instances.

All these modules changed in the original framework to suit the needs of this problem are reusable and scalable to other similar segmentation problems of road detection in aerial images, tomour segmentation in medical images etc. For the feature extractor in the modified MRCNN, the effective Feature Pyramid Network (FPN) (LDG$^+$16) (explained in Section 4.2) is used in combination with ResNeXt (XGD$^+$16) with 50 layers (architecture explained in Section 4.2). FPN have a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Whereas, ResNeXt is an upgraded version of the Resnet that increases the accuracy without increasing the complexity of the parameters and also reduces the number of hyperparameters. Therefore, using a ResNeXt-50-FPN backbone for feature extraction in Mask RCNN gives excellent gains in both accuracy and speed. ResNeXt-101 have also been experimented but there is no gain in performance in terms of segmentation quality on the contaminants problem probably because of the small dataset size in comparison to the network depth.

[9] `www.github.com/matterport/Mask_RCNN`

For better segmentation of the image in the mask head, the ROI selection is an important task. The ROIAlign technique removes the harsh quantization of ROIPool (as explained in the works of He et al. (HGDG17)), and henceforth, properly aligns the extracted features with the input. But sometimes even the ROIAlign is not enough for making the best ROI. The RPN layer selects the ROI region based on the IOU overlap between each anchor and the ground truth bounding box. For each anchor, the RPN generates two outputs :

- The anchor class which is either a foreground or a background class. The foreground class is also called a positive anchor and it implies that there is likely an object inside the anchor. An anchor is a positive anchor only when it has an IOU overlap of 0.5 (HGDG17) or greater with the ground truth box.
- The bounding box refinement is done by the RPN network after a positive anchor is selected. A positive anchor although has a high IOU overlap with the ground truth box but yet it might not be centered perfectly over the object. So the RPN estimates a delta value ($\%$ change in x, y, width, height) to refine the anchor box so that it fits the target object better. Using these RPN predictions, the top anchors that are likely to contain the objects are selected and their locations and sizes are refined. If several anchors overlap, non max suppression is applied to select the good anchors and the number of good anchors selected depends on the non max suppression threshold.

The refined proposals or regions of interest are then passed on to the next stage by the region proposal network. However, this selection criteria sometimes overlooks some of the deficiencies that might arise. Suppose if there are two different anchors that overlap the ground truth box with the same IOU, which anchor is more suitable for the image segmentation task is difficult to decide. Sometimes one anchor can be slightly smaller and tightly adhered to the target object, while the other can be slightly bigger than the real object but both of them overlap the gound truth box with the exact same IOU. Even though the RPN network has a boundary regression correction, there is no guarantee that the modified bounding box will include the target object perfectly. The best possible anchor would actually be the one that covers the entire target object and also desirably a small amount of background area (that is the second one which is slightly bigger than the target object) information, rather than

tightly adhering to the target object. Therefore, to select the best possible anchor, a new anchor selection criteria is implemented. This new selection criteria is based on the works of Liu et al. (LL18) :

1. An anchor is marked as positive if the IOU overlap with the ground truth box is greater than 0.7, and a negative sample if the overlap is less than 0.3 which means it does not contain the target object at all. Anchors in between (i.e. cover an object by IOU ≥ 0.3 but < 0.7) are considered neutral and excluded from training.

2. In order to ensure that the extracted anchor does not lose the information of the target object, all the positive anchors are sorted by the size of η where η represents the ratio of the overlap between anchor and ground truth box in the ground truth, given as:

$$\eta = \frac{Area(Anchor) \cap Area(Groundtruth)}{Area(Groundtruth)}$$

.

3. For all anchors with $\eta = 1$, they are sorted by their IOU overlap values and a non max suppression is performed to select the best anchors and then the boundaries of the anchor are refined using RPN predictions.

The new anchor selection criteria has been seen to improve the regression parameters of the selected candidate boxes or ROIs, and therefore better segmentation performances are obtained on the instances. The experimental results demonstrate this improvement empirically.

The mask head of the modified MRCNN relies on a fully convolutional network (FCN) (LSD14) like in the original works on MRCNN (HGDG17). For every selected ROI, a $m \times m$ mask is predicted that allows to maintain the explicit $m \times m$ spatial layout of the object and hence better instance segmentations are performed. The mask loss L_{mask} (as defined in the original works (HGDG17)) is defined only on positive ROIs. Also, the mask and class predictions are decoupled, thus predicting a binary mask for each class independently, without having competition among the classes, and then relying on the networks ROI classification branch to predict the category or the class. This enables parallel processing during prediction of the class and the bounding box, with the binary mask for each ROI.

Training and Inference

Images of size 2048×2048 pixels are split into 16 sub-images of 512×512 pixels during training. Each split of the image is then augmented using the data augmentation techniques in the imgaug library like random horizontal or vertical flips, random rotations in the range from 90 to -90 degrees and also random scaling in the range of 0.5 to 2.0. The input is normalized, with a mean pixel value of $[129.5, 129.47, 129.56]$ for the R, G and B channels respectively.

The backbone architecture for feature map computation is ResNeXt-50-FPN, that uses the pretrained ImageNet weights during training. This network shares features between the RPN and also with the bounding box regression and the classification head. Images are trained in a mini-batch of 8 images on 4, 2080Ti GPUs (2 images per GPU). Many other backbone networks have also been tested but ResNeXt-50-FPN gives the best performance during cross-validation for this problem.

The RPN anchors span 5 scales (chosen according to the range of sizes of the contaminant particles) and 5 aspect ratios (chosen as $0.2, 0.5, 1, 2, 5$ to generate anchors with different aspect ratios) in this framework, similar to the works of Lin et al. (LDG$^+$16). Mostly small anchor scales $(4, 8, 32, 80)$ are chosen because 80% of the particles in the images are less than 80 pixels in size. During training, each image have 1000 sampled ROIs, with a ratio of 0.67 positive to negative ROIs (Gir15). The non max suppression (NMS) threshold during training is reduced to 0.4 from 0.5 (as in the original works) to generate more proposals because in many images the distribution of contaminants is very dense and they can occur anywhere in the image. More proposals help to represent the image in a better way having minimal chance of missing a region containing a particle. The model is then trained for 100 epochs where each epoch consists of 170K iterations. An Adam optimizer is used with the following learning rate schedule as listed below:

- 25 epochs with learning rate 0.003,
- 25 epochs with learning rate 0.0001,
- 50 epochs with learning rate 0.00003.

A weight decay of 0.0001 and a momentum of 0.9 are used. The mask loss (L_{mask} defined in Equation 4.3) in the modified Mask RCNN is weighted 4 times more than the RPN (L_{R-cls} and L_{R-box}) and the classification losses (L_{cls} and L_{box}) and it is seen to increase the segmentation

precision by 3.4%. The detailed configuration file of the modified MRCNN is provided in the Table 92 in Section B.0.1. It takes approximately ~ 60 hours for training the model.

The original Matterport implementation of Mask RCNN uses only one image during validation and this leads to problems in tuning the hyperparameters well. Therefore, in this problem, the model uses 500 images to do cross-validation and selection of several parameters like learning rate, feature extraction network, weights assigned to the mask loss etc. An inference is performed using the modified MRCNN model on image splits of 512×512 pixels. Then, 16 splits of each image are reconnected to generate the entire 2048×2048 image in the post processing stage. During inference, this model takes ~ 400 ms to output an instance segmentation mask of size 512×512 pixels and therefore a entire image is predicted in 1.6 sec using 4 GPUs. Test time augmentations are also performed consisting of horizontal and vertical flips, rotations and scaling and this technique is seen to improve the segmentation precision by 5%. The NMS detection threshold in the modified MRCNN is reduced to 0.35 from 0.5 (in the original implementation) to help in better detection of very small instance boundaries.

A post processing module is also placed after the mask head and used only during inference to perform certain post processing operations in an end-to-end fashion. The different post processing operations performed.

1. Closing small holes inside the masks using morphological operations (dilation followed by erosion - closing operation).

2. When the mask head of Mask RCNN does a pixel to pixel prediction to generate the mask for each selected ROI, it does not take into account the panoptic metric (KHG+18) which says that one pixel in the image can only belong to one instance. Therefore, in Mask RCNN often a single pixel is assigned to two or more object instances. Thus a post processing of all the predicted masks are done to remove all the overlapping pixels from the last added mask.

3. All the instances that are smaller in size than 10 pixels are removed because in the contaminants problem an objects smaller than this threshold cannot be considered as a contaminant that can pose risk in the production process.

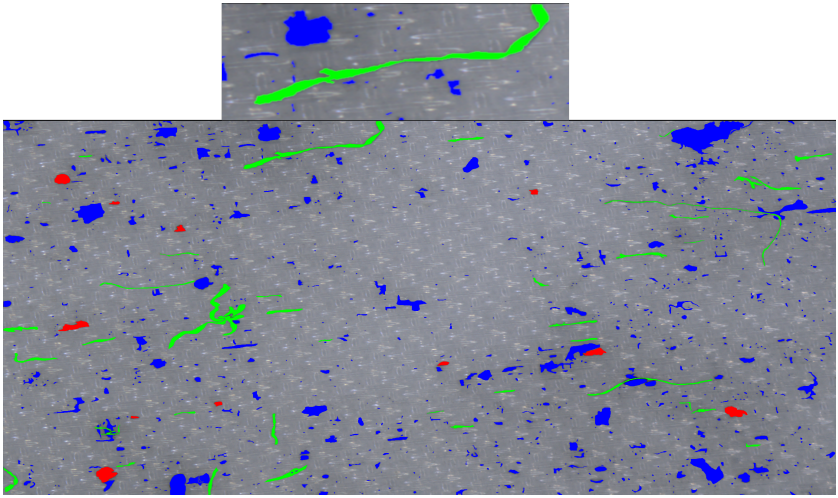


Figure 34: The ground truth annotation where red objects represent metallic particles, blue represents the non-metallic particles and green represents fiber particles. At the top, the zoomed version of a part of the image is shown.

Results

The results are shown by overlaying the mask (annotation or prediction) on the image. The modified Mask RCNN predictions on a image is given in Figure 35 and its corresponding ground truth annotation is given in Figure 34. Comparing the prediction with the ground truth annotation, it is seen that MRCNN segments the small and medium sized particles quite well. MRCNN also behaves as a good classifier. Although the segmentation and classification quality is good and precise in most of the test images but sometimes with very dense images like in Figure 34, there are few errors observed. The small rectangle at the top of each figure represents a small part of the image in its zoomed version. In Figure 35, the black lines highlight the errors in segmentation or classification of the particles. The different types of errors encountered and the possible reasons are listed below:

- Some instances especially the long fiber as seen in the zoomed window is splitted into four smaller instances as compared to its ground truth where its a single instance (Figure 34). This is one of

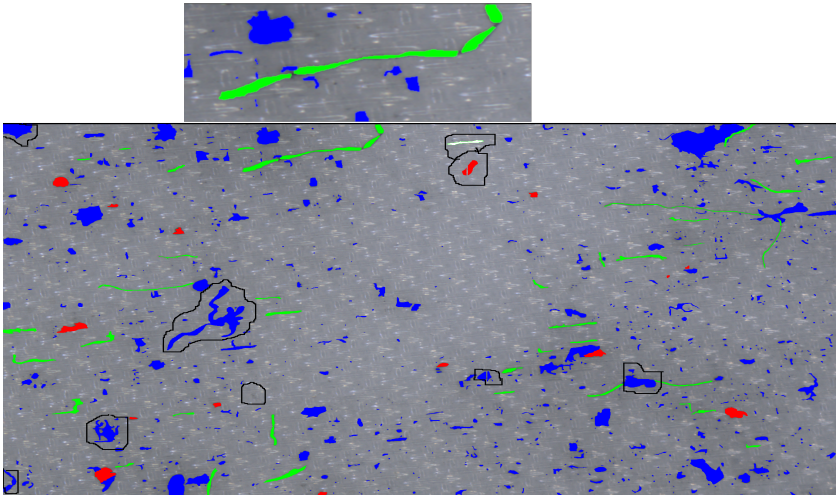


Figure 35: This is a predicted mask where black markers represent the errors in segmentation and classification.

the challenge faced by Mask RCNN in terms of the segmentation quality. Thin and long objects are often not well detected because of the downsampling errors that occur at different stages in the MRCNN framework.

- Sometimes very light and faint colored instances are not detected. Although, the modified RPN doesnot tightly adhere to the instance when proposing the ROI region, but because of the nature of the image, when ROI is proposed for an object which is very similar to the background color, it is classified as a negative sample during ROI selection.

- Sometimes a very thin line is observed dividing a single instance into two separate instances. An object gets divided into two or more parts because of the image splitting during inference time. A large particle extending over a number of 512×512 splits is predicted as number of separate instances, however this can be solved easily in the future works by using overlapping splits or some extra post processing.

- Sometimes the objects are misclassified. This happens because of

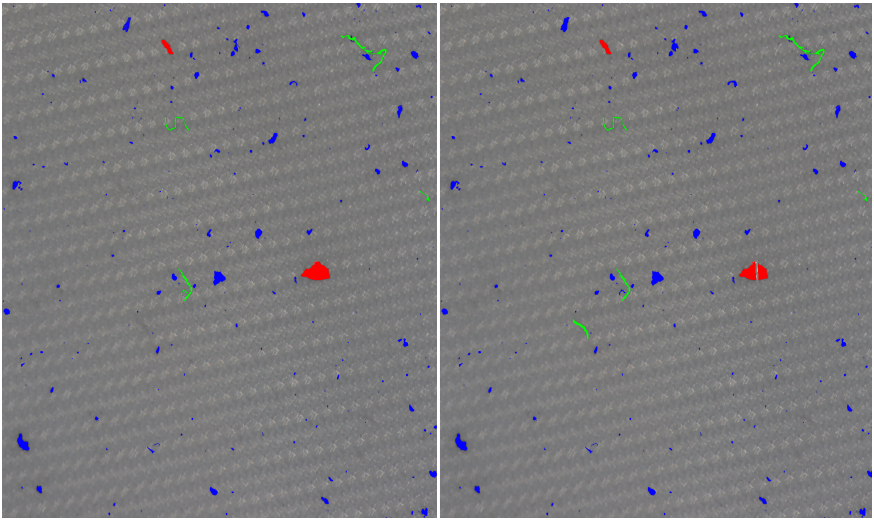


Figure 36: The groundtruth annotation is on the left side and the predicted mask is on the right side of the image.

the huge class imbalance problem that is present in the images. The number of metallic and fiber particles are much less in proportion to the number of non metallic particles. The class imbalance problem is not tackled rigorously because the observed average classification error was low and limited to only 4% during inference.

Another example of segmentation of a relatively non dense image with fewer number of particles is shown in Figure 36. The larger red instance in the image is splitted into two instances in the predicted mask.

The results of segmentation and classification are tabulated in Table 52. The segmentation precision, segmentation recall and F1 score are all calculated for a specific IOU threshold of 0.7. However, the true precision is obtained as the product of segmentation precision and classification quotient. The original version of MRCNN is compared with the modified MRCNN with a new improvement added every time to the modified MRCNN framework. In the original MRCNN, ResNeXt-101-FPN backbone network is used and the old RPN selection criteria is used. In this framework, mask loss has unity weight and its contribution to the total loss is the same as the contribution of the RPN and classification

Table 51: Comparison of original MRCNN (abbreviated as Or. MRCNN) framework with the Modified MRCNN (abbreviated as Mod. MRCNN). SP, SR, CQ and F1 represents the segmentation precision, recall, classification quotient and F1-score respectively. Wghtd. is the abbreviation for weighted. The bold numbers represent the highest segmentation precision and the F1-score for the Modified MRCNN.

Model	Techniques	SP	SR	CQ	F1
Or. MRCNN	ResNeXt-101-FPN	76.55	77.24	90.10	76.89
Mod. MRCNN	New RPN ResNeXt-101-FPN	80.46	82.11	92.09	81.28
Mod. MRCNN	New RPN ResNeXt-50-FPN	80.31	82.46	91.85	81.37
Mod. MRCNN	New RPN ResNeXt-50-FPN Wghtd. Mask Loss	83.71	85.33	93.56	84.51
Mod. MRCNN	New RPN ResNeXt-50-FPN Wghtd. Mask Loss Test Augmentation	88.10	90.88	95.91	89.47
Mod. MRCNN	New RPN ResNeXt-50-FPN Wghtd. Mask Loss Test Augmentation Post Processing	**91.41**	93.08	96.09	**92.24**

losses. The segmentation precision and recall on 1000 test images using the original MRCNN is about 76.55 and 77.24 respectively. The calculated true precision (considering segmentation quality and correct classifications) is 68.97. However, in the modified MRCNN, the improved selection criteria of the anchors in RPN network is used that obtains better precision and recall values than its original counterpart. It is also observed that the segmentation quality and the classification quotient of ResNeXt-101-FPN is slightly lower than ResNeXt-50-FPN because of the overfitting problem. Also, 101-layered ResNeXt is much slower during inference than 50-layered ResNeXt, therefore ResNeXt-50-FPN is incorporated in the final version of the modified MRCNN. There are also other modifications added to the modified MRCNN to improve the segmentation quality and the classification quotient like weighing the mask loss higher than the RPN and classification losses, adding test time data aug-

mentations and also post processing the segmented masks. The best segmentation precision and segmentation recall achieved with these modifications in the adapted MRCNN framework is 91.41 and 93.08 respectively. The calculated true precision in this case is 87.84 which is almost 18.9% higher in performance than the original MRCNN framework. This confirms the significance of each theoretical modification added to the framework to make it more suitable for this problem being able to meet the different challenges posed by the problem. The performance gain suggests empirically that each of these techniques make the Mask RCNN framework more powerful and robust.

7.3.4 Modified UNET based end-to-end solution with Watershed Transform

The UNET based solution is designed to deal with the contaminant separation problem not as an instance segmentation task but as a multi-class semantic segmentation task. Although this treats the problem in a different way, but the desired output still remains the same. It is required to obtain object instances through an end-to-end solution and therefore, this semantic segmentation UNET framework is cascaded with heavy post processing methodologies to build an instance segmentation output that can be compared in terms of speed and quality of segmentation with the MRCNN based end-to-end framework design. In comparison to the complex Mask RCNN based framework which is a two stage detector having a region proposal network and a mask segmentation network, the UNET framework is much easier to train and optimize. Also UNET is specialized to work even with small datasets as it does not require a lot of training examples to perform well. The main objective of this approach is to build a faster system with a simplified network that can be better than the existing traditional solutions in terms of accuracy and segmentation quality and on the other hand the solution should consist of a simpler framework than Mask RCNN.

Since the UNET based framework treats the contaminant separation problem as a multi-class semantic segmentation problem, the output of the segmentation will segment two or more overlapping particles of the same class as a single particle. As the industries need to measure the size and the morphology of each single particle, therefore, a module performing connected component analysis and watershed transform is loosely cascaded with the modified UNET framework. The objective of this post

processing module is to separate the overlapping objects into different objects so as to get the true measure of the size and structure of the particle. The connected component labeling helps to provide the meaning of instance or object to the semantic segmentation output of modified UNET. All the 8-way neighboring pixels sharing the same label in the semantic segmentation output, are treated as the part of the same instance or object. Watershed transform is also a labeling algorithm that works with markers that help in defining the boundary of each labeled region. While training modified UNET, the borders of the objects are also trained separately and the border outputs obtained on the test set act as markers for the watershed transform during post processing of the segmentation output from modified UNET. With the help of these markers, watershed transform, a flooding technique for image segmentation helps to divide the overlapping objects from a single instance to two or more instances.

Description

Vanilla UNET or original UNET implemented for histopathological image segmentation (RFB15) is a semantic segmentation network that predicts a semantic mask which segments many overlapping instances or closely spaced instances as a single connected component. In the original UNET implementation, a weighted softmax cross entropy loss has been used to train the full masks where the weights are determined from a pre-computed weight map. The basic idea of the pre-computed weight map is to assign weights to each class, where the foreground classes, the touching borders between the closely spaced objects and the background all belong to different classes and weighed separately according to the Equation 4.4 in Section 4.2. The weight map helps to weight the borders of close instances more thus forcing the network towards learning the gaps between the close instances. In this implementation, the borders are not used during training to make separation between the predicted instances. It is assumed that the network will learn to separate the closely spaced instances well using the weight map but it is necessary to learn the borders and predict the borders during inference to use them at the post processing stage to separate the overlapping instances. Therefore, the adapted and modified UNET solution used in this research work converts the ground truth masks into multi-class targets. It gives two masks at the output: one that predicts the output classes and the other predicts the borders. At the output of the network, the modified UNET has 2 classifier layers:

- The first classifier has 3 outputs, one each for the two classes, metallic, non-metallic (the fiber class is merged with the non metallic class) and the third one is the background class.

- The second classifier predicts the borders (in the ground truth labels the borders are obtained as contours of the instances) and the non borders (which can be objects of the two foreground classes or the background other than the border pixels).

A pre-computed weight map is also used in this segmentation task which is computed based on the works of Ronneberger et al. (RFB15) (described in Section 4.2).

The encoder used in the modified UNET is Wide Resnet-40 (ZK16). The modified UNET uses transposed convolution operations for up sampling in its decoder layers. Transposed convolution has proved to be empirically better than nearest neighbor and bilinear interpolation techniques for contaminant separation.

The loss function used in the modified UNET is a combined pixel wise softmax which consists of a three class weighted cross entropy loss and a binary cross entropy loss along with a soft Jaccard loss (ISBS18). The binary cross entropy loss and the weighted cross entropy loss are the pixel-level classification losses denoted as H_1 and H_2 that are independently applied to each output classifier layer. The weights of the H_1 loss is obtained from the precomputed weight map. The Jaccard loss is denoted as J_1 and is based on the Jaccard index (described in Section B.0.2) that measures the similarity between the pixels of the ground truth mask and the predicted mask of each class using the Equation 7.1.

$$J_1 = \frac{1}{n}\sum_{c=1}^{2} w_c \sum_{i=1}^{n} \Big(\frac{y_i^c \hat{y}_i^c}{y_i^c + \hat{y}_i^c - y_i^c \hat{y}_i^c} \Big) \tag{7.1}$$

where y_i^c and $\hat{y}_i^c$ are the binary value (label) and the corresponding predicted probability for the pixel i of the class c. The value of w_c values are chosen through cross validation based on the class frequencies of each class. The two classes contributing to this loss are the metallic and the non-metallic classes. Jaccard loss possibly penalizes pixels that violate the localization property in the masks. This hypothesis is based on the empirical results obtained by the addition of this loss during optimization.

The final loss function is obtained by combining the classification losses and the segmentation loss as in Equation 7.2.

$$L = \alpha H + (1 - \alpha)(1 - J_1) \tag{7.2}$$

where $H = H_1 + H_2$, the value α is chosen through cross-validation in between the range from 0.6 to 0.9 and 0.7 is seen to be the best for the experiments on contaminants.

Training and Inference

During pre-processing, a set of two class ground truth annotations are created automatically from the given three class ground truth masks (originally used in Mask RCNN training) such that the fiber class is merged with the non-metallic class. It is known from the industry standards that fiber objects are always non metallic. Initially, the fiber particles are in the green channel of the ground truth masks, therefore after merging the blue and the green channel, each ground truth mask contains either metallic or non-metallic objects. Therefore any pixel in the ground truth annotation can have a value $(255, 0, 0)$ indicating a red pixel, or $(0, 0, 255)$ indicating a blue pixel or $(0, 0, 0)$ which indicates a black or a background pixel. This set of ground truth labels are used to train the softmax three class weighted cross entropy loss in the modified UNET. Another set of ground truth labels are extracted from the three class ground truth instances and each of this is a binary label that indicates a border (contours of the individual instances) or no border (background) for training the binary cross entropy loss.

The pre-computed weight map w (as in Equation 4.4 as described in Section 4.2) is multiplied with the three class weighted cross entropy loss to generate the H_1 classification loss. The first weight term in the weight map w for each class (w_c) is assigned from low to high value in the order: background class ($w_1 = 0.2$), foreground instance class (metallic or non-metallic $w_2 = 0.3$) and the borders between the closest objects or the touching objects ($w_3 = 0.5$). The weight bias w_0 is equal to 10 and sigma is 5. The weight map computation takes about 1.5 to 5.9 seconds for each 2048×2048 ground truth mask depending on the number of foreground instances and closely spaced borders in each mask.

Images of size 2048×2048 pixels are split into 16 tiles each of 512×512 pixels during training. Each split of the image undergoes heavy data augmentation like piecewise affine transforms, elastic transforms, random scaling, rotating in the range from -90 to 90 and flipping. This helped the network to learn invariance to such deformations without the

need to have these transformations in the actual images.

The encoder architecture Wide Resnet-40 is pretrained on ImageNet weights. Several other deep architectures are also experimented with like Resnet-50, Resnet-101, Inception Resnet-$v2$, Wide Resnet-38 but Wide Resnet-40 is the best performing encoder architecture. The results are compared in the next section for the different encoder architectures. Modified UNET is trained using an Adam optimizer with a learning rate of 0.0002. The training is done for 1000 epochs. During training, for the initial 2 epochs, the encoder weights are frozen and only the weights in the decoder are trained. This helps in setting meaningful weights in the decoder in a very efficient way. After 2 epochs, all the weights are unfreezed and the network is trained in an end-to-end fashion. Images are trained in a mini-batch of 4 images on a single 2080Ti GPU.

The segmentation or the Jaccard loss is weighed (0.3) lower than the classification loss (0.7) and this is found to be better than weighing them equally.

An inference is performed using the modified UNET model on image splits of 512×512 pixels. Test time augmentations are also performed consisting of horizontal and vertical flips, rotations and scaling and this improves the segmentation precision by 2%. Each of the 16, 512×512 splits are now reconnected to generate a 2048×2048 prediction map. The assignment of label to every pixel is decided during inference to build the final prediction map that should resemble the two class ground truth mask. The border pixel information will be used to separate touching objects or overlapping objects. The first output softmax classifier predicts each pixel as either a red (metallic), blue (non-metallic) or black (background) pixel. To resolve the class label, an argmax is applied to the three outputs and the class label is assigned to that pixel. For example, if the output of the classifier for the pixel i is $(0.1, 0.7, 0.2)$, the pixel is blue or a non-metallic pixel. Similarly, an argmax is applied to the two outputs of the second binary classifier to assign every pixel as the probability of being a border pixel or a non-border pixel. Therefore, two predicted pixel maps for each image are obtained. For the next stage of inference, a higher priority is given to the border- non border pixel map, than to the object-class pixel map. If a pixel in the border pixel map is not a border pixel, then the pixel in the final prediction map is labeled from the object-class pixel map as either a metallic, non-metallic or background pixel. However, if a pixel is present both in the border pixel map and in the object-class pixel map, it is always considered as the border pixel (assigning a higher priority to the borders), and hence it

is labeled as a green pixel denoting a border pixel in the final prediction label. This sort of priority based assignment of labels to the pixels in the prediction map helps to separate some of the closely spaced objects or overlapping objects. After the decision of class labels are made for each pixel, a 2048×2048 RGB mask is generated. A post processing module is then attached to the prediction module that combines watershed transform and some other post-processing techniques as listed below and further improves the segmentation quality of closely spaced or overlapping objects. The post processing is applied to the newly formed RGB mask (where the red channel denotes the metallic labels, blue channel contains non metallic labels and the green channel represents the border).

1. The green channel containing the border values is converted back to border-no-border binary mask and used as the markers of the watershed transform. The RGB (composed of the non zero values in the red and blue channel) mask with the borders removed is also converted into a binary mask and this is the other input to the watershed algorithm. Now using the border values as the markers or the seeds of the watershed transform, the touching or overlapping objects are further separated into individual instances. Then the output binary mask is multiplied along the channels of the input RGB mask to obtain the new refined RGB mask containing the metallic and non metallic instances well separated.

2. The fiber objects are a part of the non metallic class in this new RGB mask. A connected component analysis helps to extract the individual instances in the predicted mask and also provide statistics like the bounding box and area of each particle. A non metallic particle is considered to be a fiber if the extent ratio (which is defined as the ratio of the particle area relative to the total bounding box area $\frac{P_{area}}{BB_{area}}$) of the particle is less than 0.2.

3. Finally all the instances that are smaller in size than 10 pixels are removed from the predicted mask. As already mentioned in the MRCNN analysis that objects smaller than this threshold cannot be considered as a contaminant particle that can pose risk in the production process.

Thus an end-to-end segmentation model is designed by combining smart techniques like a multi-class semantic segmentation framework with multiple classifier layers targeting the objects and the border of

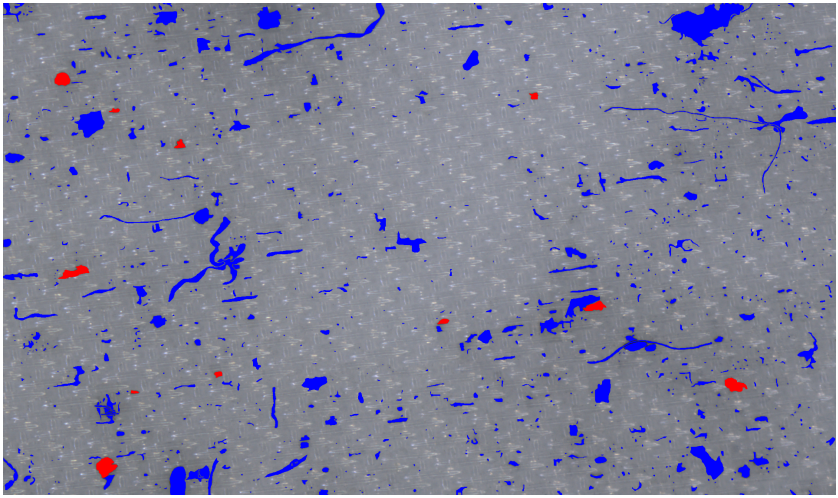


Figure 37: The two class ground truth used during training of the modified UNET.

the closely spaced objects separately, combination of classification and segmentation losses in a weighted manner, priority based class label assignment during inference, watershed transform and connected component post processing. Thus the UNET based framework that has been borrowed from the literature is adapted for the contaminant separation problem with different novel techniques added to its architecture, that makes the overall solution modular and flexible in all possible ways. During inference, the modified UNET model takes $\sim$ 0.25 s to output a segmentation mask and the post processing takes approximately 0.1 s for each image of size 2048×2048 pixels. The total time for the entire membrane filter analysis consisting of almost 200 images using modified UNET and the cascaded post processing module is approximately 1.16 mins using a single 2080Ti GPU.

Results

The results of segmentation and classification are shown by overlaying the mask on the actual image. The semantic predictions of the modified UNET on two classes (the final prediction map is equal to the argmax outputs from the first classifier layer of the new UNET) without any post

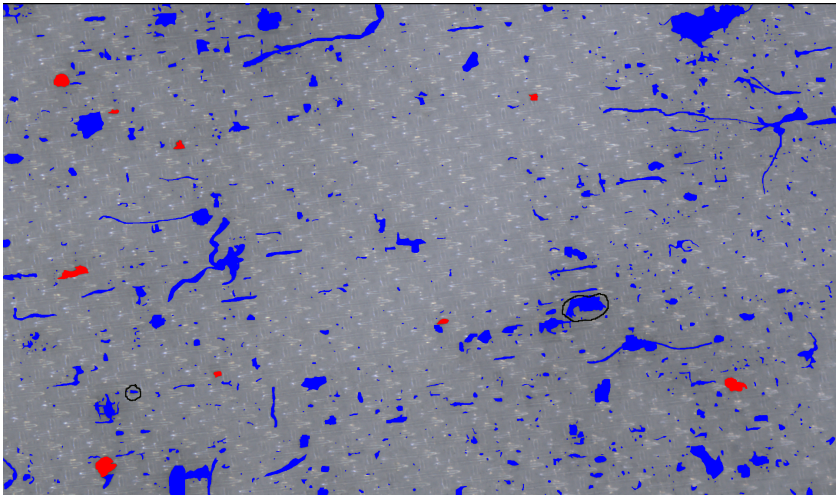


Figure 38: The modified UNET two class semantic segmentation prior to post processing. The corresponding two class ground truth image is in Figure 37. The errors are highlighted with black lines.

processing for a test image is given in Figure 38 and its corresponding instance level two class ground truth is shown in Figure 37. In this semantic segmentation, the overlapping objects are predicted as a single connected region. Because it is a pixel level classification, there is no information about the instances. By using connected component analysis, it is possible to extract the instances from this pixel segmentation map. The two class semantic segmentation output is converted into a three class instance segmentation output using connected component and extent ratio information (recreating the fiber class) as shown in Figure 39 (even this does not use the information from the border pixel map). The errors in classification are highlighted with black circles on the two class predicted image (38) and the three class output image (39).

An instance labeling connected component analysis, helps to assign labels or instance meaning to the segmented regions but still at this stage, the closely spaced objects or overlapping objects are treated as a single instance. This explains the need of using the information of the border pixels and using post processing methods like watershed transform. Priority based assignment of border pixels and object pixels while building the final segmentation map, watershed post processing with the border

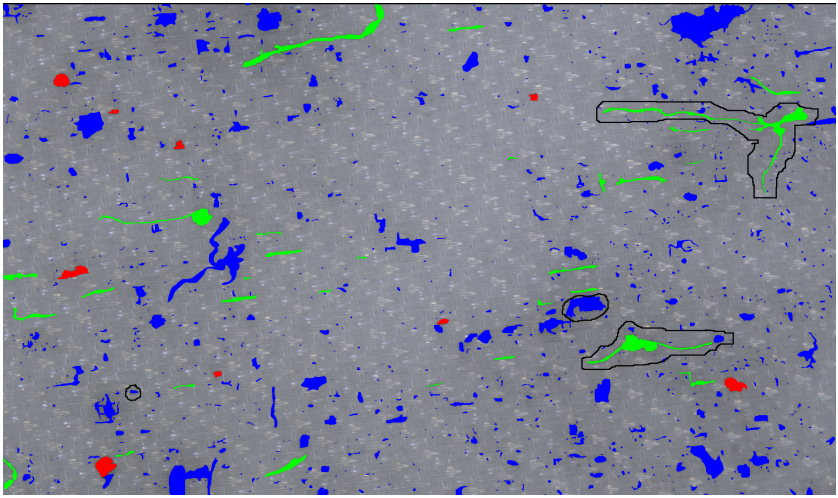


Figure 39: The modified UNET two class semantic segmentation output converted to three class instance output using connected component and extent ratio. Because no post processing is done on the overlapping instances, some of the overlapping fibers and non metallic overlaps becomes a large fiber particle. These errors are highlighted with black lines.

pixels acting as markers of the watershed transform algorithm in combination with connected component labeling and utilization of extent ratio information helps in separating the overlapping objects and also generating an instance level three class segmentation mask as given in Figure 40. Here the black markers highlights some of the separated instances using the border pixels and the seeds of watershed for some overlapping or closely spaced objects that previously belonged to a single particle. The three class instance segmentation ground truth of this image is given in Figure 34.

In all the prediction masks, it is seen that UNET not only segments the small and medium sized particles well but also segments the long and thin instances with a high precision. This framework also behaves as a good classifier and the average classification error after post processing is approximately 3% on the test set which is 1% lower than the MRCNN classifier. In MRCNN, the objects located near the boundaries of the image splits, gets segmented as two or more broken instances but in UNET since the post processing is performed on the whole image and not on the

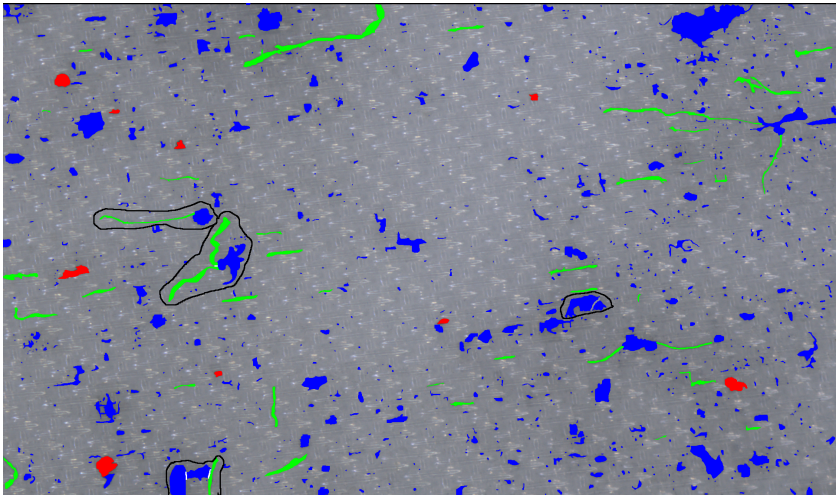


Figure 40: The three class instance segmentation mask after post processing the output of the modified UNET using watershed transform and connected component labeling. The corresponding groundtruth image is in Figure 34. Here the black lines highlight some of the closely spaced objects that have been separated into individual instances after watershed post-processing.

image splits, the connected component helps to rejoin the objects at the split boundaries into a single region. The only strong challenge faced by UNET is the closely located objects or the overlapping objects and its inability to perform instance level segmentation. Therefore, three different techniques have been used to separate these objects and overcome this challenge wisely: a weight map is used in the training that weights the borders of the closely spaced objects more forcing the network to learn them as separate objects, priority based assignment of class labels where border pixel map is given a higher priority while constructing the final segmentation output, and watershed markers are used during post processing. All these help in reducing the instance segmentation errors. Just using the last two techniques mentioned reduces the errors from 22% to 10% on the test set.

An example that explains the post processing results of this end-to-end modified UNET framework is given in Figure 41. The three sub images from left to right represents the ground truth, the semantic segmentation output and the post processed output generating instances

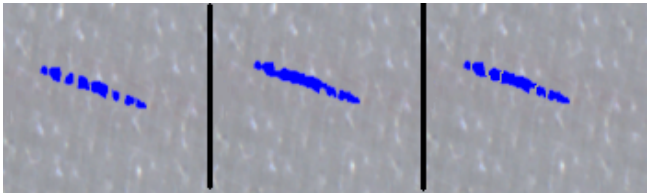


Figure 41: The image is a concatenation of three sub images each containing the same objects. From left to right it represents the ground truth, the semantic segmentation output and the post processed output generating instances for a group of very closely located non metallic objects.

for a group of very closely located non metallic objects using watershed markers. The Figure 41 clearly demonstrates how the learning of border pixels help in creating instances from the semantic output.

The results of segmentation and classification using this end-to-end framework is tabulated in Table 52. The segmentation precision, segmentation recall and F1 score are all calculated for an IOU threshold of 0.7 in a way similar to that of the MRCNN framework to make fair comparisons. The original UNET is compared with the modified UNET having the same backbone architecture and connected component analysis module to give the meaning of instances to the semantic segmented regions. The modified UNET has also been tested with different encoder architectures as explained above and some of them are compared in Table 52. Different up-sampling techniques for the decoder layers of the UNET have also been experimented with like nearest neighbor, bilinear interpolation and transposed convolution. Few of them are compared in the Table 52. Every time, a new technique is added, like an improved encoder backbone, a new up-sampling technique, watershed post processing or test time augmentation, the segmentation and the classification quotient of the framework is evaluated. The best segmentation precision and segmentation recall achieved with these modifications in the adapted UNET based end-to-end framework is 90.01 and 92.40 respectively. The calculated true precision in this case is 87.70 which is almost 16% higher in performance than the modified UNET framework with the same encoder architecture and 23% higher than the original UNET framework with Resnet-101 backbone. This huge performance gain confirms the hypothesis that each of modifications added to the original UNET framework is suitable for this problem.

Table 52: Comparison of original UNET (Or. UNET) and modified UNET (Mod. UNET) output using different CNN encoders with and without watershed transform and test time augmentations. Con Comp. is an abbreviation for connected component analysis. Watershed transform post processing is abbreviated as watershed trns. SP, SR, CQ and F1 represents the segmentation precision, recall, classification quality and F1-score respectively. Bil. Interpol. and Trans. Conv. are abbreviations for bilinear interpolation and transposed convolution. The bold numbers represent the highest segmentation precision and the F1-score for the Modified MRCNN.

Model	Techniques	SP	SR	CQ	F1
Or. UNET	Resnet-101 Con Comp.	70.46	73.19	90.70	71.80
Mod. UNET	Wide Resnet-38 Con Comp.	75.10	75.41	91.05	75.25
Mod. UNET	Wide Resnet-40 Con Comp.	77.31	78.46	92.85	77.88
Mod. UNET	Wide Resnet-38 Bil. Interpol. Watershed Trns. Con Comp.	80.71	80.33	95.26	80.52
Mod. UNET	Wide Resnet-38 Trans. Conv. Watershed Trns. Con Comp.	80.91	80.45	95.26	80.68
Mod. UNET	Wide Resnet-40 Bil. Interpol. Watershed Trns. Con Comp.	86.91	89.23	97.0	88.05
Mod. UNET	Wide Resnet-40 Trans. Conv. Watershed Trns. Con Comp.	88.04	90.55	97.01	89.27
Mod. UNET	Wide Resnet-40 Trans. Conv. Watershed Trns. Con Comp. Test Augmentation	**90.01**	92.40	97.43	**91.19**

The performance of this framework is comparable in precision and classification accuracy with the MRCNN based framework proposed previously. The classification performance of the UNET framework is

1% higher than the MRCNN framework whereas the true precision is only 0.14% lower. The F1 score of this approach is 91.19 which is slightly lower than the modified MRCNN framework which is 92.24. But the biggest gains obtained using this approach are the gains in the training time, simplicity in the optimization of the network and almost 4 times gain in the inference speed.

7.4 Summary

The two different deep learning based approaches discussed in this chapter to perform contaminant separation in cleanliness analysis and quality control, are very compact and efficient solutions in terms of cost, time and accuracy. Although, there are still some challenges to overcome in the future works but the success of these methods suggest that using deep learners is the right direction for solving such real world problems. Although the cost of an integrated system using any of the solutions is much lower than the polarized system presently used in the market, but UNET based framework is the definite winner and is more scalable than MRCNN framework which is a heavier architecture to train and has some associated overhead costs in terms of the use of more GPUs and hardware requirements. Both of these solutions depend on the previous works in literature mostly for the framework design used, but it is important to note the main contributions of this research work in terms of originality, impact and usability.

- Definition of a ROI selection criteria in Mask RCNN to propose better regions and obtain better quality of segmentation masks.
- Assignment of different weights to the mask loss, the RPN loss and the classification loss in the objective function of MRCNN.
- Cascaded post processing module with the mask segmentation head of MRCNN implementing panoptic segmentation metric.
- Transposed convolution in the up-sampling layers of the modified UNET architecture.
- Using of multiple classification layers in the UNET for learning the object masks and the border masks separately.
- Introducing the concept of extent ratio to differentiate particles of different sizes and aspect ratios.

- New objective function developed with a combination of two different classification losses (three class weighted cross entropy and binary cross entropy) and one segmentation loss (Jaccard loss).
- Using the pre-computed weight map during the training that provides higher weights to the borders of the closely spaced objects forcing the network to learn them as separate objects.
- Assignment of higher priorities to border pixels than the object-class pixels, a simple technique used in the modified UNET framework for separating the overlapped or closely spaced objects from a semantic segmentation output.
- An end-to-end solution cascaded with the UNET framework implementing watershed transform and connected component analysis to further improve the instance level segmentation. This post processing module has a huge impact on the segmentation quality of objects and hence helps to effectively solve the contaminant separation problem.
- Test time augmentations in modified MRCNN and modified UNET framework helps to improve the segmentation and classification quality by $2 - 3\%$.

There are also some suggested ways to further improve the segmentation and classification performance in future in each of these frameworks using the following techniques:

- Using overlapping tiles in MRCNN to correct the segmentation errors occurring for the objects present near the border of the image splits.
- Deal with the class imbalance problem more efficiently in MRCNN using weighted classification losses based on the class frequencies or using focal losses (CZY$^+$18; LGG$^+$17).
- Using more efficient way to learn border pixels in UNET and therefore contribute towards better segmentation of the instances.
- Replace watershed transform algorithm with more efficient algorithms like deep watershed transform (BU16), discriminative loss (BNG17) etc. to obtain higher performance.

Chapter 8

Conclusion and Future Directions

To conclude, the thesis consists of two parts. The core part of this research work describe image and video classification using supervised and semi-supervised deep learning techniques in combination with domain knowledge. The theoretical additions are supported empirically through exhaustive experimental research conducted in this work. The second part also relates certain domain specific image segmentation techniques and deep learning approaches to solve a supervised instance segmentation task.

To summarize, the first part of the thesis, it presents a statistical relational learning framework, that tightly integrates perception and reasoning together in a hybrid learning system known as Semantic based Regularization (SBR). The framework efficiently learns from data examples and logic rules expressed via a set of FOL clauses and then relaxed into their continuous fuzzy representations. In literature, the SBR framework used kernel machines as its learning machinery and incorporated FOL logic rules into the kernel machines. In this research work, for the first time SBR integrates prior knowledge with different deep convolutional neural networks. These integrated frameworks allow to learn complex feature representations from raw inputs in a much more efficient way as compared to the old kernel based shallow networks. The SBR distills prior knowledge into the model weights of the deep CNN learners during training and therefore it effectively addresses the main limitation

of deep neural networks which is their heavy dependency on the availability of labeled data. The important and notable contributions of this research work includes:

- Using deep neural networks as the learning machinery in SBR and therefore improving the classification outputs over the shallow networks with experimental results demonstrated on image classification datasets.

- Defining a novel and efficient backpropagation schema to optimize deep neural networks with logical constraints making use of expression trees.

- Increasing the generalization ability of very deep neural networks trained with supervised examples and prior knowledge which otherwise overfit and have poor generalization ability when trained with only supervised data examples.

- Demonstrating the benefits of training SBR in learning tasks with scarcity of data examples in a semi-supervised environment making it a very powerful and unified approach to be used in several real world applications.

- Highlighting the importance of prior knowledge in transductive learning emulating scenarios like photo tagging in social websites where the labeled examples contain several missing tags.

- Confirming the empirically the theoretical findings of the positive implications during optimization while using certain types of fuzzy logics like Weak-Lukasiewicz through experimental evaluations on image classification problems.

- Using optimization heuristics for inserting concave constraints in complex optimization problem of SBR, hence making SBR scalable and flexible with respect to the size of the knowledge base and the nature of domain knowledge that can be integrated.

- Demonstrating the advantages of using variable regularizers in vanilla SBR depending on the outputs of supervised learners thus providing higher degree of regularization and designing better trained models.

- Enforcing consistency among the neural network predictions of the test set by using collective classification with results demonstrated on image and video classification tasks.

- Comparing the classification performance of a large collection of deep neural networks integrated in SBR for the first time with different fuzzy logics on small, medium and learge sized image classification datasets.

- Enforcing temporal consistency among the video frames through collective classification in a video sequencing and tool annotation problem. The results of temporal smoothing using SBR fairly competes with the state-of-the-art framework, DResSys on CATARACTS dataset.

The extensive and the thorough experimental evaluations demonstrate the effectiveness of all these techniques in SBR that help in improving the accuracy of image and video classification tasks with different deep neural architectures like residual networks, network in network models etc. and achieving the state-of-the-art results in many of them. The experimental work done in this thesis provides solid evidence that integrating domain knowledge into deep neural networks has significant benefits. Therefore, exploiting the available domain knowledge is beneficial in many real world applications and future work will apply this methodology to new application areas.

- Although the focus of this research using SBR is on image and video classification, but the framework is flexible enough to be extended in future to other machine learning tasks like adversarial and generative learning as well.

- In this work, the knowledge base (KB) used in the experiments is mostly hand crafted or built from the taxonomical information available in the WordNet hierarchy associated with most of the image datasets. As a future work, it is necessary to study how to automatically design the KB by learning the background knowledge/rules from labeled data in a more principled way.

- Also, it would be useful to study the process of automatically assigning weights or priorities to the rules during the learning process.

- The optimization strategies should also be explored and improved by searching better the hypothesis space.
- Another interesting direction to explore as future work is the possibility to deal with uncertainty in rules and using rules that are only valid in a subset of the data patterns.

To summarize the second part of the thesis, that covers an industrial problem of identifying particle contamination, measuring them and helping in their automatic removal for cleanliness analysis, two competing deep learning based approaches of object detection and segmentation are proposed namely modified Mask RCNN framework with improved region proposal selection criteria and panoptic post processing and modified UNET with Jaccard loss and watershed post processing. Although the second part of the thesis is bit different from the first part but it can still be viewed on a broader perspective as using domain specific image segmentation techniques with deep neural networks integrated in a less sophisticated fashion but still aiming to solve vision tasks in a supervised setting. Here, an industrial problem is solved using deep learning approaches that are more efficient and accurate than the existing solutions in the market. They help in segmenting and classifying the contaminant particles from the background filters and performing different statistical analysis to determine the number, size, particle type, and morphology of the particles. Although the frameworks used in solving the industrial problem is dependent on the previous works in literature, but there are significant amount of novel contributions added through this research work that not only makes the existing frameworks more powerful and more adaptable to real world problems but also shows the pathway to build end-to-end solutions. The modifications of the existing frameworks are summarized in Section 7.4. The success of the two approaches discussed in Chapter 7 suggest that using deep learners is the right direction for solving this problem. Both of the solutions are compact, easily integrable, portable and scalable to other related problems in different industries like automobile, aerospace and manufacturing industries. For example: these solutions can be applied in detection of roads in aerial satellite imagery, for counting red and white blood cells in histopathological microscopic images etc. There are few weaknesses of the proposed solutions as already discussed in Section 7.4 that can be pursued as the future work to improve the system and design more accurate solutions for such real world problems.

Appendix A

Chapter 5 and Chapter 6: Image Classification and Video Classification

A.0.1 Experimental Analysis on ANIMAL Dataset

Results

The following tables represent the comparative results of different t-norms used in different experimental simulations in Chapter 5 on Winston Animal image classification benchmark.

Table 53 and Table 54 compares all the different t-norms used with rules and with collective classification in constant weight based SBR and vanilla SBR for 32×32 and 64×64 pixel images on Fully transductive setting of Winston Animal Benchmark. Table 55 and Table 56 compares all the different t-norms used with rules and with collective classification in constant weight based SBR and vanilla SBR for 32×32 and 64×64 pixel images on Partially transductive setting of Winston Animal Benchmark.

Table 57 compares all the t-norms with different image resolutions in constant and vanilla non transductive SBR. In all of these tables, the naming conventions like CC-WL and CC-WL (λ_l^k) are Weak-Lukasiewicz, CC-L and CC-L (λ_l^k) are Lukasiewicz, CC-P and CC-P (λ_l^k) are Product and CC-M and CC-M (λ_l^k) are Minimum t-norms used in collective classification in constant and vanilla SBR respectively.

Table 53: Comparison of the classification accuracies using rules for the 7 final classes for Fully Labeled Transductive setting of Winston benchmark.

#Pat	WL	WL(λ_l^k)	L	L(λ_l^k)	P	P(λ_l^k)	M	M(λ_l^k)
32×32 pixel Images								
3325	93.3	94.2	93.2	93.8	93.3	93.8	93.1	93.8
2450	90.8	91.2	90.9	91.2	90.8	91.1	90.7	91.1
1076	84.2	85.0	84.1	84.8	84.1	84.8	84.1	84.9
500	81.0	81.9	81.0	81.7	81.0	81.7	81.0	81.6
100	68.2	68.8	68.1	68.7	68.0	68.8	68.0	68.8
64×64 pixel Images								
3325	96.4	97.5	96.1	97.2	96.0	97.2	96.0	97.1
2450	92.0	94.0	91.8	93.2	91.6	93.2	91.6	93.5
1076	84.9	85.3	84.2	85.0	84.2	85.0	84.2	85.0
500	82.6	83.0	82.1	82.9	82.1	82.8	82.1	82.8
100	69.0	70.1	69.0	69.8	69.0	69.9	69.0	69.9

Table 54: Comparison of the collective classification accuracies for the 7 final classes for Fully Labeled Transductive setting of Winston benchmark.

#Pat	CC-WL	CC-WL(λ_l^k)	CC-L	CC-L(λ_l^k)	CC-P	CC-P(λ_l^k)	CC-M	CC-M(λ_l^k)
32×32 pixel Images								
3325	94.1	94.4	94.0	94.3	93.9	94.0	93.7	94.0
2450	90.9	91.4	90.9	91.4	90.8	91.3	90.7	91.2
1076	84.9	85.3	84.9	85.3	84.8	85.2	84.6	84.9
500	81.8	82.0	81.7	81.9	81.6	81.7	81.5	81.6
100	68.4	69.0	68.4	69.0	68.2	68.8	68.2	68.8
64×64 pixel Images								
3325	97.2	97.8	97.1	97.8	97.0	97.5	97.0	97.4
2450	93.9	94.1	93.8	94.1	93.5	93.8	93.5	93.7
1076	85.2	85.4	85.2	85.3	85.0	85.2	85.0	85.2
500	83.0	83.0	82.9	83.0	82.7	82.8	82.7	82.8
100	70.0	70.3	70.0	70.3	69.9	70.1	69.9	70.0

Table 55: Comparison of the classification accuracies for the 7 final classes trained with rules in Partially Labeled Transductive setting on Winston benchmark.

#Pat	WL	WL(λ_l^k)	L	L(λ_l^k)	P	P(λ_l^k)	M	M(λ_l^k)
32×32 pixel Images								
3325	94.2	**96.3**	94.0	95.7	94.0	95.6	94.1	95.8
2450	91.2	92.2	91.0	91.6	91.2	91.7	91.2	91.7
1076	84.2	85.3	84.0	85.4	84.0	85.2	84.1	85.2
500	81.9	82.9	81.7	82.6	81.6	82.6	81.6	82.6
100	69.2	70.5	69.2	70.1	69.1	70.1	69.0	69.9
64×64 pixel Images								
3325	97.7	**99.0**	97.6	98.9	97.5	98.9	97.2	98.9
2450	92.9	94.5	92.9	94.4	92.8	94.5	92.7	94.4
1076	85.1	85.9	85.1	85.9	85.1	85.7	85.1	85.8
500	81.9	83.2	81.8	83.2	81.9	83.2	81.9	83.2
100	74.6	75.5	74.4	75.5	74.5	75.5	74.4	75.4

Table 56: Comparison of the collective classification accuracies for the 7 final classes for Partially Labeled Transductive setting on Winston benchmark.

#Pat	CC-WL	CC-WL(λ_l^k)	CC-L	CC-L(λ_l^k)	CC-P	CC-P(λ_l^k)	CC-M	CC-M(λ_l^k)
32×32 pixel Images								
3325	96.1	**96.8**	96.0	96.7	95.8	96.4	95.7	96.4
2450	92.1	92.4	92.0	92.3	91.8	92.0	91.7	91.9
1076	85.0	85.6	85.0	85.4	84.8	85.0	84.8	85.1
500	82.8	83.0	82.7	83.0	82.6	82.8	82.6	82.8
100	70.4	70.7	70.3	70.5	70.1	70.5	70.1	70.5
64×64 pixel Images								
3325	98.9	**99.2**	98.6	99.2	98.3	98.5	98.2	98.6
2450	94.1	94.9	94.1	94.9	94.0	94.5	94.0	94.4
1076	85.8	85.9	85.8	86.0	85.4	85.7	85.5	85.7
500	83.0	83.4	83.0	83.4	82.9	83.3	82.9	83.3
100	75.4	75.6	75.4	75.6	75.3	75.6	73.2	75.3

Table 57: Comparison of the collective classification accuracies for the 7 final classes in non transductive setting with 32×32 and 64×64 images of Winston benchmark.

#Pat	CC-WL	CC-WL(λ_l^k)	CC-L	CC-L(λ_l^k)	CC-P	CC-P(λ_l^k)	CC-M	CC-M(λ_l^k)
(32×32 pixel Images)								
3325	92.9	**93.9**	92.5	93.2	91.5	91.6	91.6	91.8
2450	88.1	90.1	87.9	89.2	87.9	88.0	87.9	88.0
1076	83.5	83.6	83.4	83.5	81.5	82.8	81.8	82.8
500	81.2	81.3	81.0	81.1	80.0	81.5	81.1	81.2
100	68.0	68.1	67.8	68.0	67.0	67.5	67.1	67.8
(64×64 pixel Images)								
3325	96.5	**96.6**	96.4	**96.6**	95.3	96.5	95.3	96.5
2450	92.0	93.1	91.9	92.8	91.9	92.5	91.9	92.2
1076	84.7	84.9	84.6	84.7	84.4	84.7	84.4	84.8
500	82.1	82.3	81.9	82.0	81.9	82.3	81.7	82.0
100	69.4	69.6	69.3	69.6	69.3	69.6	69.3	69.4

A.0.2 Experimental Analysis on CIFAR-10 Dataset

Results

Table 58 compares the collective classification error rates obtained on the test set from the trained CNN models in the non transductive mode using different selection of t-norms.

Table 59represents the comparison of the collective classification error rates for the 10 final classes on different t-norms using scarce training data in CIFAR-10 experiments in non transductive mode.

Table 60 represents the error rate for the 10 final classes of CIFAR-10 for different deep architectures in transductive learning using prior knowledge. It also lists the collective classification errors over the network outputs during inference. All the comparisons are made between different t-norm selections and using constant and predicate dependent regularizers.

In all of these tables, the naming conventions are the same as in the ANIMAL dataset like CC-WL and CC-WL (λ_l^k) are Weak-Lukaseiwicz, CC-L and CC-L (λ_l^k) are Lukaseiwicz, CC-P and CC-P (λ_l^k) are Product and CC-M and CC-M (λ_l^k) are Minimum t-norms used in collective classification in constant and vanilla SBR respectively. The deep CNN architecture names are presented in the columns called as Models. NIN is abbreviated for Network in Network model, Res for Resnets, PRes for Pre-activated Resnets, APN and MPN for Additive Pyramid Resnets and Multiplicative Pyramid Resnets respectively.

Table 58: Error rate for the 10 final classes on CIFAR-10 for different deep architectures in non transductive learning and collective classification over the network outputs using different selection of t-norms with constant metaparameters or predicate dependent metaparameters λ_l^k.

Models	CNN	CC-WL	CC-WL(λ_l^k)	CC-L	CC-L(λ_l^k)	CC-P	CC-P(λ_l^k)	CC-M	CC-M(λ_l^k)
NIN-50	8.81	8.71	8.66	8.77	8.68	8.78	8.70	8.79	8.71
Res-20	8.75	8.01	7.89	8.07	7.92	8.5	8.0	8.5	8.0
Res-32	7.51	7.09	6.88	7.12	6.92	7.13	6.92	7.13	6.93
Res-44	7.17	6.58	6.32	6.78	6.42	6.8	6.5	6.9	6.6
Res-56	6.97	6.39	6.01	6.44	6.13	6.52	6.29	6.56	6.31
Res-110	6.61	5.96	5.54	6.0	5.82	6.3	6.0	6.32	6.01
Res-164	5.93	5.76	5.53	5.88	5.61	6.0	5.8	6.0	5.84
Res-1202	**7.93**	5.17	**5.07**	5.19	5.10	5.23	5.1	5.3	5.15
PRes-110	6.37	6.15	6.10	6.21	6.15	6.17	6.12	6.18	6.13
PRes-164	5.46	5.20	5.15	5.36	5.20	5.30	5.26	5.31	5.21
PRes-1202	**6.85**	6.05	**6.01**	6.08	6.03	6.1	6.05	6.3	6.1
APN-110, α=48	4.62	4.6	4.43	4.6	4.48	4.6	4.5	4.6	4.52
APN-110, α=84	4.27	4.15	4.09	4.2	4.16	4.2	4.18	4.21	4.19
APN-110, α=270	3.73	3.64	3.60	3.70	3.67	3.7	3.65	3.7	3.64
APN-164, α=48	4.21	4.0	3.95	4.0	3.96	4.1	4.0	4.1	4.01
APN-164, α=84	3.96	3.95	3.66	3.95	3.67	3.95	3.7	3.95	3.72
APN-164, α=270	3.48	3.44	3.40	3.44	3.44	3.45	3.41	3.45	3.42
APN-200, α=240	3.44	3.40	3.38	3.41	3.39	3.41	3.40	3.41	3.40
APN-236, α=220	3.40	3.40	3.31	3.40	3.40	3.40	3.5	3.40	3.5
APN-272, α=200	3.31	3.30	**3.22**	3.30	3.25	3.31	3.25	3.32	3.26
MPN-110, α=8	4.50	4.39	4.30	4.40	4.32	4.41	4.35	4.41	4.35
MPN-110, α=27	4.06	3.94	3.8	3.95	3.8	3.98	3.81	3.91	3.84

Table 59: Comparison of the collective classification error rate for the 10 final classes using scarce training data in non transductive mode.

Models	%Data	CNN	CC-WL	CC-WL(λ_l^k)	CC-L	CC-L(λ_l^k)	CC-P	CC-P(λ_l^k)	CC-M	CC-M(λ_l^k)
NIN-50	10	27.89	27.14	26.66	27.77	26.68	27.78	26.70	27.79	26.71
	20	21.2	20.44	20.37	20.45	20.38	20.51	20.45	20.53	20.48
	50	13.10	12.99	12.95	13.0	12.97	13.01	12.99	13.01	12.99
Res-20	10	27.13	26.44	26.21	26.47	26.23	26.5	26.25	26.51	26.27
	20	20.7	19.23	19.01	19.41	19.12	19.45	19.2	19.45	19.19
	50	12.46	12.32	12.1	12.35	12.11	12.36	12.12	12.36	12.12
Res-32	10	26.62	25.51	25.07	25.55	25.10	25.56	25.12	25.56	25.13
	20	18.61	17.89	17.75	17.92	17.80	17.95	17.81	17.95	17.81
	50	12.14	11.99	11.90	12.04	11.92	12.1	12.0	12.1	12.0
Res-44	10	25.36	25.12	25.04	25.24	25.18	25.24	25.18	25.24	25.18
	20	19.02	17.98	17.44	17.99	17.47	17.99	17.47	17.99	17.47
	50	12.08	11.94	11.82	11.97	11.85	12.01	11.9	12.01	11.95
Res-56	10	25.19	25.03	24.9	25.05	24.91	25.04	24.91	25.04	24.91
	20	16.39	15.98	15.90	16.01	15.91	16.0	15.91	16.0	15.91
	50	11.88	11.42	11.14	11.47	11.15	11.47	11.15	11.47	11.15
Res-110	10	24.78	24.67	24.10	24.69	24.10	24.70	24.11	24.70	24.11
	20	15.14	15.01	14.84	15.05	14.88	15.06	14.90	15.06	14.90
	50	10.14	10.10	10.0	10.11	10.01	10.12	10.05	10.12	10.05
PRes-110	10	23.96	23.78	23.64	23.81	23.67	23.81	23.67	23.81	23.67
	20	14.77	14.32	13.98	14.35	14.0	14.35	14.1	14.40	14.30
	50	9.99	9.91	9.85	9.92	9.87	9.91	9.88	9.91	9.88
APN-164, $\alpha = 270$	10	22.85	22.78	22.67	22.80	22.70	22.80	22.70	22.80	22.70
	20	13.56	13.41	13.30	13.40	13.30	13.42	13.32	13.42	13.32
	50	8.77	8.56	8.23	8.59	8.25	8.60	8.26	8.60	8.26

Table 60: Error rate for the 10 final classes of CIFAR-10 for different deep architectures in transductive learning and collective classification over the network outputs using different selection of t-norms with constant metaparameters or predicate dependent metaparameters λ_l^k.

Models	CNN	WL	CC-WL	CC-WL(λ_l^k)	P	CC-P	CC-P(λ_l^k)	M	CC-M	CC-M(λ_l^k)
NIN-50	8.81	8.21	8.12	8.07	8.25	8.18	8.1	8.3	8.2	8.19
Res-20	8.75	7.7	7.44	7.05	7.9	7.5	7.1	8.0	7.7	7.2
Res-32	7.51	6.85	6.6	6.46	6.9	6.7	6.5	6.9	6.75	6.6
Res-44	7.17	6.23	6.0	5.94	6.3	6.1	6.0	6.31	6.12	6.05
Res-56	6.97	6.2	5.94	5.81	6.2	6.0	5.85	6.21	6.01	5.85
Res-110	6.61	5.52	5.13	4.89	5.57	5.2	4.95	5.55	5.17	4.94
Res-164	5.93	5.2	4.92	4.7	5.28	4.98	4.72	5.29	4.99	4.75
Res-1202	7.93	4.91	4.74	4.5	4.92	4.75	4.55	4.93	4.83	4.67
PRes-110	6.37	5.88	5.82	5.75	5.89	5.82	5.76	5.90	5.85	5.77
PRes-164	5.46	5.02	4.98	4.91	5.05	5.0	4.95	5.05	5.0	4.95
PRes-1202	6.85	5.8	5.72	5.67	5.85	5.75	5.69	5.88	5.74	5.70
APN-110,α=48	4.62	4.0	3.84	3.7	4.05	3.9	3.8	4.06	3.9	3.8
APN-110,α=84	4.27	3.65	3.6	3.49	3.62	3.59	3.45	3.65	3.6	3.5
APN-110,α=270	3.73	3.04	2.99	2.88	3.05	3.0	2.9	3.1	3.06	3.0
APN-164,α=48	4.21	3.6	3.56	3.40	3.6	3.5	3.4	3.61	3.59	3.48
APN-164,α=84	3.96	3.1	2.8	2.67	3.1	2.9	2.7	3.1	2.9	2.8
APN-164,α=270	3.48	2.94	2.87	2.71	2.95	2.89	2.75	2.95	2.90	2.76
APN-200,α=240	3.44	2.88	2.78	2.67	2.89	2.8	2.7	2.9	2.81	2.74
APN-236,α=220	3.40	2.8	2.75	2.64	2.84	2.72	2.67	2.85	2.75	2.68
APN-272,α=200	3.31	2.7	2.61	2.56	2.78	2.64	2.59	2.78	2.65	2.59
MPN-110,α=8	4.50	3.89	3.7	3.66	3.88	3.71	3.67	3.9	3.75	3.68
MPN-110,α=27	4.06	3.49	3.41	3.36	3.50	3.42	3.38	3.5	3.43	3.40

A.0.3 Experimental Analysis on CIFAR-100 Dataset

The knowledge base used in CIFAR-100 experiments is given in the long Table 61 below. The first set of rules exploits the hierarchical relationship between super and fine classes whereas the next set of rules exploits the relationship using the additional predicates.

Table 61: Rules used for CIFAR-100 experiments. The rules are divided into two groups: First 150 rules expressing the class taxonomy, and the next group of 50 hand-crafted rules express additional semantic information.

$\forall X$ APPLE(X) $\vee$AQUARIUM FISH(X) $\vee$BABY(X) $\vee$BEAR(X) $\vee$BEAVER (X) $\vee$ BED(X) $\vee$BEE(X) $\vee$BEETLE(X) $\vee$BICYCLE(X) $\vee$BOTTLE(X) $\vee$BOWL (X) $\vee$BOY(X) $\vee$BRIDGE(X) $\vee$BUS(X) $\vee$BUTTERFLY(X) $\vee$CAMEL(X) $\vee$CAN(X) $\vee$CASTLE(X) $\vee$ CATERPILLAR(X) $\vee$CATTLE(X) $\vee$CHAIR(X) $\vee$CHIMPANZEE(X) $\vee$ CLOCK(X) $\vee$CLOUD(X) $\vee$COCKROACH(X) $\vee$COUCH(X) $\vee$CRAB(X) $\vee$ CROCODILE(X) $\vee$CUP(X) $\vee$DINOSAUR(X) $\vee$DOLPHIN(X) $\vee$ELEPHANT(X) $\vee$FLATFISH(X) $\vee$FOREST(X) $\vee$FOX(X) $\vee$GIRL(X) $\vee$HAMSTER(X) $\vee$HOUSE(X) $\vee$KANGAROO(X) $\vee$KEYBOARD(X) $\vee$LAMP(X) $\vee$ LAWN_MOWER(X) $\vee$LEOPARD(X) $\vee$LION(X) $\vee$LIZARD(X) $\vee$LOBSTER(X) $\vee$MAN(X) $\vee$MAPLE_TREE(X) $\vee$MOTORCYCLE(X) $\vee$MOUNTAIN(X) $\vee$MOUSE(X) $\vee$MUSHROOM(X) $\vee$OAK_TREE(X) $\vee$ORANGE(X) $\vee$ORCHID(X) $\vee$OTTER(X) $\vee$PALM_TREE(X) $\vee$PEAR(X) $\vee$PICKUP_TRUCK(X) $\vee$PINE_TREE(X) $\vee$PLAIN(X) $\vee$PLATE(X) $\vee$POPPY(X) $\vee$PORCUPINE(X) $\vee$POSSUM(X) $\vee$RABBIT(X) $\vee$RACCOON(X) $\vee$RAY(X) $\vee$ROAD(X) $\vee$ROCKET(X) $\vee$ROSE(X) $\vee$SEA(X) $\vee$SEAL(X) $\vee$SHARK(X) $\vee$SHREW(X) $\vee$SKUNK(X) $\vee$SKYSCRAPER(X) $\vee$SNAIL(X) $\vee$SNAKE(X) $\vee$SPIDER(X) $\vee$SQUIRREL(X) $\vee$ STREETCAR(X) $\vee$SUNFLOWER(X) $\vee$SWEET_PEPPER(X) $\vee$TABLE(X) $\vee$TANK(X) $\vee$TELEPHONE(X) $\vee$TELEVISION(X) $\vee$TIGER(X) $\vee$ TRACTOR(X) $\vee$TRAIN(X) $\vee$TROUT(X) $\vee$TULIP(X) $\vee$TURTLE(X) $\vee$WARDROBE(X) $\vee$WHALE(X) $\vee$WILLOW_TREE(X) $\vee$WOLF(X) $\vee$WOMAN(X) $\vee$WORM(X) $\forall X$ AQUATIC MAMMALS(X) $\vee$FISH(X) $\vee$FLOWERS(X) $\vee$FOOD CONTAINERS(X) $\vee$ FRUIT AND VEGETABLES(X) $\vee$HOUSEHOLD ELECTRICAL (X) $\vee$HOUSEHOLD FURNITURE(X) $\vee$INSECTS(X) $\vee$ LARGE CARNIVORES(X) $\vee$ MAN-MADE OUTDOOR (X) $\vee$ NATURAL OUTDOOR SCENES(X) $\vee$ OMNIVORES AND HERBIVORES(X) $\vee$MEDIUM MAMMALS(X) $\vee$INVERTEBRATES(X) $\vee$PEOPLE(X) $\vee$REPTILES(X) $\vee$SMALL MAMMALS(X) $\vee$TREES(X) $\vee$VEHICLES 1(X) $\vee$VEHICLES 2(X)

$\forall X$ AQUATIC MAMMALS$(X) \Rightarrow$ BEAVER$(X) \vee$DOLPHIN$(X) \vee$OTTER$(X) \vee$SEAL$(X) \vee$WHALE(X)
$\forall X$ BEAVER$(X) \Rightarrow$ AQUATIC MAMMALS(X)
$\forall X$ DOLPHIN$(X) \Rightarrow$ AQUATIC MAMMALS(X)
$\forall X$ OTTER$(X) \Rightarrow$ AQUATIC MAMMALS(X)
$\forall X$ SEAL$(X) \Rightarrow$ AQUATIC MAMMALS(X)
$\forall X$ WHALE$(X) \Rightarrow$ AQUATIC MAMMALS(X)

$\forall X$ FISH$(X) \Rightarrow$ AQUARIUM FISH$(X) \vee$FLATFISH$(X) \vee$RAY$(X) \vee$SHARK$(X) \vee$TROUT(X)
$\forall X$ AQUARIUM_FISH$(X) \Rightarrow$ FISH(X)
$\forall X$ FLATFISH$(X) \Rightarrow$ FISH(X)
$\forall X$ RAY$(X) \Rightarrow$ FISH(X)
$\forall X$ SHARK$(X) \Rightarrow$ FISH (X)
$\forall X$ TROUT$(X) \Rightarrow$ FISH(X)

$\forall X$ FLOWERS$(X) \Rightarrow$ ORCHID$(X) \vee$POPPY$(X) \vee$ROSE$(X) \vee$SUNFLOWER$(X) \vee$TULIP(X)
$\forall X$ ORCHID$(X) \Rightarrow$ FLOWERS(X)
$\forall X$ POPPY$(X) \Rightarrow$ FLOWERS(X)
$\forall X$ ROSE$(X) \Rightarrow$ FLOWERS(X)
$\forall X$ SUNFLOWER$(X) \Rightarrow$ FLOWERS(X)
$\forall X$ TULIP$(X) \Rightarrow$ FLOWERS(X)

$\forall X$ FOOD_CONTAINERS$(X) \Rightarrow$ BOTTLE$(X) \vee$BOWL$(X) \vee$CAN$(X) \vee$CUP$(X) \vee$PLATE(X)
$\forall X$ BOTTLE$(X) \Rightarrow$ FOOD_CONTAINERS (X)
$\forall X$ BOWL$(X) \Rightarrow$ FOOD_CONTAINERS (X)
$\forall X$ CAN$(X) \Rightarrow$ FOOD_CONTAINERS (X)

$\forall X$ CUP(X) $\Rightarrow$ FOOD_CONTAINERS (X)
$\forall X$ PLATE(X) $\Rightarrow$ FOOD_CONTAINERS (X)

$\forall X$ FRUIT_AND_VEGETABLES(X) $\Rightarrow$ APPLE(X) $\vee$MUSHROOM(X) $\vee$ORANGE(X) $\vee$PEAR(X)
$\vee$SWEET_PEPPER(X)
$\forall X$ APPLE(X) $\Rightarrow$ FRUIT_AND_VEGETABLES(X)
$\forall X$ MUSHROOM(X) $\Rightarrow$ FRUIT_AND_VEGETABLES(X)
$\forall X$ ORANGE(X) $\Rightarrow$ FRUIT_AND_VEGETABLES(X)
$\forall X$ PEAR(X) $\Rightarrow$ FRUIT_AND_VEGETABLES(X)
$\forall X$ SWEET_PEPPER(X) $\Rightarrow$ FRUIT_AND_VEGETABLES(X)
$\forall X$ HOUSEHOLD_ELECTRICAL_DEVICES(X) $\Rightarrow$ CLOCK(X) $\vee$KEYBOARD(X) $\vee$LAMP(X)
$\vee$TELEPHONE(X) $\vee$TELEVISION(X)
$\forall X$ CLOCK(X) $\Rightarrow$ HOUSEHOLD_ELECTRICAL_DEVICES(X)
$\forall X$ KEYBOARD(X) $\Rightarrow$ HOUSEHOLD_ELECTRICAL_DEVICES(X)
$\forall X$ LAMP(X) $\Rightarrow$ HOUSEHOLD_ELECTRICAL_DEVICES(X)
$\forall X$ TELEPHONE(X) $\Rightarrow$ HOUSEHOLD_ELECTRICAL_DEVICES(X)
$\forall X$ TELEVISION(X) $\Rightarrow$ HOUSEHOLD_ELECTRICAL_DEVICES(X)

$\forall X$ HOUSEHOLD_FURNITURE(X) $\Rightarrow$ BED(X) $\vee$ CHAIR(X) $\vee$COUCH(X) $\vee$TABLE(X) $\vee$WARDROBE(X)
$\forall X$ BED(X) $\Rightarrow$ HOUSEHOLD_FURNITURE(X)
$\forall X$ CHAIR(X) $\Rightarrow$ HOUSEHOLD_FURNITURE(X)
$\forall X$ COUCH(X) $\Rightarrow$ HOUSEHOLD_FURNITURE(X)
$\forall X$ TABLE(X) $\Rightarrow$ HOUSEHOLD_FURNITURE(X)
$\forall X$ WARDROBE(X) $\Rightarrow$ HOUSEHOLD_FURNITURE(X)

$\forall X$ INSECTS$(X) \Rightarrow$ BEE$(X) \vee$ BEETLE$(X) \vee$ BUTTERFLY$(X) \vee$ CATERPILLAR$(X) \vee$ COCKROACH(X)
$\forall X$ BEE$(X) \Rightarrow$ INSECTS(X)
$\forall X$ BEETLE$(X) \Rightarrow$ INSECTS(X)
$\forall X$ BUTTERFLY$(X) \Rightarrow$ INSECTS(X)
$\forall X$ CATERPILLAR$(X) \Rightarrow$ INSECTS(X)
$\forall X$ COCKROACH$(X) \Rightarrow$ INSECTS(X)

$\forall X$ LARGE_CARNIVORES$(X) \Rightarrow$ BEAR$(X) \vee$LEOPARD$(X) \vee$LION$(X) \vee$TIGER $(X) \vee$WOLF(X)
$\forall X$ BEAR$(X) \Rightarrow$ LARGE_CARNIVORES(X)
$\forall X$ LEOPARD$(X) \Rightarrow$ LARGE_CARNIVORES(X)
$\forall X$ LION$(X) \Rightarrow$ LARGE_CARNIVORES(X)
$\forall X$ TIGER$(X) \Rightarrow$ LARGE_CARNIVORES(X)
$\forall X$ WOLF$(X) \Rightarrow$ LARGE_CARNIVORES(X)

$\forall X$ LARGE_MAN-MADE_OUTDOOR_THINGS$(X) \Rightarrow$ BRIDGE$(X) \vee$CASTLE$(X) \vee$HOUSE$(X) \vee$ROAD(X)
$\vee$SKYSCRAPER(X)
$\forall X$ BRIDGE$(X) \Rightarrow$ LARGE_MAN-MADE_OUTDOOR_THINGS(X)
$\forall X$ CASTLE$(X) \Rightarrow$ LARGE_MAN-MADE_OUTDOOR_THINGS(X)
$\forall X$ HOUSE$(X) \Rightarrow$ LARGE_MAN-MADE_OUTDOOR_THINGS(X)
$\forall X$ ROAD$(X) \Rightarrow$ LARGE_MAN-MADE_OUTDOOR_THINGS(X)
$\forall X$ SKYSCRAPER$(X) \Rightarrow$ LARGE_MAN-MADE_OUTDOOR_THINGS(X)
$\forall X$ LARGE_NATURAL_OUTDOOR_SCENES$(X) \Rightarrow$CLOUD$(X) \vee$FOREST$(X) \vee$MOUNTAIN(X)
$\vee$PLAIN$(X) \vee$SEA(X)
$\forall X$ CLOUD$(X) \Rightarrow$ LARGE_NATURAL_OUTDOOR_SCENES(X)
$\forall X$ FOREST$(X) \Rightarrow$ LARGE_NATURAL_OUTDOOR_SCENES(X)

$\forall X$ MOUNTAIN$(X) \Rightarrow$ LARGE_NATURAL_OUTDOOR_SCENES(X)
$\forall X$ PLAIN$(X) \Rightarrow$ LARGE_NATURAL_OUTDOOR_SCENES(X)
$\forall X$ SEA$(X) \Rightarrow$ LARGE_NATURAL_OUTDOOR_SCENES(X)

$\forall X$ LARGE_OMNIVORES_AND_HERBIVORES$(X) \Rightarrow$ CAMEL$(X) \vee$ CATTLE$(X) \vee$ CHIMPANZEE(X)
$\vee$ ELEPHANT$(X) \vee$ KANGAROO(X)
$\forall X$ CAMEL$(X) \Rightarrow$ LARGE_OMNIVORES_AND_HERBIVORES(X)
$\forall X$ CATTLE$(X) \Rightarrow$ LARGE_OMNIVORES_AND_HERBIVORES(X)
$\forall X$ CHIMPANZEE$(X) \Rightarrow$ LARGE_OMNIVORES_AND_HERBIVORES(X)
$\forall X$ ELEPHANT$(X) \Rightarrow$ LARGE_OMNIVORES_AND_HERBIVORES(X)
$\forall X$ KANGAROO$(X) \Rightarrow$ LARGE_OMNIVORES_AND_HERBIVORES(X)

$\forall X$ MEDIUM_MAMMALS$(X) \Rightarrow$ FOX$(X) \vee$ PORCUPINE$(X) \vee$ POSSUM$(X) \vee$ RACCOON(X)
$\vee$ SKUNK(X)
$\forall X$ FOX$(X) \Rightarrow$ MEDIUM_MAMMALS(X)
$\forall X$ PORCUPINE$(X) \Rightarrow$ MEDIUM_MAMMALS(X)
$\forall X$ POSSUM$(X) \Rightarrow$ MEDIUM_MAMMALS(X)
$\forall X$ RACCOON$(X) \Rightarrow$ MEDIUM_MAMMALS(X)
$\forall X$ SKUNK$(X) \Rightarrow$ MEDIUM_MAMMALS(X)

$\forall X$ NON-INSECT_INVERTEBRATES$(X) \Rightarrow$ CRAB$(X) \vee$ LOBSTER$(X) \vee$ SNAIL$(X) \vee$SPIDER(X)
$\vee$WORM(X)
$\forall X$ CRAB$(X) \Rightarrow$ NON-INSECT_INVERTEBRATES(X)
$\forall X$ LOBSTER$(X) \Rightarrow$ NON-INSECT_INVERTEBRATES(X)
$\forall X$ SNAIL$(X) \Rightarrow$ NON-INSECT_INVERTEBRATES(X)

$\forall X$ SPIDER(X) $\Rightarrow$ NON-INSECT_INVERTEBRATES(X)
$\forall X$ WORM(X) $\Rightarrow$ NON-INSECT_INVERTEBRATES(X)

$\forall X$ PEOPLE(X) $\Rightarrow$ BABY(X) $\vee$MAN(X) $\vee$WOMAN(X) $\vee$BOY(X) $\vee$GIRL(X)
$\forall X$ BABY(X) $\Rightarrow$ PEOPLE(X)
$\forall X$ BOY(X) $\Rightarrow$ PEOPLE(X)
$\forall X$ GIRL(X) $\Rightarrow$ PEOPLE(X)
$\forall X$ MAN(X) $\Rightarrow$ PEOPLE(X)
$\forall X$ WOMAN(X) $\Rightarrow$ PEOPLE(X)

$\forall X$ REPTILES(X) $\Rightarrow$ CROCODILE(X) $\vee$ DINOSAUR(X) $\vee$ LIZARD(X) $\vee$SNAKE(X) $\vee$ TURTLE(X)
$\forall X$ CROCODILE(X) $\Rightarrow$ REPTILES(X)
$\forall X$ DINOSAUR(X) $\Rightarrow$ REPTILES(X)
$\forall X$ LIZARD(X) $\Rightarrow$ REPTILES(X)
$\forall X$ SNAKE(X) $\Rightarrow$ REPTILES(X)
$\forall X$ TURTLE(X) $\Rightarrow$ REPTILES(X)

$\forall X$ SMALL_MAMMALS(X) $\Rightarrow$ HAMSTER(X) $\vee$ MOUSE(X) $\vee$ RABBIT(X) $\vee$SHREW(X) $\vee$ SQUIRREL(X)
$\forall X$ HAMSTER(X) $\Rightarrow$ SMALL_MAMMALS(X)
$\forall X$ MOUSE(X) $\Rightarrow$ SMALL_MAMMALS(X)
$\forall X$ RABBIT(X) $\Rightarrow$ SMALL_MAMMALS(X)
$\forall X$ SHREW(X) $\Rightarrow$ SMALL_MAMMALS(X)
$\forall X$ SQUIRREL(X) $\Rightarrow$ SMALL_MAMMALS(X)

$\forall X$ TREES(X) $\Rightarrow$ MAPLE_TREE(X) $\vee$ OAK_TREE(X) $\vee$ PALM_TREE(X) $\vee$PINE_TREE(X)
$\vee$ WILLOW_TREE(X)

$\forall X$ MAPLE_TREE$(X) \Rightarrow$ TREES(X)
$\forall X$ OAK_TREE$(X) \Rightarrow$ TREES(X)
$\forall X$ PALM_TREE$(X) \Rightarrow$ TREES(X)
$\forall X$ PINE_TREE$(X) \Rightarrow$ TREES(X)
$\forall X$ WILLOW_TREE$(X) \Rightarrow$ TREE(X)

$\forall X$ VEHICLE 1$(X) \Rightarrow$ BIKE$(X) \vee$ BUS$(X) \vee$ MOTORBIKE$(X) \vee$PICKUP_TRUCK$(X) \vee$ TRAIN(X)
$\forall X$ BIKE$(X) \Rightarrow$ VEHICLE 1(X)
$\forall X$ BUS$(X) \Rightarrow$ VEHICLE 1(X)
$\forall X$ MOTORBIKE$(X) \Rightarrow$ VEHICLE 1(X)
$\forall X$ PICKUP$(X) \Rightarrow$ VEHICLE 1(X)
$\forall X$ TRAIN$(X) \Rightarrow$ VEHICLES 1(X)

$\forall X$ VEHICLES 2$(X) \Rightarrow$ LAWN MOWER$(X) \vee$ ROCKET$(X) \vee$ STREETCAR$(X) \vee$ TANK$(X) \vee$ TRACTOR(X)
$\forall X$ LAWN MOWER$(X) \Rightarrow$ VEHICLES 2(X)
$\forall X$ ROCKET$(X) \Rightarrow$ VEHICLES 2(X)
$\forall X$ STREETCAR$(X) \Rightarrow$ VEHICLES 2(X)
$\forall X$ TANK$(X) \Rightarrow$ VEHICLES 2(X)
$\forall X$ TRACTOR$(X) \Rightarrow$ VEHICLES 2(X)

HAND CRAFTED RULES
$\forall X$ MAMMALS $(X) \Rightarrow$ LARGE CARNIVORES$(X) \vee$ OMNIVORES AND HERBIVORES(X) $\vee$ MEDIUM MAMMALS$(X) \vee$ SMALL MAMMALS$(X) \vee$ AQUATIC MAMMALS(X) $\forall X$ LARGE CARNIVORES$(X) \Rightarrow$ MAMMALS(X) $\forall X$ OMNIVORES AND HERBIVORES$(X) \Rightarrow$ MAMMALS(X) $\forall X$ MEDIUM MAMMALS$(X) \Rightarrow$ MAMMALS(X) $\forall X$ SMALL MAMMALS$(X) \Rightarrow$ MAMMALS(X) $\forall X$ AQUATIC MAMMALS$(X) \Rightarrow$ MAMMALS(X) $\forall X$ HOUSEHOLD$(X) \Rightarrow$ HOUSEHOLD_ELECTRICAL_DEVICES$(X) \vee$ HOUSEHOLD_FURNITURE(X) $\vee$ FOOD_CONTAINERS(X) $\forall X$ HOUSEHOLD_ELECTRICAL_DEVICES$(X) \Rightarrow$ HOUSEHOLD(X) $\forall X$ HOUSEHOLD_FURNITURE$(X) \Rightarrow$ HOUSEHOLD(X) $\forall X$ FOOD_CONTAINERS$(X) \Rightarrow$ HOUSEHOLD(X) $\forall X$ OUTDOOR$(X) \Rightarrow$ LARGE_MAN-MADE_OUTDOOR_THINGS(X) $\vee$ LARGE_NATURAL_OUTDOOR_SCENES$(X) \vee$ TREES(X) $\forall X$ LARGE_MAN-MADE_OUTDOOR_THINGS $(X) \Rightarrow$ OUTDOOR(X) $\forall X$ LARGE_NATURAL_OUTDOOR_SCENES$(X) \Rightarrow$ OUTDOOR(X) $\forall X$ TREES$(X) \Rightarrow$ OUTDOOR(X) $\forall X$ TRANSPORT$(X) \Rightarrow$ VEHICLES_1$(X) \vee$ VEHICLES_2(X) $\forall X$ VEHICLES_1$(X) \Rightarrow$ TRANSPORT(X) $\forall X$ VEHICLES_2$(X) \Rightarrow$ TRANSPORT(X)

$\forall X$ HAS WHEELS$(X) \Rightarrow$ BICYCLE$(X) \vee$ BUS$(X) \vee$ MOTORCYCLE$(X) \vee$ PICKUP TRUCK$(X) \vee$ TRAIN(X)
$\vee$ LAWN MOWER$(X) \vee$ STREETCAR$(X) \vee$ TANK$(X) \vee$ TRACTOR(X)
$\forall X$ BICYCLE$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ BUS$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ MOTORCYCLE$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ PICKUP TRUCK$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ TRAIN$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ LAWN MOWER$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ STREETCAR$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ TANK$(X) \Rightarrow$ HAS WHEELS(X)
$\forall X$ TRACTOR$(X) \Rightarrow$ HAS WHEELS(X)

Results

Table 62 and Table 63 compares the collective classification error rates in a constant weight based SBR and in a vanilla SBR in non transductive learning respectively using different t-norms for 150 rules created with the hierarchical information of the super and the fine classes. Res, PRes, APN, MPN, WRes, WRes-D, WRes-D-RE and WRes-D-C refers to Resnets, Pre-Activated Resnets, Additive Pyramidal Resnets, Multiplicative Pyramidal Resnets, Wide Resnets, Wide Resnets with dropout, Wide Resnets with dropout and random erasing and Wide Resnets with dropout and cut outs respectively. Weak-Lukasiewicz, Lukasiewicz, Product, Minimum t-norms and Combination and Half-Weight Combination (different permutations of the t-norms mentioned earlier) are used to inject logical constraints into the CNN networks integrated in the SBR. They are abbreviated as CC-WL, CC-L, CC-P, CC-M, CC-C and CC-HW respectively in constant SBR and CC-WL(λ_l^k), CC-L(λ_l^k), CC-P(λ_l^k), CC-M(λ_l^k), CC-C(λ_l^k) and CC-HW(λ_l^k) in vanilla SBR. Table 64 and Table 65 compares the collective classification error rates using scarce training data in non transductive mode in a constant weight based and in a vanilla SBR respectively. All the abbreviations in these tables are identical to the ones described above. Table 66 and Table 68 demonstrates the classification error rates whereas Table 67 and Table 69 demonstrates the collective classification error rates when exploiting fine and coarse classes in transductive mode for different selection of t-norms with constant λ_l and variable λ_l^k SBR. Table 70 and Table 71 demonstrates the collective classification error rates for different deep architectures over 125 network outputs (the coarse and fine classes plus the additional 5 classes) using different selection of t-norms with constant and variable regularizers in non transductive mode. Table 72 and Table 74 demonstrates the classification error rates whereas Table 73 and Table 75 refers to the identical experimental settings as in Table 70 and Table 71 respectively with 125 output classes but in transductive mode.

Table 62: Error rate for the 100 final classes on CIFAR-100 for different deep architectures in non transductive mode with collective classification using different selection of t-norms with constant λ_l.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-32	29.47	28.23	28.23	28.71	29.26	28.23	28.43
Res-110	25.16	23.46	23.47	23.56	24.02	23.45	23.45
Res-164	25.16	23.79	23.81	24.0	24.04	23.77	23.78
PRes-164	24.33	22.44	22.44	22.67	23.09	22.43	22.42
APN-110,α=84	20.21	19.88	19.88	19.94	20.0	19.85	19.85
APN-110, α=270	18.25	18.04	18.04	18.20	18.23	18.04	18.02
APN-164, α=84	18.32	18.11	18.11	18.26	18.27	18.10	18.09
APN-164, α=270	17.01	16.89	16.89	16.91	17.01	16.89	16.90
APN-236, α=220	16.37	16.21	16.22	16.33	16.37	16.21	16.20
APN-272, α=200	16.35	16.05	16.05	16.24	16.32	16.04	16.04
MPN-110, α=27	18.79	18.47	18.49	18.58	18.62	18.49	18.5
WRes-16, γ=8	22.07	21.79	21.79	21.88	22.01	21.78	21.79
WRes-28, γ=10	20.50	20.30	20.30	20.40	20.50	20.30	20.30
WRes-40, γ=4	22.89	21.49	21.49	21.8	22.05	21.45	21.44
WRes-D-28, γ=10	20.04	19.98	19.98	20.0	20.04	19.96	19.95
WRes-D-RE-28, γ=10	17.73	16.42	16.44	16.49	16.52	16.44	16.44
WRes-D-C-28, γ=10	15.2	**14.96**	14.96	15.0	15.2	14.96	**14.96**

Table 63: Error rate for the 100 final classes on CIFAR-100 for different deep architectures in non transductive mode with collective classification over the network outputs using different selection of t-norms with variable λ_l^k.

Models	CNN	CC-WL(λ_l^k)	CC-L(λ_l^k)	CC-P(λ_l^k)	CC-M(λ_l^k)	CC-C(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.47	28.04	28.04	28.58	29.22	28.04	28.04
Res-110	25.16	22.76	22.76	22.91	23.54	22.74	22.75
Res-164	25.16	23.46	23.45	23.89	23.98	23.7	23.7
PRes-164	24.33	22.03	22.02	22.34	23.0	22.03	22.0
APN-110, α=84	20.21	19.83	19.83	19.91	19.98	19.8	19.8
APN-110, α=270	18.25	18.01	18.01	18.17	18.31	18.02	18.02
APN-164, α=84	18.32	18.09	18.07	18.19	18.23	18.07	18.07
APN-164, α=270	17.01	16.75	16.72	16.88	16.90	16.73	16.75
APN-236, α=220	16.37	16.01	16.01	16.33	16.33	16.0	16.12
APN-272, α=200	16.35	16.02	16.02	16.22	16.32	16.02	16.02
MPN-110, α=27	18.79	18.41	18.41	18.51	18.62	18.41	18.45
WRes-16, γ=8	22.07	21.75	21.75	21.82	22.01	21.74	21.74
WRes-28, γ=10	20.50	19.94	19.94	19.97	20.1	19.9	19.9
WRes-40, γ=4	22.89	21.42	21.44	21.7	21.99	21.42	21.42
WRes-D-28, γ=10	20.04	19.84	19.84	19.96	19.98	19.85	19.85
WRes-D-RE-28,γ=10	17.73	16.41	16.41	16.44	16.47	16.41	16.40
WRes-D-C-28, γ=10	15.2	**14.81**	14.81	14.99	14.99	14.80	**14.79**

Table 64: Comparison of the collective classification error rate for the 100 final classes using scarce training data in non transductive mode with constant SBR.

Models	%Data	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-32	10	43.44	41.57	41.58	41.66	41.69	41.56	41.60
	20	39.27	38.18	38.19	38.19	38.19	38.17	38.19
	50	34.56	33.91	33.91	33.93	33.93	33.90	33.92
PRes-164	10	40.41	40.05	40.1	40.2	40.2	40.05	40.1
	20	37.59	37.49	37.50	37.50	37.50	37.49	37.49
	50	31.34	31.31	31.30	31.32	31.32	31.30	31.32
APN-236, $\alpha = 220$	10	31.85	30.57	30.67	30.80	30.70	30.50	30.70
	20	28.16	27.41	27.40	27.40	27.42	27.40	27.40
	50	20.19	19.56	19.6	19.61	19.61	19.52	19.55
APN-272, $\alpha = 200$	10	30.85	29.78	29.79	29.80	29.79	29.80	29.80
	20	27.56	26.41	26.40	26.40	26.41	26.39	26.40
	50	19.77	18.56	18.57	18.59	18.59	18.56	18.56
WRes-28, $\gamma = 10$	10	35.85	34.78	34.79	34.80	34.79	34.78	34.80
	20	33.56	32.41	32.40	32.40	32.40	32.39	32.40
	50	24.77	23.56	23.56	23.59	23.60	23.55	23.56
WRes-D-28, $\gamma = 10$	10	33.5	32.30	32.31	32.31	32.31	32.29	32.30
	20	31.56	31.41	31.42	31.43	31.43	31.41	31.40
	50	23.77	22.56	22.57	22.59	22.59	22.56	22.59
WRes-D-RE-28, $\gamma = 10$	10	29.34	27.20	22.21	27.21	27.21	27.20	27.20
	20	27.21	26.1	26.15	26.16	26.16	26.1	26.15
	50	19.42	18.16	18.20	18.26	18.25	18.16	18.26
WRes-D-C-28, $\gamma = 10$	10	28.81	26.79	26.87	26.80	26.80	26.80	26.81
	20	26.24	25.51	25.52	25.50	25.51	25.50	25.51
	50	18.77	**17.23**	17.23	17.27	17.25	17.23	**17.14**

Table 65: Comparison of the collective classification error rate for the 100 final classes using scarce training data in non transductive mode with vanilla SBR.

Models	%Data	CNN	CC-WLλ_l^k	CC-Lλ_l^k	CC-Pλ_l^k	CC-Mλ_l^k	CC-Cλ_l^k	CC-HWλ_l^k
Res-32	10	43.44	40.14	40.15	40.15	40.15	40.14	40.14
	20	39.27	37.44	37.45	37.45	37.45	37.44	37.46
	50	34.56	30.99	30.99	31.0	30.99	30.99	30.99
PRes-164	10	40.41	38.12	38.15	38.14	38.18	38.14	38.18
	20	37.59	35.98	35.99	35.99	35.99	35.99	35.97
	50	31.34	29.94	29.95	29.97	29.98	29.94	29.95
APN-236, $\alpha = 220$	10	31.85	29.78	29.78	29.80	29.80	29.80	29.80
	20	28.16	26.41	26.41	26.40	26.41	26.39	26.39
	50	20.19	18.56	18.59	18.59	18.59	18.56	18.56
APN-272, $\alpha = 200$	10	30.85	27.78	27.77	27.80	27.80	27.77	27.79
	20	27.56	23.41	23.41	23.40	23.40	23.39	23.39
	50	19.77	17.56	17.57	17.59	17.59	17.54	17.57
WRes-28, $\gamma = 10$	10	35.85	33.78	33.79	33.80	33.80	33.78	33.80
	20	33.56	31.41	31.42	31.42	31.42	31.40	31.40
	50	24.77	22.19	22.19	22.20	22.20	22.19	22.19
WRes-D-28, $\gamma = 10$	10	33.5	29.70	29.70	29.71	29.70	29.69	29.70
	20	31.56	28.41	28.41	28.47	28.47	28.40	28.40
	50	23.77	21.47	21.48	21.47	21.48	21.47	21.47
WRes-D-RE-28, $\gamma = 10$	10	29.34	26.90	26.91	26.90	26.90	26.90	26.90
	20	27.21	25.41	25.40	25.40	25.41	25.39	25.40
	50	19.42	17.56	17.63	17.59	17.59	17.55	17.56
WRes-D-C-28, $\gamma = 10$	10	28.81	26.88	26.89	26.89	26.89	26.86	26.89
	20	26.24	23.68	23.70	23.70	23.70	23.62	23.68
	50	18.77	16.61	16.63	16.69	16.65	**16.60**	16.61

Table 66: Classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with constant λ_l in transductive mode.

Models	CNN	WL	L	P	M	C	HW
Res-32	29.47	26.23	26.22	26.23	26.26	26.16	26.13
Res-110	25.16	23.40	23.45	23.46	23.49	23.44	23.24
Res-164	25.16	22.81	22.82	22.84	22.87	22.81	22.81
PRes-164	24.33	21.82	21.84	21.84	21.84	21.72	21.63
APN-110,α=84	20.21	19.10	19.11	19.11	19.10	19.10	19.05
APN-110, α=270	18.25	17.01	17.05	17.04	17.05	17.04	17.01
APN-164, α=84	18.32	16.31	16.35	16.36	16.37	16.29	16.21
APN-164, α=270	17.01	16.10	16.19	16.11	16.11	16.19	16.08
APN-236, α=220	16.37	15.17	15.18	15.18	15.18	15.18	15.17
APN-272, α=200	16.35	14.67	14.69	14.69	14.69	14.64	14.62
MPN-110, α=27	18.79	17.09	17.09	17.18	17.18	17.10	17.05
WRes-16, γ=8	22.07	21.04	21.10	21.14	21.14	20.99	20.92
WRes-28, γ=10	20.50	19.23	19.23	19.24	19.25	19.23	19.23
WRes-40, γ=4	22.89	20.55	20.54	20.57	20.59	20.54	20.57
WRes-D-28, γ=10	20.04	19.18	19.18	19.20	19.22	19.26	19.11
WRes-D-RE-28, γ=10	17.73	15.22	15.21	15.29	15.22	15.24	15.24
WRes-D-C-28, γ=10	15.20	**13.90**	14.11	14.11	14.12	**13.90**	**13.80**

Table 67: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with constant λ_l in transductive mode.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-32	29.47	26.03	26.02	26.03	26.06	26.01	26.03
Res-110	25.16	22.90	22.91	22.92	23.02	22.90	22.92
Res-164	25.16	22.59	22.60	22.61	22.64	22.59	22.59
PRes-164	24.33	21.39	21.40	21.40	21.40	21.39	21.40
APN-110,α=84	20.21	18.82	18.83	18.84	19.01	18.82	18.85
APN-110, α=270	18.25	16.74	16.75	16.74	16.75	16.74	16.72
APN-164, α=84	18.32	16.01	16.0	16.06	16.07	16.0	16.01
APN-164, α=270	17.01	15.80	15.79	15.81	15.81	15.79	15.8
APN-236, α=220	16.37	14.97	14.98	14.98	14.98	14.98	14.97
APN-272, α=200	16.35	14.52	14.53	14.54	14.54	14.52	14.54
MPN-110, α=27	18.79	16.79	16.79	16.8	16.82	16.79	16.79
WRes-16, γ=8	22.07	20.91	20.91	20.94	20.94	20.90	20.92
WRes-28, γ=10	20.50	19.03	19.03	19.04	19.05	19.03	19.03
WRes-40, γ=4	22.89	20.40	20.39	20.41	20.45	20.40	20.44
WRes-D-28, γ=10	20.04	18.98	18.98	19.0	19.02	18.96	18.97
WRes-D-RE-28, γ=10	17.73	15.12	15.11	15.19	15.12	15.14	15.14
WRes-D-C-28, γ=10	15.20	**13.60**	13.61	13.61	13.62	**13.60**	**13.60**

Table 68: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with variable λ_l^k in transductive mode.

Models	CNN	WL(λ_l^k)	L(λ_l^k)	P(λ_l^k)	M(λ_l^k)	C(λ_l^k)	HW(λ_l^k)
Res-32	29.47	25.94	25.94	25.98	25.99	25.94	25.94
Res-110	25.16	21.92	21.93	21.91	21.94	21.84	21.85
Res-164	25.16	21.62	21.64	21.69	21.68	21.61	21.61
PRes-164	24.33	20.73	20.82	20.84	20.76	20.73	20.74
APN-110, α=84	20.21	18.13	18.13	18.19	18.19	18.08	18.08
APN-110, α=270	18.25	16.51	16.51	16.57	16.51	16.51	16.51
APN-164, α=84	18.32	15.89	15.87	15.89	15.93	15.87	15.87
APN-164, α=270	17.01	15.05	15.07	15.08	15.09	15.07	15.07
APN-236, α=220	16.37	13.81	13.81	13.83	13.83	13.80	13.82
APN-272, α=200	16.35	13.12	13.12	13.22	13.22	13.12	13.12
MPN-110, α=27	18.79	15.91	15.91	16.05	15.96	15.91	15.95
WRes-16, γ=8	22.07	20.95	20.95	20.99	20.98	20.94	20.94
WRes-28, γ=10	20.50	18.14	18.14	18.17	18.21	18.21	18.10
WRes-40, γ=4	22.89	19.82	19.84	19.87	19.89	19.82	19.82
WRes-D-28, γ=10	20.04	19.04	19.04	19.06	19.08	19.05	19.05
WRes-D-RE-28,γ=10	17.73	15.11	15.11	15.14	15.17	15.12	15.11
WRes-D-C-28, γ=10	15.20	**13.21**	13.21	13.29	13.29	13.21	**13.19**

Table 69: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with variable λ_l^k in transductive mode.

Models	CNN	CC-WL(λ_l^k)	CC-L(λ_l^k)	CC-P(λ_l^k)	CC-M(λ_l^k)	CC-C(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.47	25.14	25.14	25.18	25.22	25.14	25.14
Res-110	25.16	21.72	21.73	21.81	21.84	21.74	21.75
Res-164	25.16	21.02	21.04	21.09	21.08	21.01	21.07
PRes-164	24.33	20.03	20.02	20.04	20.06	20.03	20.04
APN-110, α=84	20.21	17.83	17.83	17.89	17.89	17.8	17.8
APN-110, α=270	18.25	16.01	16.01	16.07	16.01	16.01	16.01
APN-164, α=84	18.32	15.09	15.07	15.09	15.13	15.07	15.07
APN-164, α=270	17.01	14.75	14.72	14.88	14.90	14.73	14.75
APN-236, α=220	16.37	13.01	13.01	13.03	13.03	13.0	13.02
APN-272, α=200	16.35	12.92	12.92	13.02	13.02	12.92	12.92
MPN-110, α=27	18.79	15.41	15.41	15.50	15.60	15.41	15.45
WRes-16, γ=8	22.07	20.75	20.75	20.82	20.80	20.74	20.74
WRes-28, γ=10	20.50	17.94	17.94	17.97	18.1	17.9	17.90
WRes-40, γ=4	22.89	19.42	19.44	19.47	19.49	19.42	19.42
WRes-D-28, γ=10	20.04	18.84	18.84	18.96	18.98	18.85	18.85
WRes-D-RE-28,γ=10	17.73	15.01	15.01	15.04	15.07	15.01	15.04
WRes-D-C-28, γ=10	15.20	**12.81**	12.81	12.99	12.99	12.81	**12.79**

Table 70: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures over 125 network outputs using different selection of t-norms with constant λ_l in non transductive mode.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-32	29.47	27.5	27.5	28.2	28.81	27.5	27.44
Res-110	25.16	22.80	22.82	23.01	23.94	22.81	22.81
Res-164	25.16	22.81	22.82	23.94	23.98	22.81	22.80
PRes-164	24.33	21.65	21.64	21.83	22.34	22.65	22.64
APN-110,α=84	20.21	19.01	19.01	19.11	19.24	19.15	19.15
APN-110, α=270	18.25	17.22	17.22	17.51	17.64	17.22	17.21
APN-164, α=84	18.32	17.34	17.34	17.66	17.99	17.33	17.33
APN-164, α=270	17.01	16.05	16.05	16.22	16.28	16.04	16.04
APN-236, α=220	16.37	15.40	15.41	15.53	15.57	15.41	15.40
APN-272, α=200	16.35	15.35	15.35	15.42	15.52	15.34	15.34
MPN-110, α=27	18.79	17.74	17.72	17.83	17.92	17.72	17.70
WRes-16, γ=8	22.07	21.0	21.0	21.04	21.23	21.0	21.01
WRes-28, γ=10	20.5	19.6	19.7	19.8	19.9	19.7	19.7
WRes-40, γ=4	22.89	20.74	20.74	21.0	21.2	20.71	20.71
WRes-D-28, γ=10	20.04	19.16	19.16	19.2	19.3	19.16	19.15
WRes-D-RE-28, γ=10	17.73	15.67	15.67	15.70	15.72	15.67	15.67
WRes-D-C-28, γ=10	15.2	14.24	14.24	14.30	14.32	14.2	**14.14**

Table 71: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures over 125 network outputs using different selection of t-norms with variable λ_l^k in non transductive mode.

Models	CNN	CC-WL(λ_l^k)	CC-L(λ_l^k)	CC-P(λ_l^k)	CC-M(λ_l^k)	CC-C(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.47	27.2	27.2	28.04	28.44	27.2	27.0
Res-110	25.16	22.68	22.68	22.90	23.50	22.65	22.65
Res-164	25.16	22.65	22.65	23.05	23.56	22.65	22.65
PRes-164	24.33	21.20	21.20	21.55	22.28	21.21	21.21
APN-110, α=84	20.21	18.99	18.99	19.01	19.18	19.0	19.0
APN-110, α=270	18.25	17.11	17.11	17.23	17.31	17.11	17.11
APN-164, α=84	18.32	17.16	17.16	17.25	17.25	17.16	17.16
APN-164, α=270	17.01	15.98	15.98	16.01	16.14	15.98	15.98
APN-236, α=220	16.37	15.20	15.20	15.53	15.53	15.2	15.20
APN-272, α=200	16.35	15.07	15.07	15.86	15.72	15.02	15.02
MPN-110, α=27	18.79	17.40	17.38	17.61	17.83	17.38	17.35
WRes-16, γ=8	22.07	20.94	20.93	21.0	21.20	20.90	20.94
WRes-28, γ=10	20.5	19.22	19.24	19.37	19.40	19.24	19.24
WRes-40, γ=4	22.89	20.66	20.65	20.82	21.0	20.46	20.46
WRes-D-28, γ=10	20.04	19.02	19.02	19.14	19.18	19.02	19.0
WRes-D-RE-28,γ=10	17.73	15.48	15.48	15.66	15.68	15.47	15.47
WRes-D-C-28, γ=10	15.20	14.05	14.10	14.14	14.14	**14.0**	**14.0**

Table 72: Classification error rates for the 100 final classes with 125 outputs from different deep architectures using different selection of t-norms with constant λ_l in transductive mode.

Models	CNN	WL	L	P	M	C	HW
Res-32	29.47	25.81	25.79	25.81	25.82	25.78	25.74
Res-110	25.16	22.68	22.69	22.69	22.69	22.68	22.64
Res-164	25.16	22.26	22.26	22.26	22.36	21.18	21.07
PRes-164	24.33	21.53	21.54	21.54	21.54	21.49	21.44
APN-110,α=84	20.21	18.91	18.93	18.94	18.89	18.82	18.60
APN-110, α=270	18.25	16.51	16.55	16.54	16.55	16.44	16.42
APN-164, α=84	18.32	16.04	16.10	16.16	16.17	16.05	16.04
APN-164, α=270	17.01	15.80	15.79	15.71	15.71	15.69	15.38
APN-236, α=220	16.37	14.91	14.90	14.91	14.91	14.87	14.84
APN-272, α=200	16.35	14.41	14.40	14.40	14.39	14.37	14.34
MPN-110, α=27	18.79	16.73	16.70	16.72	16.71	16.67	16.65
WRes-16, γ=8	22.07	20.80	20.81	20.80	20.79	20.74	20.74
WRes-28, γ=10	20.50	19.03	19.04	19.04	19.05	19.03	18.93
WRes-40, γ=4	22.89	20.41	20.43	20.44	20.40	20.32	20.27
WRes-D-28, γ=10	20.04	18.62	18.62	18.60	18.57	18.51	18.50
WRes-D-RE-28, γ=10	17.73	15.02	15.01	15.02	15.02	14.97	14.92
WRes-D-C-28, γ=10	15.20	13.55	13.56	13.55	13.56	13.48	**13.41**

Table 73: Collective classification error rates for the 100 final classes with 125 outputs from different deep architectures using different selection of t-norms with constant λ_l in transductive mode.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-32	29.47	25.38	25.39	25.41	25.46	25.21	25.20
Res-110	25.16	22.06	22.10	22.09	22.94	22.02	22.02
Res-164	25.16	22.05	22.06	22.06	22.16	21.94	21.95
PRes-164	24.33	21.03	21.04	21.04	21.04	21.03	21.04
APN-110,α=84	20.21	18.61	18.63	18.64	18.81	18.62	18.60
APN-110, α=270	18.25	16.11	16.15	16.14	16.15	16.14	16.12
APN-164, α=84	18.32	15.60	15.60	15.66	15.67	15.60	15.58
APN-164, α=270	17.01	15.40	15.39	15.41	15.41	15.39	15.38
APN-236, α=220	16.37	14.71	14.78	14.78	14.78	14.70	14.70
APN-272, α=200	16.35	14.13	14.13	14.14	14.14	14.12	14.11
MPN-110, α=27	18.79	16.27	16.27	16.28	16.29	16.27	16.25
WRes-16, γ=8	22.07	20.30	20.31	20.34	20.34	20.30	20.29
WRes-28, γ=10	20.50	18.73	18.74	18.84	18.85	18.83	18.83
WRes-40, γ=4	22.89	20.11	20.13	20.14	20.15	20.10	20.09
WRes-D-28, γ=10	20.04	18.12	18.12	18.10	18.62	18.11	18.10
WRes-D-RE-28, γ=10	17.73	14.92	14.91	14.92	14.92	14.91	14.90
WRes-D-C-28, γ=10	15.20	13.05	13.06	13.05	13.06	13.04	**13.01**

Table 74: Classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with variable λ_l^k in transductive mode.

Models	CNN	WL(λ_l^k)	L(λ_l^k)	P(λ_l^k)	M(λ_l^k)	C(λ_l^k)	HW(λ_l^k)
Res-32	29.47	25.21	25.24	25.22	25.22	25.24	25.28
Res-110	25.16	21.62	21.63	21.61	21.64	21.62	21.68
Res-164	25.16	21.02	21.04	21.09	21.08	21.01	21.01
PRes-164	24.33	20.28	20.29	20.29	20.29	20.28	20.28
APN-110, α=84	20.21	17.91	17.93	17.99	17.99	17.88	17.88
APN-110, α=270	18.25	15.84	15.85	15.87	15.91	15.91	15.90
APN-164, α=84	18.32	15.39	15.37	15.39	15.38	15.37	15.37
APN-164, α=270	17.01	15.05	15.02	15.08	15.10	15.13	15.10
APN-236, α=220	16.37	13.33	13.31	13.33	13.33	13.40	13.42
APN-272, α=200	16.35	13.10	13.12	13.12	13.12	13.11	13.12
MPN-110, α=27	18.79	15.61	15.61	15.60	15.60	15.61	15.55
WRes-16, γ=8	22.07	20.57	20.57	20.58	20.58	20.54	20.57
WRes-28, γ=10	20.50	17.84	17.89	17.87	17.91	17.86	17.85
WRes-40, γ=4	22.89	19.82	19.84	19.84	19.84	19.89	19.88
WRes-D-28, γ=10	20.04	18.54	18.54	18.56	18.58	18.50	18.52
WRes-D-RE-28,γ=10	17.73	14.89	14.89	14.84	14.87	14.89	14.89
WRes-D-C-28, γ=10	15.20	12.90	12.91	12.99	12.99	**12.98**	**12.98**

Table 75: Collective classification error rate for the 100 final classes on CIFAR-100 for different deep architectures using different selection of t-norms with variable λ_l^k in transductive mode.

Models	CNN	CC-WL(λ_l^k)	CC-L(λ_l^k)	CC-P(λ_l^k)	CC-M(λ_l^k)	CC-C(λ_l^k)	CC-HW(λ_l^k)
Res-32	29.47	24.91	25.04	25.02	25.02	24.84	24.84
Res-110	25.16	21.22	21.23	21.31	21.34	21.22	21.20
Res-164	25.16	20.62	20.64	20.69	20.68	20.61	20.57
PRes-164	24.33	19.88	19.89	19.94	19.96	19.83	19.84
APN-110, α=84	20.21	17.16	17.13	17.19	17.19	17.08	17.08
APN-110, α=270	18.25	15.44	15.45	15.47	15.51	15.41	15.40
APN-164, α=84	18.32	14.79	14.77	14.79	14.83	14.77	14.77
APN-164, α=270	17.01	14.25	14.22	14.28	14.30	14.23	14.20
APN-236, α=220	16.37	12.67	12.61	12.63	12.63	12.60	13.62
APN-272, α=200	16.35	12.12	12.12	12.42	12.42	12.11	12.02
MPN-110, α=27	18.79	14.41	14.41	14.50	14.60	14.41	14.45
WRes-16, γ=8	22.07	20.37	20.37	20.38	20.38	20.34	20.34
WRes-28, γ=10	20.50	17.14	17.19	17.17	17.61	17.16	17.15
WRes-40, γ=4	22.89	19.12	19.14	19.14	19.14	19.12	19.12
WRes-D-28, γ=10	20.04	18.34	18.34	18.36	18.38	18.30	18.25
WRes-D-RE-28,γ=10	17.73	14.81	14.81	14.84	14.87	14.81	14.84
WRes-D-C-28, γ=10	15.20	12.10	12.11	12.19	12.19	**12.08**	**12.08**

A.0.4 Experimental Analysis on ImageNet Dataset

Results

Res, PRes, APN, MPN, WRes, WRes-D, WRes-D-RE and WRes-D-C refers to Resnets, Pre-Activated Resnets, Additive Pyramidal Resnets, Multiplicative Pyramidal Resnets, Wide Resnets, Wide Resnets with dropout, Wide Resnets with dropout and random erasing and Wide Resnets with dropout and cutouts respectively. Weak-Lukasiewicz, Lukasiewicz, Product, Minimum t-norms and Combination and Half-Weight-Combination (different permutations of the t-norms mentioned earlier) are used to inject logical constraints into the CNN networks integrated in the SBR.

They are abbreviated as CC-WL, CC-L, CC-P, CC-M, CC-C and CC-HW respectively in constant SBR and CC-WL(λ_l^k), CC-P(λ_l^k), CC-M(λ_l^k), CC-C(λ_l^k) and CC-HW(λ_l^k) in vanilla SBR.

Table 76 and Table 77 represents the classification error rates with different t-norms for 1000 classes on ImageNet test set for different deep architectures in a constant weight based and vanilla SBR respectively using 26 intermediate classes. Table 78 and Table 79 compares the collective classification error rates for different t-norm selections using scarce training data in non transductive mode with constant λ_l and variable λ_l^k settings.

Table 80and Table 81 represents the classification error rates with different t-norms for different deep architectures in a constant weight based and vanilla SBR respectively using 581 intermediate classes and 4 levels of hierarchy. Table 82 and Table 83 compares the collective classification error rates for different t-norm selections using scarce training data and 581 intermediate predicates in non transductive mode with constant λ_l and variable λ_l^k settings.

Table 84and Table 85 represents the classification error rates with different t-norms for different deep architectures in a constant weight based and vanilla SBR respectively using 1605 intermediate classes and all levels of hierarchy. Table 86 compares the collective classification error rates for different t-norm selections using scarce training data and 1605 intermediate predicates.

Table 76: Classification error rate for 1000 classes on ImageNet for different deep architectures in a constant weight based SBR using 26 intermediate classes.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	24.70	24.46	24.49	24.56	24.56	24.48	24.49
Res-101	23.60	23.57	23.58	23.58	23.59	23.56	23.57
Res-152	23.0	22.80	22.82	22.95	22.95	22.82	22.80
PRes-152	22.20	22.05	22.05	22.14	22.15	22.05	22.06
PRes-200	21.90	21.78	21.78	21.80	21.83	21.80	21.78
APN-200 $\alpha = 300$	20.50	20.14	20.14	20.20	20.23	20.16	20.17
APN-200 $\alpha = 450$	20.10	**19.96**	20.06	20.08	20.09	**19.96**	**19.96**

Table 77: Classification error rate for 1000 classes on ImageNet for different deep architectures in a vanilla SBR using 26 intermediate classes.

Models	CNN	CC-WL (λ_l^k)	CC-L (λ_l^k)	CC-P (λ_l^k)	CC-M (λ_l^k)	CC-C (λ_l^k)	CC-HW (λ_l^k)
Res-50	24.70	24.41	24.43	24.50	24.51	24.40	24.41
Res-101	23.60	23.52	23.55	23.56	23.57	23.52	23.52
Res-152	23.0	22.75	22.76	22.80	22.80	22.75	22.76
PRes-152	22.20	22.0	22.0	22.11	22.11	22.0	22.01
PRes-200	21.90	21.73	21.73	21.76	21.79	21.73	21.71
APN-200 $\alpha = 300$	20.50	20.03	20.03	20.17	20.13	20.04	20.07
APN-200 $\alpha = 450$	20.10	**19.94**	20.04	20.04	20.06	**19.90**	**19.90**

Table 78: Comparison of the collective classification error rate for 1000 final classes of ImageNet using scarce training data in non transductive mode with constant λ_l values.

Models	%Data	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	10	36.09	35.24	35.27	35.27	35.27	35.20	35.20
	20	34.2	33.01	33.07	33.05	33.04	33.01	33.02
	50	30.10	28.59	28.55	28.88	28.57	28.57	28.52
Res-101	10	35.13	33.94	33.91	33.87	33.87	33.92	33.90
	20	33.7	31.93	31.91	31.91	31.92	31.90	31.90
	50	28.46	25.83	25.83	25.85	25.84	25.83	25.82
Res-152	10	34.62	33.1	33.15	33.15	33.15	33.15	33.15
	20	32.61	31.19	31.19	31.2	31.19	31.18	31.18
	50	26.14	25.99	25.99	26.01	25.99	25.94	25.94
PRes-152	10	33.36	32.12	32.14	32.24	32.18	32.14	32.12
	20	31.02	31.98	31.99	31.99	31.97	31.99	31.97
	50	26.08	24.94	24.95	24.97	24.95	24.91	24.89
PRes-200	10	33.19	32.81	32.78	32.85	32.89	32.84	31.79
	20	30.39	29.29	29.29	29.31	29.39	29.29	29.28
	50	25.88	25.12	25.14	25.17	25.15	25.11	25.11
APN-200, $\alpha = 300$	10	32.85	32.78	32.77	32.80	32.80	32.78	32.76
	20	29.56	29.11	29.20	29.20	29.20	29.11	29.13
	50	25.77	24.88	24.87	24.89	24.95	24.86	24.86
APN-200, $\alpha = 450$	10	32.75	31.17	31.17	31.18	31.18	31.16	31.15
	20	29.22	28.04	28.04	28.04	28.04	28.04	28.03
	50	24.97	**24.16**	24.17	24.19	24.16	**24.15**	**24.15**

Table 79: Comparison of the collective classification error rate for 1000 final classes of ImageNet using scarce training data in non transductive mode with variable λ_l values.

Models	%Data	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	10	36.09	34.74	34.76	34.77	34.78	34.70	34.72
	20	34.2	32.44	32.47	32.45	32.44	32.41	32.42
	50	30.10	27.99	27.95	28.0	27.97	27.97	27.99
Res-101	10	35.13	33.44	33.41	33.47	33.47	33.42	33.40
	20	33.7	31.23	31.21	31.31	31.32	31.20	31.20
	50	28.46	25.32	25.31	25.35	25.41	25.30	25.32
Res-152	10	34.62	32.51	32.57	32.55	32.55	32.50	32.52
	20	32.61	30.89	30.95	30.92	30.90	30.85	30.81
	50	26.14	24.99	24.99	25.04	24.99	24.91	24.90
PRes-152	10	33.36	32.12	32.14	32.24	32.18	32.14	32.12
	20	31.02	31.98	31.99	31.99	31.97	31.99	31.97
	50	26.08	24.94	24.95	24.97	24.95	24.91	24.89
PRes-200	10	33.19	32.03	31.9	32.05	32.09	32.04	31.91
	20	30.39	28.98	28.98	29.01	28.99	28.90	28.91
	50	25.88	24.42	24.44	24.47	24.45	24.41	24.41
APN-200, $\alpha = 300$	10	32.85	31.78	31.77	31.80	31.80	31.78	31.76
	20	29.56	28.41	28.40	28.40	28.40	28.40	28.39
	50	25.77	24.16	24.17	24.19	24.25	24.16	24.16
APN-200, $\alpha = 450$	10	32.75	30.78	30.78	30.80	30.80	30.80	30.77
	20	29.22	27.41	27.40	27.40	27.42	27.40	27.38
	50	24.97	**23.56**	23.63	23.59	23.65	**23.55**	**23.56**

Table 80: Classification error rate for 1000 classes on ImageNet for different deep architectures in a constant weight based SBR using 581 intermediate classes.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	24.70	24.40	24.41	24.46	24.46	24.38	24.39
Res-101	23.60	23.47	23.48	23.48	23.49	23.45	23.46
Res-152	23.0	22.65	22.72	22.75	22.75	22.62	22.60
PRes-152	22.20	21.85	21.85	21.84	21.85	21.80	21.80
PRes-200	21.90	21.48	21.48	21.49	21.48	21.40	21.41
APN-200 $\alpha = 300$	20.50	20.01	20.04	20.10	20.13	20.01	20.01
APN-200 $\alpha = 450$	20.10	19.56	19.6	19.8	19.79	19.56	**19.55**

Table 81: Classification error rate for 1000 classes on ImageNet for different deep architectures in a vanilla SBR using 581 intermediate classes.

Models	CNN	CC-WL (λ_l^k)	CC-L (λ_l^k)	CC-P (λ_l^k)	CC-M (λ_l^k)	CC-C (λ_l^k)	CC-HW (λ_l^k)
Res-50	24.70	24.20	24.28	24.25	24.25	24.20	24.19
Res-101	23.60	23.29	23.35	23.36	23.37	23.22	23.22
Res-152	23.0	22.15	22.16	22.20	22.20	22.15	22.16
PRes-152	22.20	21.40	21.40	21.41	21.41	21.30	21.31
PRes-200	21.90	21.23	21.23	21.26	21.29	21.23	21.21
APN-200 $\alpha = 300$	20.50	19.73	19.73	19.77	19.83	19.64	19.67
APN-200 $\alpha = 450$	20.10	19.35	19.44	19.46	19.46	**19.20**	**19.20**

Table 82: Comparison of the collective classification error rate for 1000 final classes of ImageNet using scarce training data in non transductive mode with constant λ_l values using 581 predicates.

Models	%Data	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	10	36.09	34.88	34.87	34.87	34.82	34.70	34.70
	20	34.2	32.41	32.47	32.45	32.44	32.41	32.42
	50	30.10	27.55	27.45	27.58	27.56	27.54	27.51
Res-101	10	35.13	32.99	32.91	32.97	32.97	32.92	32.90
	20	33.7	30.97	30.97	30.96	30.92	30.90	30.90
	50	28.46	24.39	24.38	24.5	24.4	24.31	24.32
Res-152	10	34.62	31.91	31.95	31.95	31.95	31.95	31.95
	20	32.61	29.19	29.19	29.20	29.19	29.18	29.18
	50	26.14	24.39	24.39	24.49	24.49	24.44	24.34
PRes-152	10	33.36	31.41	31.44	31.24	31.18	31.14	31.12
	20	31.02	28.84	28.89	28.99	30.17	28.89	28.87
	50	26.08	24.14	24.15	24.17	24.15	24.11	24.09
PRes-200	10	33.19	31.11	31.18	31.15	31.19	31.14	31.09
	20	30.39	28.89	28.89	28.91	28.99	28.79	28.78
	50	25.88	23.12	23.14	23.17	23.15	23.11	23.11
APN-200, $\alpha = 300$	10	32.85	30.78	30.77	30.80	30.80	30.78	30.76
	20	29.56	27.01	27.02	27.02	27.02	27.01	27.01
	50	25.77	23.90	23.87	23.89	23.95	23.86	23.86
APN-200, $\alpha = 450$	10	32.75	29.70	29.71	29.78	29.78	29.76	29.75
	20	29.22	26.04	26.04	26.04	26.04	26.04	26.03
	50	24.97	**21.77**	21.77	21.79	21.76	**21.75**	**21.75**

Table 83: Comparison of the collective classification error rate for 1000 final classes of ImageNet using scarce training data and 581 predicates in non transductive mode with variable λ_l^k values.

Models	%Data	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	10	36.09	33.74	33.76	33.77	33.78	33.70	33.72
	20	34.2	31.81	31.87	31.85	31.84	31.81	31.82
	50	30.10	26.94	26.95	26.97	26.97	26.97	26.91
Res-101	10	35.13	32.84	32.89	32.87	32.87	32.82	32.80
	20	33.7	30.93	30.95	30.95	30.95	30.90	30.90
	50	28.46	23.93	23.91	23.95	23.94	23.93	23.92
Res-152	10	34.62	30.51	30.57	30.55	30.55	30.50	30.50
	20	32.61	28.17	28.25	28.32	28.29	28.25	28.16
	50	26.14	23.71	23.79	23.74	23.79	23.71	23.70
PRes-152	10	33.36	31.12	31.14	31.14	31.18	31.14	31.12
	20	31.02	27.98	27.99	27.99	27.97	27.99	27.93
	50	26.08	23.90	23.95	23.97	23.95	23.91	23.89
PRes-200	10	33.19	31.03	31.04	31.05	31.09	31.04	31.01
	20	30.39	28.05	28.08	28.09	28.09	28.09	28.01
	50	25.88	22.42	22.44	22.47	22.45	22.41	22.41
APN-200, $\alpha = 300$	10	32.85	30.01	30.07	30.08	30.08	30.02	30.01
	20	29.56	26.84	26.84	26.84	26.84	26.84	26.83
	50	25.77	23.16	23.17	23.19	23.21	23.16	23.16
APN-200, $\alpha = 450$	10	32.75	28.97	28.98	28.98	28.98	28.98	28.97
	20	29.22	25.91	25.94	25.94	25.94	25.94	25.93
	50	24.97	**20.71**	20.73	20.72	20.75	**20.70**	**20.70**

Table 84: Classification error rate for 1000 classes on ImageNet for different deep architectures in a constant weight based SBR using 1605 intermediate classes.

Models	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	24.70	24.14	24.14	24.16	24.16	24.11	24.11
Res-101	23.60	23.07	23.08	23.08	23.09	23.05	23.06
Res-152	23.0	21.91	22.02	22.05	22.05	21.92	21.91
PRes-152	22.20	20.95	20.95	20.94	20.95	20.90	20.90
PRes-200	21.90	20.88	20.88	20.89	20.88	20.80	20.81
APN-200 $\alpha = 300$	20.50	19.24	19.24	19.30	19.33	19.21	19.21
APN-200 $\alpha = 450$	20.10	18.96	18.96	18.98	18.99	18.96	**18.95**

Table 85: Classification error rate for 1000 classes on ImageNet for different deep architectures in a vanilla SBR using 1605 intermediate classes.

Models	CNN	CC-WL (λ_l^k)	CC-L (λ_l^k)	CC-P (λ_l^k)	CC-M (λ_l^k)	CC-C (λ_l^k)	CC-HW (λ_l^k)
Res-50	24.70	23.80	23.84	23.85	23.85	23.80	23.81
Res-101	23.60	22.90	22.95	22.96	22.97	22.92	22.92
Res-152	23.0	21.25	21.26	21.29	21.30	21.25	21.26
PRes-152	22.20	20.91	20.91	20.91	20.91	20.90	20.90
PRes-200	21.90	20.33	20.33	20.36	20.39	20.32	20.31
APN-200 $\alpha = 300$	20.50	18.93	18.93	18.97	18.98	18.90	18.90
APN-200 $\alpha = 450$	20.10	18.14	18.14	18.16	18.16	**18.10**	**18.10**

Table 86: Comparison of the collective classification error rate for 1000 final classes of ImageNet using scarce training data in non transductive mode with constant λ_l values using 1605 intermediate predicates.

Models	%Data	CNN	CC-WL	CC-L	CC-P	CC-M	CC-C	CC-HW
Res-50	10	36.09	33.91	33.97	33.97	33.92	33.70	33.70
	20	34.2	31.16	31.17	31.15	31.14	31.14	31.12
	50	30.10	26.50	26.50	26.80	26.60	26.40	26.40
Res-101	10	35.13	32.02	32.01	32.07	32.07	32.02	32.0
	20	33.7	30.07	30.07	30.06	30.02	30.0	30.0
	50	28.46	24.03	24.08	24.05	24.04	24.01	24.01
Res-152	10	34.62	30.97	30.95	30.95	30.95	30.95	30.95
	20	32.61	28.85	28.89	28.88	28.89	28.88	28.82
	50	26.14	23.97	23.98	23.99	23.99	23.94	23.94
PRes-152	10	33.36	30.91	30.94	30.94	30.91	30.91	30.90
	20	31.02	28.07	28.08	28.09	28.91	28.09	28.01
	50	26.08	23.94	23.95	23.97	23.95	23.91	23.90
PRes-200	10	33.19	30.90	30.98	30.95	30.99	30.94	30.90
	20	30.39	27.91	27.91	27.91	27.99	27.91	27.88
	50	25.88	22.2	22.24	22.27	22.25	22.21	22.21
APN-200, $\alpha = 300$	10	32.85	29.80	29.87	29.87	29.87	29.78	29.76
	20	29.56	26.81	26.82	26.82	26.82	26.81	26.81
	50	25.77	23.10	23.17	23.19	23.15	23.16	23.16
APN-200, $\alpha = 450$	10	32.75	28.17	28.17	28.18	28.18	28.16	28.15
	20	29.22	25.54	25.54	25.54	25.54	25.54	25.53
	50	24.97	**20.92**	20.97	20.99	20.96	**20.92**	**20.92**

A.0.5 Experimental Analysis on CATARACTS Dataset

Cohen's Kappa

Cohen's Kappa (κ) is a coefficient used to measure the inter-rate agreement between two raters for categorical variables (Coh60; Coh68). It is considered to be a more robust measurement than the simple percentage of agreement, as it takes into account the possibility that agreement between the raters occurs by chance.

Given N items classified into C mutually exclusive categories by 2 independent raters, Cohen's Kappa κ can be calculated as:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{A.1}$$

where p_0 is the relative observed agreement among raters and p_e is the hypothetical probability of chance agreement. If n_{ci} is the number of times rater i predicted category c, p_e will be:

$$p_e = \frac{1}{N^2} \sum_c n_{c1} n_{c2} \tag{A.2}$$

Cohen's Kappa ranges between 0 and 1, where 0 is the agreement only resulting from chance and 1 is perfect agreement between raters. Substantial disagreement would have the κ falling in the range $[0.10, 0.60]$, substantial agreement would have κ value ranging between $[0.61, 0.80]$, and for nearly perfect agreement κ would fall in the range $[0.81, 0.99]$.

Results

In all of the tables below, the AUC-ROC score for each annotated tool by the CNN ensemble is listed using different techniques.

Table 87: Area under ROC curve for each tool using a trained CNN ensemble of 3 deep CNNs on uniform sampling of frames followed by using a median filtering model for temporal correlation on the test predictions.

Tool Name	AUC-ROC
Biomarker	0.722
Charleux cannula	0.935
Hydrodissection cannula	0.983
Rycroft cannula	0.997
Viscoelastic cannula	0.935
Cotton	0.931
Capsulorhexis cystotome	0.999
Bonn forceps	0.970
Capsulorhexis forceps	0.995
Troutman forceps	0.952
Needle holder	0.981
Aspiration handpiece	0.998
Phacoemulsifier handpiece	0.999
Vitrectomy handpiece	0.989
Implant injector	0.989
Primary incision knife	0.991
Secondary incision knife	0.998
Micromanipulator	0.997
Suture needle	0.922
Mendez ring	0.781
Vannas scissors	0.985

Table 88: Area under ROC curve for each tool using a trained CNN ensemble of 3 deep CNNs on selective sampling of frames followed by using a median filtering model for temporal correlation on the test predictions.

Tool Name	AUC-ROC
Biomarker	0.910
Charleux cannula	0.941
Hydrodissection cannula	0.980
Rycroft cannula	0.990
Viscoelastic cannula	0.930
Cotton	0.944
Capsulorhexis cystotome	0.987
Bonn forceps	0.969
Capsulorhexis forceps	0.981
Troutman forceps	0.962
Needle holder	0.979
Aspiration handpiece	0.981
Phacoemulsifier handpiece	0.997
Vitrectomy handpiece	0.988
Implant injector	0.988
Primary incision knife	0.994
Secondary incision knife	0.991
Micromanipulator	0.991
Suture needle	0.942
Mendez ring	0.947
Vannas scissors	0.981

Table 89: Area under ROC curve for each tool using a trained CNN ensemble of 4 deep CNNs on selective sampling of frames followed by using a median filtering model for temporal correlation on the test predictions.

Tool Name	AUC-ROC
Biomarker	0.932
Charleux cannula	0.966
Hydrodissection cannula	0.984
Rycroft cannula	0.991
Viscoelastic cannula	0.955
Cotton	0.966
Capsulorhexis cystotome	0.988
Bonn forceps	0.985
Capsulorhexis forceps	0.980
Troutman forceps	0.971
Needle holder	0.985
Aspiration handpiece	0.985
Phacoemulsifier handpiece	0.996
Vitrectomy handpiece	0.988
Implant injector	0.989
Primary incision knife	0.991
Secondary incision knife	0.997
Micromanipulator	0.994
Suture needle	0.962
Mendez ring	0.969
Vannas scissors	0.982

Table 90: Area under ROC curve for each tool using a trained CNN ensemble of 4 deep CNNs on selective sampling of frames followed by using a MRF model on the test predictions. The final design of the DResSys is compared with the second best solution of the challenge.

Tool Name	AUC-ROC	
	DResSys	LaTIM
Biomarker	0.999	0.985
Charleux cannula	0.989	0.984
Hydrodissection cannula	0.996	0.987
Rycroft cannula	0.998	0.995
Viscoelastic cannula	0.987	0.982
Cotton	0.999	0.998
Capsulorhexis cystotome	0.997	0.998
Bonn forceps	0.999	0.994
Capsulorhexis forceps	0.997	0.998
Troutman forceps	0.999	0.997
Needle holder	0.989	0.994
Aspiration handpiece	0.995	0.998
Phacoemulsifier handpiece	0.999	0.999
Vitrectomy handpiece	0.999	0.972
Implant injector	0.998	0.994
Primary incision knife	0.999	0.996
Secondary incision knife	0.999	0.999
Micromanipulator	0.999	0.998
Suture needle	0.999	0.999
Mendez ring	**1.000**	0.998
Vannas scissors	0.997	0.984

Table 91: Area under ROC curve for each tool using a trained CNN ensemble of 4 deep CNNs on selective sampling of frames followed by collective classification using rules in the SBR framework during inference.

Tool Name	AUC-ROC
Biomarker	0.996
Charleux cannula	0.990
Hydrodissection cannula	0.996
Rycroft cannula	0.995
Viscoelastic cannula	0.987
Cotton	0.991
Capsulorhexis cystotome	0.995
Bonn forceps	0.996
Capsulorhexis forceps	0.997
Troutman forceps	0.997
Needle holder	0.981
Aspiration handpiece	0.995
Phacoemulsifier handpiece	0.991
Vitrectomy handpiece	0.989
Implant injector	0.989
Primary incision knife	0.996
Secondary incision knife	0.999
Micromanipulator	0.996
Suture needle	0.999
Mendez ring	0.996
Vannas scissors	0.997

Appendix B

Chapter 7: Contaminants Separation

B.0.1 Modified Mask RCNN based end-to-end solution with improved RPN

Mask RCNN configuration

The table 92 represents the configuration parameters used for training and inference of the modified Mask RCNN model used in this thesis for the contaminant separation problem. Each of the parameter names in the left column are self explanatory. The parameter names preceded by the keyword $DETECTION$ are only used in the detection or inference time whereas most of the parameters are used only during the training time.

The $IMAGE\_MIN\_DIM$ and $IMAGE\_MAX\_DIM$ represents the input image height and width and in this case since we use the image split, the input dimensions are $512, 512$. The $BATCH\_SIZE$ is defined as the product of the $IMAGES\_PER\_GPU \times GPU\_COUNT$.

B.0.2 Modified UNET based end-to-end solution with Watershed Transform

Jaccard Index and Jaccard Loss

The Jaccard index, also known as Intersection over Union and the Jaccard similarity coefficient (originally coined coefficient de communaut

Table 92: Configuration of Modified Mask RCNN.

PARAMETERS	VALUES
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	8
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
DETECTION_MAX_INSTANCES	500
DETECTION_MIN_CONFIDENCE	0.9
DETECTION_NMS_THRESHOLD	0.4
GPU_COUNT	4
IMAGES_PER_GPU	2
IMAGE_MAX_DIM	512
IMAGE_MIN_DIM	512
IMAGE_PADDING	False
IMAGE_SHAPE	[512 512 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.003
MASK_POOL_SIZE	14
MASK_SHAPE	[96, 96]
MAX_GT_INSTANCES	500
MEAN_PIXEL	[129.5 129.46 129.56]
MINI_MASK_SHAPE	(96, 96)
NUM_CLASSES	4
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
ROI_POSITIVE_RATIO	0.67
RPN_ANCHOR_RATIOS	[0.2, 0.5, 1, 2, 5]
RPN_ANCHOR_SCALES	(4, 8, 32, 80, 128)
RPN_ANCHOR_STRIDE	2
RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_TRAIN_ANCHORS_PER_IMAGE	1000
STEPS_PER_EPOCH	170000
TRAIN_ROIS_PER_IMAGE	500
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001

by Paul Jaccard), is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between two finite sample sets, and is defined as the size of the intersection divided by the size of the union of these sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{B.1}$$

Since, an image consists of pixels, the Jaccard index can be adapted in the form of a loss for non discrete objects to be converted into a differentiable form (IMO17) in the following way:

$$J = \frac{1}{n} \sum_{c=1}^{m} w_c \sum_{i=1}^{n} \Big(\frac{y_i^c \hat{y}_i^c}{y_i^c + \hat{y}_i^c - y_i^c \hat{y}_i^c} \Big) \tag{B.2}$$

where y_i^c and $\hat{y}_i^c$ are the binary values (labels) and the corresponding predicted probability for the pixel i of the class c. The value of w_c values are chosen through cross validation based on the class frequencies of each class. The Jaccard loss penalizes for the errors in the segmentation quality.

References

[AHY+18] Md. Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M. Taha, and Vijayan K. Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *CoRR*, abs/1802.06955, 2018. 171

[ARDK15] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Deep compositional question answering with neural module networks. *CoRR*, abs/1511.02799, 2015. xiii, 19, 20

[Ben09] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. 7, 14

[BLP+07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007. 15, 47, 48

[BMG10] Matthias Broecheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 73–82, 2010. 9

[BNG17] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017. 199

[Bra14] Max Bramer. *Logic Programming with Prolog*. Springer Publishing Company, Incorporated, 2nd edition, 2014. 11

[BU16] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. *CoRR*, abs/1611.08303, 2016. 199

[CL97] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc., Orlando, FL, USA, 1997. 8

[CLX+17] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks, 2017. 139

[Coh60] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. 249

[Coh68] Jason A. Cohen. Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological bulletin*, 70 4:213–20, 1968. 249

[Coh16] William W. Cohen. Tensorlog: A differentiable deductive database. *CoRR*, abs/1605.06523, 2016. 18

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. 168

[CR03a] C. Cumby and D. Roth. Learning with feature description logics. In *Proceedings of the 12th international conference on Inductive logic programming*, pages 32–47. Springer, 2003. 14

[CR03b] C. Cumby and D. Roth. On kernel methods for relational learning. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 107–114, 2003. 14

[CSYU14] Liang-Chieh Chen, Alexander G. Schwing, Alan L. Yuille, and Raquel Urtasun. Learning deep structured models. *CoRR*, abs/1407.2538, 2014. 22

[CZH+18] Wanli Chen, Yue Zhang, Junjun He, Yu Qiao, Yifan Chen, Hongjian Shi, and Xiaoying Tang. W-net: Bridged u-net for 2d medical image segmentation. *CoRR*, abs/1807.04459, 2018. 170

[CZY+18] Jie Chang, Xiaoci Zhang, Minquan Ye, Daobin Huang, Peipei Wang, and Chuanwen Yao. Brain tumor segmentation based on 3d unet with multi-class focal loss. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5. IEEE, 2018. 170, 199

[Dar11] Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *IJCAI*, 2011. 21

[DGMR10a] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Multitask kernel-based learning with first-order logic constraints. In *The 20th International Conference on Inductive Logic Programming*, 2010. 13

[DGMR10b] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Multitask Kernel-based Learning with Logic Constraints. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 433–438. IOS Press, 2010. 13

[DGMR10c] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Multitask kernel-based learning with logic constraints. In *The 19th European Conference on Artificial Intelligence (ECAI)*, 2010. 13

[DGMR12] Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Machine learning*, 86(1):57–88, 2012. 13, 20, 22, 30

[DGS15] Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 2015. 22, 30, 42

[DGS16] Michelangelo Diligenti, Marco Gori, and Vincenzo Scoca. Learning efficiently in semantic based regularization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 33–46. Springer, 2016. 23, 73, 79, 117

[DR04] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *ICML-2004 Workshop on Statistical Relational Learning*, pages 49–54, 2004. 9

[DT17] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. xiv, 59, 60, 61, 105, 106, 109

[DXYZ18] Wang-Zhou Dai, Qiu-Ling Xu, Yang Yu, and Zhi-Hua Zhou. Tunneling neural perception and logic reasoning through abductive learning. *CoRR*, abs/1802.01173, 2018. 7, 21

[DYP12] Li Deng, Dong Yu, and John Platt. Scalable stacking and learning for building deep architectures. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 2133–2136. IEEE, 2012. 15

[EBC+10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11, March 2010. 15

[EMGG09] Andreas Ess, Tobias Mueller, Helmut Grabner, and Luc Van Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009. 168

[FCNL13] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013. 48

[FMS02] Glenn M Fung, Olvi L Mangasarian, and Jude W Shavlik. Knowledge-based support vector machine classifiers. In *Advances in neural information processing systems*, pages 521–528, 2002. 13

[GB] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. 48

[GDGM17] Francesco Giannini, Michelangelo Diligenti, Marco Gori, and Marco Maggini. Learning łukasiewicz logic fragments by quadratic programming. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 410–426, Cham, 2017. Springer International Publishing. 42, 43, 81, 97

[Gei12] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3354–3361, Washington, DC, USA, 2012. IEEE Computer Society. 168

[Gir15] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 180

[GLG08] Artur S. d'Avila Garcez, Lus C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer Publishing Company, Incorporated, 1 edition, 2008. 8

[GOO+17] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017. xv, 168

[GT07] Lise Getoor and Ben Taskar. 2007. 9

[Hau99] D. Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999. 12

[HDY+12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke,

Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. 7

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 49, 65, 66, 162, 167, 169, 177, 178, 179

[HHS04] P. Hitzler, S. Holldobler, and A. K. Sedab. Logic programs and connectionist networks. *Journal of Applied Logic*, 2(3):245–272, 2004. 12

[Hin12] Geoffrey E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 47, 48

[HKK16] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *CoRR*, abs/1610.02915, 2016. 48, 53, 54, 94, 105, 109, 126

[HLC+19] Hassan Al Hajj, Mathieu Lamard, Pierre-Henri Conze, Soumali Roychowdhury, Xiaowei Hu, Gabija Marsalkaite, Odysseas Zisimopoulos, Muneer Ahmad Dedmari, Fenqiang Zhao, Jonas Prellberg, Manish Sahu, Adrian Galdran, Teresa Araujo, Duc My Vo, Chandan Panda, Navdeep Dahiya, Satoshi Kondo, Zhengbing Bian, Jonas Bialopetravicius, Chenghui Qiu, Sabrina Dill, Anirban Mukhopadyay, Pedro Costa, Guilherme Aresta, Senthil Ramamurthy, Sang-Woong Lee, Aurelio Campilho, Stefan Zachow, Shunren Xia, Sailesh Conjeti, Jogundas Armaitis, Pheng-Ann Heng, Arash Vahdat, Beatrice Cochener, and Gwenole Quellec. Cataracts: Challenge on automatic tool annotation for cataract surgery. *Medical Image Analysis*, 52(2):24 – 41, 2019. xiv, 143, 144, 146, 148, 158, 159

[HML+16] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. Harnessing deep neural networks with logic rules. *CoRR*, abs/1603.06318, 2016. xiii, 16

[Hor51] Alfred Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16(1):1421, 1951. 29

[HZRS15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 94, 105, 109, 126, 149

[HZRS15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. 48

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. xiv, 48, 51

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. xiv, 52, 94, 105, 109, 126

[IMO17] Vladimir Iglovikov, Sergey Mushinskiy, and Vladimir Osin. Satellite imagery feature detection using deep convolutional neural network: A kaggle competition. *CoRR*, abs/1706.06169, 2017. 257

[IS18] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation. *CoRR*, abs/1801.05746, 2018. 169, 171

[ISBS18] Vladimir I. Iglovikov, Selim S. Seferbekov, Alexander V. Buslaev, and Alexey Shvets. Ternausnetv2: Fully convolutional network for instance segmentation. *CoRR*, abs/1806.00844, 2018. 188

[Joh18] Jeremiah W. Johnson. Adapting mask-rcnn for automatic nucleus segmentation. *CoRR*, abs/1805.00500, 2018. 171

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 150

[KBB+12] Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. 2012. 8, 9

[KD05] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 441–448. ACM, 2005. 9

[KDR02] Kristian Kersting and Luc De Raedt. Towards combining inductive logic programming with bayesian networks. 09 2002. 10

[KGHD19] Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. *CoRR*, abs/1901.02446, 2019. 49

[KHG+18] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *CoRR*, abs/1801.00868, 2018. 181

[KMP00] E.P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publisher, 2000. 33

[Kol99] Daphne Koller. Probabilistic relational models. In *Proceedings of the 9th International Workshop on Inductive Logic Programming*, ILP '99, 1999. 10

[Kri09] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009. 103, 124

[KSH12a] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 7, 48

[KSH12b] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 23, 48

[LCQ+17] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-denseunet: Hybrid densely connected unet for liver and liver tumor segmentation from CT volumes. *CoRR*, abs/1709.07330, 2017. 170

[LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. xiii, 48, 49, 50, 94

[LDD+18] Menglu Liu, Junyu Dong, Xinghui Dong, Hui Yu, and Lin Qi. Segmentation of lung nodule in ct images based on mask r-cnn. In *2018 9th International Conference on Awareness Science and Technology (iCAST)*, pages 1–6. IEEE, 2018. 171

[LDG+16] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. xiv, 63, 177, 180

[LeC15] Yann LeCun. Deep learning. *Nature*, 521, 2015. xiii, 7, 14, 15, 47

[LF09] Marco Lippi and Paolo Frasconi. Prediction of protein β-residue contacts by markov logic networks with grounding–specific weights. *Bioinformatics*, 25(18):2326–2333, 2009. 23

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 199

[LGRN09] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009. 48

[LHB04] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–104. IEEE, 2004. 48

[LL18] Jun Liu and PengFei Li. A mask r-cnn model with improved region proposal network for medical ultrasound image. In *International Conference on Intelligent Computing*, pages 26–33. Springer, 2018. 171, 177, 179

[LLY+18] Ruirui Li, Wenjie Liu, Lei Yang, Shihao Sun, Wei Hu, Fan Zhang, and Wei Li. Deepunet: A deep fully convolutional network for pixel-level sea-land segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, (99):1–9, 2018. 171

[LPDRF06] Niels Landwehr, Andrea Passerini, Luc De Raedt, and Paolo Frasconi. kfoil: Learning simple relational kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 6, pages 389–394, 2006. 13

[LSD14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 169, 179

[LSRvdH15] Guosheng Lin, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. Efficient piecewise training of deep structured models for semantic segmentation. *CoRR*, abs/1504.01013, 2015. 22

[MBRR18] Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, and Sebastian Riedel. Towards neural theorem proving at scale. *CoRR*, abs/1807.08204, 2018. 19

[MDK+18] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *CoRR*, abs/1805.10872, 2018. 18

[MDP+11] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE, 2011. 48

[Met10] DE) Metzger, Johann (Munich. Method for particle analysis and particle analysis system, February 2010. xv, 166

[MH16] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354, 2016. 21

[MLAS05] Stephen Muggleton, Huma Lodhi, Ata Amini, and Michael JE Sternberg. Support vector inductive logic programming. In *Discovery science*, pages 163–175. Springer, 2005. 13

[MR18] Pasquale Minervini and Sebastian Riedel. Adversarially regularising neural NLI models to integrate logical background knowledge. *CoRR*, abs/1808.08609, 2018. 18

[Mug] Stephen Muggleton. Inductive logic programming: Derivations, successes and shortcomings. *SIGART Bull.*, pages 5–11. 8

[Mug96] Stephen Muggleton. Stochastic logic programs. In *New Generation Computing*. Academic Press, 1996. 10, 18

[Nak16] Makoto Nakatsuji. Semantic sensitive simultaneous tensor factorization. pages 411–427, 2016. 17

[Nev15] Gomes Vicente Santos Neves Machado Novais Paulo Neves, Guimaraes. Logic programming and artificial neural networks in breast cancer detection. In *Advances in Computational Intelligence*, pages 211–224, 2015. 17

[NJ07] Jennifer Neville and David Jensen. Relational dependency networks. *J. Mach. Learn. Res.*, 8, May 2007. 11

[Nov87] Vilém Novák. First-order fuzzy logic. *Studia Logica*, 46(1):87–109, 1987. 30, 34

[NS56] A. Newell and H. Simon. The logic theory machine–a complex information processing system. *IRE Transactions on Information Theory*, 2(3):61–79, 1956. 8

[NYC14] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014. 24

[OWL15] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *CoRR*, abs/1502.06807, 2015. 168

[PPU03] Alexandrin Popescul, Rin Popescul, and Lyle H. Ungar. Structural logistic regression for link analysis, 2003. 11

[RAHL18] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search, 2018. 139

[RD06a] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1–2):107–136, 2006. 9

[RD06b] Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006. xiii, 9, 10, 23

[RDG18] Soumali Roychowdhury, Michelangelo Diligenti, and Marco Gori. Image classification using deep learning and prior knowledge, 2018. 97

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. xiv, 67, 68, 162, 167, 169, 170, 187, 188

[RFKE08] L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton (Eds). *Probabilistic Inductive Logic Programming*, volume 4911. Springer, Lecture Notes in Artificial Intelligence, 2008. 18

[RHGS15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 64, 65

[RKT07] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007. 8, 18, 19, 23

[RR16] Tim Rocktschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. pages 45–50, 01 2016. 19

[SDdG17] Luciano Serafini, Ivan Donadello, and Artur S. d'Avila Garcez. Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation. In *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*, pages 125–130, 2017. 17, 23

[SGDG18] Sohil Shah, Pallabi Ghosh, Larry S. Davis, and Tom Goldstein. Stacked u-nets: A no-frills approach to natural image segmentation. *CoRR*, abs/1804.10343, 2018. 169

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. xiv, 24, 48, 56, 57

[SMKR13] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*, pages 8614–8618. IEEE, 2013. 48

[SNB+08] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008. 40

[SS18] Janpreet Singh and Shashank Shekhar. Road damage detection and classification in smartphone captured images using mask r-cnn. *arXiv preprint arXiv:1811.04535*, 2018. 171

[STE13] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *NIPS*, 2013. 7

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 56, 149

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 48, 55

[TAWK] Ben Taskar, Pieter Abbeel, Ming-Fai Wong, and Daphne Koller. Relational markov networks. 11

[TJLB14] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014. 48

[TL19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019. 138

[TS94] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1):119 – 165, 1994. xiii, 7, 11, 12, 13, 23

[TYW14] Yaniv Taigman, Ming Yang, and Lior Wolf. L.: Deepface: Closing the gap to human-level performance in face verification. In *In: IEEE CVPR*, 2014. 7

[WD08] J. Wang and P. Domingos. Hybrid markov logic networks. In *Proceedings of the 23-rd AAAI Conference on Artificial Intelligence*, pages 1106–1111, 2008. 9, 23

[WH86] Patrick Henry Winston and Berthold K Horn. *Lisp*. Addison Wesley Pub., Reading, MA, 1986. 73

[Win] Particle contamination. `https://www.leica-microsystems.com/science-lab/cleanliness-analysis-in-\relation-to-particulate-contamination/`. xv, 162, 163

[WWH+14] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 157–166, New York, NY, USA, 2014. ACM. 168

[WWZY17] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2915–2921, 2017. 17

[XGD+16] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. xiv, 49, 55, 56, 177

[XLZS18] Xinpeng Xie, Yuexiang Li, Menglu Zhang, and Linlin Shen. Robust segmentation of nucleus in histopathology images via mask

r-cnn. In *International MICCAI Brainlesion Workshop*, pages 428–436. Springer, 2018. 171

[XSNT18] Xiaozheng Xie, Faqiang Shi, Jianwei Niu, and Xiaolan Tang. Breast ultrasound image classification and segmentation using convolutional neural networks. In *Pacific Rim Conference on Multimedia*, pages 200–211. Springer, 2018. 171

[XZF+18] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. pages 5502–5511, 2018. 20

[YTRW18] Fan Yu, Xin Tan, Tongwei Ren, and Gangshan Wu. Human-centric visual relation segmentation using mask r-cnn and vtranse. In *European Conference on Computer Vision*, pages 582–589. Springer, 2018. 169

[Zad65] Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965. 30

[ZF13] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 48

[Zhu18] Juntang Zhuang. Laddernet: Multi-path networks based on u-net for medical image segmentation. *CoRR*, abs/1810.07810, 2018. 170

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 48, 54, 55, 105, 109, 188

[ZLW17] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *CoRR*, abs/1711.10684, 2017. 171

[ZMRSTL18] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. *UNet++: A Nested U-Net Architecture for Medical Image Segmentation: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings*, pages 3–11. 09 2018. 170

[ZVSL17] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. 57, 59, 139, 149

[ZYZ+18] Zhuo Zhao, Lin Yang, Hao Zheng, Ian H Guldner, Siyuan Zhang, and Danny Z Chen. Deep learning based instance segmentation in 3d biomedical images using weak annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 352–360. Springer, 2018. 171

[ZZK+17] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017. xiv, 59, 60, 105, 106, 109