Novel Image Representations and Learning Tasks

by

Ragav Venkatesan

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2017 by the
Graduate Supervisory Committee:

Baoxin Li, Chair
Hasan Davulcu
Pavan Turaga
Yezhou Yang

ARIZONA STATE UNIVERSITY

December 2017

ABSTRACT

Computer Vision as a field has gone through significant changes in the last decade. The field has seen tremendous success in designing learning systems with hand-crafted features and in using representation learning to extract better features. In this dissertation some novel approaches to representation learning and task learning are studied. Multiple-instance learning which is generalization of supervised learning, is one example of task learning that is discussed. In particular, a novel non-parametric k-NN-based multiple-instance learning is proposed, which is shown to outperform other existing approaches. This solution is applied to a diabetic retinopathy pathology detection problem effectively.

In cases of representation learning, generality of neural features are investigated first. This investigation leads to some critical understanding and results in feature generality among datasets. The possibility of learning from a mentor network instead of from labels is then investigated. Distillation of dark knowledge is used to efficiently mentor a small network from a pre-trained large mentor network. These studies help in understanding representation learning with smaller and compressed networks.

To my advisor Dr. Baoxin Li,

for six years of unremitting inspiration, inexhaustible patience and sage counsel.

ACKNOWLEDGMENTS

The completion of my doctorate and this dissertation would be incomplete without an appreciation for the many, who made this possible. I would like to begin by thanking Prof. Baoxin Li. His presence and advice has helped me survive many-a-failure during the course of this doctoral study. My greatest achievement in life would be living up to his standard of integrity, pragmatism, perseverance and relentless curiosity. I would also like to thank him for making my dream of teaching come true in letting me teach all those important lectures both as a TA for the machine learning course and as an instructor for the deep learning course. Preparing those lectures helped me learn deeper insights and the deep learning course would not have been possible without his help.

I would also like to acknowledge my dissertation committee - Prof. Pavan Turaga, Prof. Hassan Davulcu and Prof. Yezhou Yang for all the insightful discussions and guidance. I would also like to thank Prof. Jingrui He and Prof. Pitu Mirchandani of ASU, Dr. Farshad Akhbari and Dr. Zafer Kadi of Intel for their discussions and insights during key moments of my doctoral study.

I would like to thank current and past members of the Visual Representation and Processing Group (VRPG), Yochen Lab, Data Mining and Machine Learning Lab (DMML) and Cognitive and Ubiquitous Learning Center (CUbiC) at Arizona State University for all the insightful discussions, arguments, debates and white board sessions that led to the key insights in this dissertation. In particular, I would like to acknowledge Dr. Parag Chandakkar, Dr. Hemanth Venkateshwara, Dr. Qiang Zhang, Dr. Peng Zhang, Lydia Manikonda, Dr. Suhas Renganath and Dr. Archana Paladugu. I would particularly like to acknowledge my colleagues at VRPG, Yuzhen Ding, Vijetha Gauttpalli, Yilin Wang and Yikang Li for being tolerant of my consistent overuse of the lab's shared resources.

I would like to extend my gratitude to the IT staff at the School for Computing Informatics and Decision Systems Engineering (CIDSE), Brint MacMillan and James White for their relentless commitment to quality of service. I also would like to acknowledge CIDSE support staff including Monica Dugan, Pamela Dunn and the advising staff including Arzuhan Kavak and Christina Sebring for helping me wade through the sea of bureaucracy and clear every stage of my study at ASU on-time.

This work stemmed from efforts in several projects sponsored by the National Science Foundation, the Office of Naval Research, the Army Research Office, and Nokia, whose supports are greatly appreciated, although any views/conclusions in this dissertation are solely of the author's and do not necessarily reflect those of the sponsors. I also gratefully acknowledge the support of NVIDIA Corporation with the donation of GPU compute, which has been used in my research. I would also like to acknowledge CRC press for their help in getting the Convolutional Neural Networks in Visual Computing book published. Writing the book helped provide important historical context for this dissertation.

Finally, I would like to thank the unwavering support of my parents and my sister. My career would not have been possible if not for them. I would also like to acknowledge Harshil Shah, my roommate and forever friend for all tolerating the marker stench, loose papers and unwashed dishes for 5 years. It was a stress-free time for me because of his constant advice and motivation.

Ragav Venkatesan

October 2017.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 INTRODUCTION 1

1.0.1 Image Representations 3

1.0.2 Multiple Instance Learning 4

1.0.3 Representation Learning 5

1.1 Background on Hand-Crafted Features 8

1.2 Deep Image Features 13

1.2.1 Convolutional Neural Networks 13

1.2.2 CNN Architecture Design 18

2 NON-PARAMETRIC MULTIPLE INSTANCE LEARNING 21

2.1 Introduction 21

2.2 Related Works 24

2.3 The Non-parametric MIL Approach 28

2.3.1 Learning Under This Formulation 32

2.4 Experiments and Results 34

2.4.1 Musk Dataset 35

2.4.2 Andrew's Datasets 35

2.4.3 Corel Dataset 37

2.4.4 A DR Dataset 37

2.4.5 Sensitivity to Labeling Error 39

2.5 A Simple Case Study Describing the Effectiveness of the Proposed Method 40

CHAPTER Page

2.6 Analogical Difference Between DD and the Proposed Formulation. . 45
2.7 Computational Complexity 46
2.8 Sensitivity to k 47
2.9 Conclusion 48
3 NEURAL DATASET GENERALITY 49
3.1 Introduction 49
3.2 Related Work 53
3.3 Design of Experiments 55
3.3.1 Dataset Generality 57
3.3.2 Class Generality 57
3.3.3 Datasets Used 58
3.3.4 Network Architecture and Learning 59
3.4 Results and observations 61
3.4.1 Character Datasets 61
3.4.2 CIFAR 10 vs. Caltech 101 64
3.4.3 Caltech 101 vs. Colonoscopy 64
3.4.4 Summary of Results 65
3.5 Conclusions 65
4 MENTEE NETS 71
4.1 Introduction 71
4.2 Related Works 74
4.3 Generalized Mentored Learning 75
4.4 Design of Experiments 80
4.4.1 Effectiveness 80

CHAPTER Page

4.4.2 Generality of the Learnt Representations 81
4.4.3 Learning the VGG-19 Representation 82
4.4.4 Implementation Details 83
4.5 Results 84
4.6 Conclusions 88
5 INCREMENTAL LEARNING 90
5.1 Introduction 90
5.2 Proposed Method 94
5.3 Related Work 100
5.4 Experiments and Results 103
5.4.1 Single Dataset Experiments 105
5.4.2 Cross-Domain Increments 108
5.5 Extension to Bounded-Continual Learning 110
5.5.1 Experiments and Results 113
5.6 Conclusions 114
6 CONCLUSIONS 115
REFERENCES 116
APPENDIX
A PERMISSION STATEMENTS FROM CO-AUTHORS 126

LIST OF TABLES

Table Page

2.1 Performance of Various MIL Algorithms On the Musk Dataset. 36

2.2 Performance of Various MIL Algorithms on Andrew's Dataset. 37

2.3 Performance of Various MIL Algorithms on Corel Dataset. 38

2.4 Performance of various MIL Algorithms on DR Dataset. 40

3.1 Sub-sample Experiment Results 63

LIST OF FIGURES

Figure		Page
1.1	Sampling and Quantization	8
1.2	Histogram of Oriented Gradients	11
1.3	A Typical Dot-Product Layer	14
1.4	Locally Connected Neurons	15
1.5	A Typical Convolution Layer	17
1.6	Learnt Filters	19
2.1	DR Image Classification as a MIL Problem	22
2.2	An Illustrative Feature Space for Multiple-Instance Setting	23
2.3	Parsing the MIL Feature Space with a Parzen Window	29
2.4	2D MIL Feature Space and its Parse Using The k-NN	31
2.5	Performance of NP-MIL	41
2.6	Noise Performance of NP-MIL	42
2.7	EMDD's Failure Case	43
2.8	Accuracy vs k	47
3.1	Generality Thought Experiment	50
3.2	Protocol for Obstination	55
3.3	Dataset Samples	67
3.4	Dataset Generalities	68
3.5	Learnt Filters for Datasets	69
3.6	Validation Errors vs Epoch Number for Base-MNIST-rotated-bg Re-trained on MNIST	69
3.7	Sub-class Generalities for MNIST $[4, 5, 8]$	70
4.1	Mentor Mentoring Mentee on the Second Hidden Layer.	73
4.2	Annealing Rates α, β and η	79

Figure Page

4.3 VGG-19 and Caltech 101, Annealing 82
4.4 Architecture and Results for the Experiments with CIFAR and MNIST Datasets. 85
4.5 Architecture and results for the Experiments with Caltech Datasets. . . . 86
4.6 VGG-19 First Layer Filters and Filters Probed using Caltech101 87
5.1 Catastrophic Forgetting 91
5.2 Incremental Learning Protocol 95
5.3 Results for the MNIST Dataset. 104
5.4 Results for the CIFAR10 Dataset 109
5.5 Results for the SVHN Dataset 110
5.6 results for MNIST-rotated × MNIST 111
5.7 Results for MNIST × SVHN 112
5.8 Results for the Bounded-continual Learning Experiments 113

Chapter 1

INTRODUCTION

Men and women walk out of the cave, and they look over the hill and they see fire; and they cross the ocean; and they take to the sky; and they plant a flag on the moon. The history of man is hung on a timeline of exploration and discovery; for the goal of human scientific exploration is to advance human capabilities, if only to explore further and beyond.

Humans figured out the creation of fire to cook food. This outgrew their dependence on the primitive and limited processing capability of their digestive system. No other species is known to use fire in a deliberate manner in assisting digestion, let alone in satisfying their taste buds. This increased caloric consumption with limited labor. Organized farming and food production surplus led to the development of civilization, art and science. The invention the wheel and vehicles implied that our speed of travel is no more limited by the speed of walking. The legend of human scientific and technological growth is a narrative of the humanity endlessly out-growing their capabilities in the march towards future.

These developments are courtesy the wiring in the human brain. We posses a complex neural system capable of thought, emotion, reasoning, imagination and philosophy. The human visual system is an integral part of this system. Humans do not have the most powerful visual system among all the species. For instance, when acuity or night-vision capabilities are concerned humans have significant deficiencies. The human visual system is also slow Thorpe *et al.* (1996); Watamaniuk and Duchon (1992). Humans are also receptive to a very narrow range of the electromagnetic spectrum. To compensate for the lack of a powerful visual system using our powerful

neural system, computer vision scientists have been studying images and videos in various forms to identify and make sense of patterns.

Humans have been developing tools to try and exceed the capability of the human eye. Devices such as telescopes, binoculars, microscopes and magnifiers were invented in accomplishing this goal. Other devices such as radio, infrared and X-ray devices make us see parts of the electromagnetic spectrum, that we can not naturally perceive. Modern achievements such as the LIGO interferometers extend human vision to include gravity waves. This makes way for yet another way to look at the world. Computer vision in the progress of technological additions to human capabilities, is the progress of gaining knowledge from all these modalities. This, is informally, the field of computer vision and is an amalgamation of ideas from signal processing, statistics and machine learning.

Computer vision, the process of extracting knowledge from images and image datasets. It is currently a prominent area of research in computer science. A typical computer vision system involves two steps: gathering representations of images and making inferences from those representations. This dissertation concerns the research and development of some novel techniques in both these steps. This dissertation is broadly divided into two parts: The first is the research and development of non-parametric multiple instance learning algorithms using hand-crafted representations. In this step, a new representation for a particular type of images are introduced. A new form of non-parametric learning designed particularly for this problem statement is studied. The second concerns the study of representations learnt using deep convolutional neural networks (CNN) and some novel learning tasks performed using these

representations. Firstly, let us survey the basics of image representations.

1.0.1 Image Representations

Since this dissertation deals with images, it is important to understand how images are captured, stored and represented. Images are mathematical representations of the natural world. A view of the natural world is captured and stored as a matrix of numbers in a computer. The process of capturing an image involves sampling and quantization. Sampling is the process of choosing samples of light intensities arriving at a camera. Quantization is the process of assigning a value for the light intensity for the sample. The stored numbers are called pixels. The pixel representation is the most fundamental image representation.

There are other representations that are secondary and can be derived from the pixel representations. These can be derived using some deterministic process. The mean of the image for instance is a representation of the image that can be useful in determining if an image is dark or bright. The histogram of pixels in an image is another useful representation that can provide more information on the tonality of the image globally. These feature representations are often referred to as global features. Local representations are features that work on blocks or regions of images. Histogram of oriented gradients(HOG) for instances is a local histogram of gradients that represents shape features locally Dalal and Triggs (2005a).

These feature representations are mostly hand-crafted with intuition and domain-knowledge. Recently, neural networks are used that can learn feature representations which are task-specific. In the case of images, the preferred neural network architecture is that of the convolutional neural network Venkatesan and Li (2017). We will study these features in more detail, in chapter 1.1.

1.0.2 Multiple Instance Learning

The most common formulation of machine learning and the one most studied is supervised learning. Supervised learning is a setting where, we have labeled data $\mathcal{D} = \begin{bmatrix} x^{(1)} & y^{(1)} \\ x^{(2)} & y^{(2)} \\ \vdots & \vdots \\ x^{(n)} & y^{(n)} \end{bmatrix}$ where $x^{(i)}$ is the i^{th} data-point in the dataset and $y^{(i)} \in \{0, 1\}$ is its label. Supervised learning is the process of learning a model that can emulate the mapping $m : x \rightarrow y$. Every sample of instance in this setup has its own unique label.

Other forms of learning also exists. Multiple instance learning is a generalization of supervised learning. It is a paradigm where labels are not available for every sample in the training dataset. Labels are available only for sets of samples called bags. The labels for bags and instances map on to the same space. Suppose we have labeled data $\mathcal{D} = \begin{bmatrix} X^{(1)} & y^{(1)} \\ X^{(2)} & y^{(2)} \\ \vdots & \vdots \\ X^{(n)} & y^{(n)} \end{bmatrix}$ where $X^{(i)}$ is the i^{th} bag in the dataset and $y^{(i)} \in \{0, 1\}$ is its label. Each bag $X^{(i)}$ contains k_i (often is a constant k by design, particularly in image classification contexts) instances such that $X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots x_{k_i}^{(i)}\}$. This definition of a dataset is perfect for MIL setups.

Consider a case with only two types of labels, the positive and the negative. Every positively labeled bag must have at least one inherently positive instance and all negative bags must not have even a single positive instance. While this paradigm is well-studied, non-parametric approaches are widely unexplored. Several non-parametric ideas were explored in this effort and chapter 2 will study some of them in detail Venkatesan *et al.* (2015, 2012a); Chandakkar *et al.* (2013a, 2017, 2012).

1.0.3 Representation Learning

The second thrust of research in this dissertation is in studying data-driven representations extracted from images and image datasets using Convolutional Neural Networks (CNNs). Obviously, in a processing pipeline where the representations are hand-crafted by domain expertise, opportunities for performance improvements lie in finding an informative feature representation. It is often difficult to rely on any set of domain expertise for designing perfect features for a given problem. Computer vision researchers have been employing techniques to learn optimal features directly from data instead of from domain expertise.

For example, Nagest et al., Nagesh and Li (2009) proposed a novel image representation for face recognition. They used compressive sensing theories in compactly representing a set of images from the same subject Candes *et al.* (2006). Kulkarni et al., defined image features as affine sparse codes which are learned from a large collection of image patches Kulkarni and Li (2011). These features were then used for image categorization. Joint feature and classier learning techniques are also popular, where a dictionary for sparse coding as well as a classifier using the sparse representation under the dictionary can be jointly learned from labeled data Zhang and Li (2010). Many manifold learning techniques creates efficient representations of the data in some sub-spaces where features for classification may be done better Belkin and Niyogi (2002); Tenenbaum *et al.* (2000); Roweis and Saul (2000). The most popular of such approaches to learn representations from data (and the most general) is by using neural networks.

Recently CNNs have resurfaced as a potent tool for addressing computer vision problems, many under the re-branding of *deep-learning*. Artificial neural networks are a class of learning approaches where the weights for an input-output algebraic

functional mapping is structured according to a pre-defined architecture. A brief background on learning these techniques have been provided in chapter 1.2. The rest of this dissertation discusses three major efforts in studying neural networks for computer vision tasks.

Neural Dataset Generality

This effort concerns the question of generality of representations learnt from CNNs. It is widely believed that networks that are trained with large datasets often learns sophisticated representations. These representations can then be *fine-tuned* to work with any other smaller datasets. Some datasets have data, larger in number of classes and/or number of images than others. Some datasets can provide more diversity of images to learn neural features from. Since the images across these datasets also *appear* similar, an argument could be made that a network trained on one dataset could potentially be used as a feature extractor for another without explicitly being learnt or simply fine-tuned from the latter. Suppose we have two datasets where one has considerably more number of images and object classes than the other. Suppose we are building a CNN to learn to categorize the latter, and we use the training set of the latter dataset alone. Since we have less data, we might overfit. Also, with fewer images, it is difficult to train deeper and larger CNNs. If we were to train a network on the first and destroy the softmax layer, we could now rebuild another softmax layer for the categorization of latter dataset. All the layers but the last softmax layer can be carried over in the hope that two datasets can use the same features. This study challenges that view and studies the generality of some datasets over others in regard to representations learnt from them Venkatesan *et al.* (2016a). Chapter 3 will cover this in more detail.

Mentee Networks

It is often very difficult to train deep or even mid-sized networks from random initializations with small datasets. Weights explode and gradients vanish. Overfitting becomes an issue in networks that are over-parameterized. Strong regularization techniques are often used and needed to make the training of these networks well-behaved. Regularization techniques such as penalizing the network for learning large L_1 or L_2 of parameters are popular in avoiding these problems. They simply keep the weights small and impose sparsity. Mentoring is a regularization process where we force a smaller or a mid-sized network to be constrained by the activities of a larger network. Consider that we have a large network trained previously by some dataset. To train a smaller network, we can use the outputs of the larger network as a guide. In essence, we can penalize a small mentee network for not learning the parameters space as approved by the large mentor network. These techniques are often applied using the distillation process Hinton *et al.* (2015). This effort proposes and studies some mentoring protocols Venkatesan and Li (2016) using these techniques. Chapter 4 will cover this in more detail.

Incremental Learning

Multi-class supervised learning systems require the knowledge of the entire range of labels they predict. Often when learnt incrementally, they suffer from catastrophic forgetting. In this effort, a strategy is developed involving generative models and the distillation of dark knowledge as a means of hallucinating data along with appropriate targets from past distributions. This strategy is found to alleviate problems concerning catastrophic forgetting and help us achieve incremental learning Venkatesan *et al.* (2017). Chapter 5 will cover this in more detail.

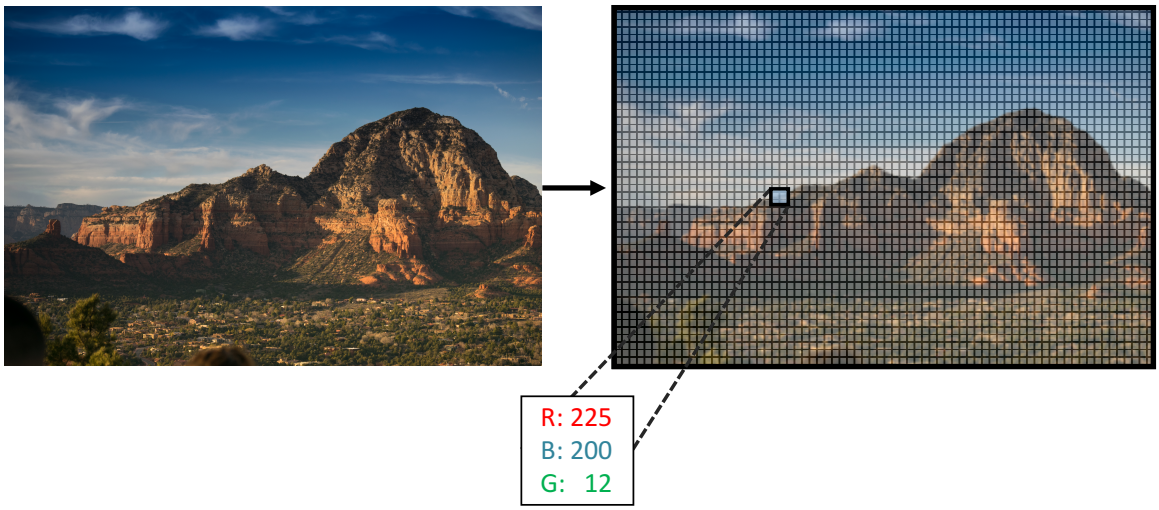


Figure 1.1: Sampling and Quantization: Collecting primary feature representations from real-world.

1.1 Background on Hand-Crafted Features.

A typical computer vision pipeline starts with the lens of an imaging system that *incidents* light rays from the scene and converts into into an image. This image should be in a format that a computer could process. Earlier, the image was obtained by digitizing an analog film or a printed picture. Modern images are typically obtained directly by digital cameras. A digital image is usually represented by a set of ordered numbers called pixels. For a complete study of image acquisition refer to Jain (1989); Gonzalez and Woods (2002).

Consider figure 1.1. This figure shows two key process in image acquisition: sampling (i.e., discretization via the image grid) and quantization (i.e., representing each pixels color values with only a finite set of integers). An image is represented as a matrix (or three matrices for RGB) of quantized numbers. Consider Figure 1. The picture shown was captured by a camera using these steps. Let us suppose that each sensor in the camera's sensor array grabbed a sample of light that was incident on

that area of the sensor after it passed through a lens. The sensor will produce an electric current of a certain magnitude, related to the intensity of light incident on it. This electric current is then sampled as a sample a value between 0 and $2^b - 1$) for a b-bit image. This process is called sampling and quantization, as illustrated in figure 1.1. If the sensor array had $n \times m$ sensors, the image it produces will be $n \times m$ in size. Each sensor grabbed a sample of light that was incident on that area of the sensor after it passed through a lens. With sampling and quantization we are able to represent an image in the computer as stored digital data. This representation of the image is called the pixel representation of the image. Each representation is typically a matrix or tensor of 1 (grayscale) or 3 (colored) channels. The ordering of the pixel intensities is typically the same as that of the ordering of the sensor locations from which they were collected, although vectorized representations can also be formed.

While this representation is holistic and is good to *look at*, it is not a great feature representation for most ML tasks. Several secondary features could be derived from these features that are more suitable for ML tasks. A popular secondary feature representation is the basis representation. A basis representation can be described as a linear projection as follows:

$$I_T(u, v) = \sum_{n_1=0}^{n-1} \sum_{n_2=0}^{m-1} I(n_1, n_2) b(n_1, n_2, u, v), \tag{1.1}$$

where, $I_T(u, v) \in \mathbb{R}^{m \times n}$ is a representation projected on a basis b if $I(n_1, n_2)$ is the representation of the image. Often times, these representations are reversible and one can obtain the image back using an inverse basis b'. A common set of basis are the Fourier basis represented by,

$$b = \frac{1}{nm} e^{-j2\pi\left(\frac{un_1}{n} + \frac{vn_2}{m}\right)} \tag{1.2}$$

and

$$b' = e^{j2\pi\left(\frac{un_1}{n} + \frac{vn_2}{m}\right)}. \tag{1.3}$$

Perhaps the most simplest of secondary feature is the mean intensity as described by,

$$I_m = \frac{1}{nm} \sum_{n_1=0}^{n-1} \sum_{n_2=0}^{m-1} I(n_1, n_2). \tag{1.4}$$

This representation gives us the mean of the entire image and can be used to estimate the overall brightness of the image. As simple as it is, it holds very less entropy and is therefore not much useful. An image histogram is another common global image feature. An 8-bit image has pixel values in the range of $0 - 255$. Generally, if we had a b-bit representation of image, we can have 2^b unique quantization levels. A normalized histogram representation of the image in this case is,

$$I_h(i) = \frac{1}{nm} \sum_{n_1=0}^{n-1} \sum_{n_2=0}^{m-1} \mathbb{I}(I(n_1, n_2) = i), i \in [0, 1, \ldots, 2^b - 1], \tag{1.5}$$

where, $\mathbb{I}$, is an indicator function. While histograms are more informative than the mean, they are still only a coarse global representation and loses a lot of useful information. For instance, Images with different visual content might still have similar histograms. Different retinal-scanned, fundus camera images can all have very similar histograms even though they may can contain different pathologies Venkatesan *et al.* (2012a).

We can seek representations that are local. The most common and widely-used image features represent local image properties are edges. Edges are typically first order gradients of images. Typically, most edges are extracted by convolving the image with an edge filter. Consider a filter $F(l, w)$ that is crafted such that we can use to detect the pattern in the filter $F(l, w)$ as defined in range $[-a, a] \times [-b, b]$ by the following convolution:

$$Z(n_1, n_2) = \sum_{l=-a}^{a} \sum_{w=-b}^{b} F(l, w) I(n_1 + l, n_2 + w) \quad \forall n_1, n_2. \tag{1.6}$$

We can use this *filtering* approach to detect not just edges but any patterns. We can

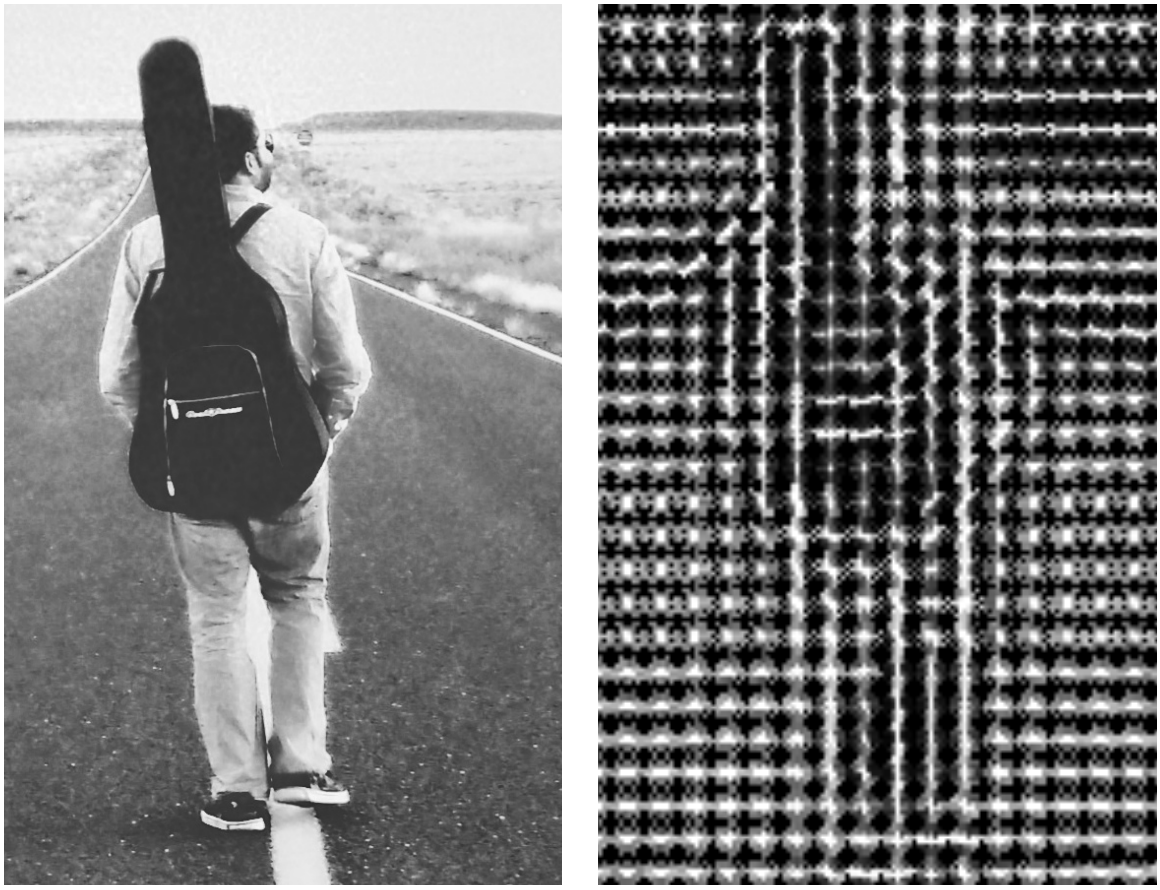


Figure 1.2: Histogram of Oriented Gradients

do this by carefully creating pattern templates for filters. One popular way to create a sophisticated *shape feature* is to use the Histogram of Oriented Gradients(HOG) extractor Dalal and Triggs (2005b).

Given an image we can extract the gradients. We may also quantize the gradient directions into j bins. Using a non-overlapping sliding window of $k \times k$, we could slide through the entire image and extract patch-wise histograms of these gradients. The value of the bin in these histograms is the magnitude of the gradient at the corresponding direction. We will then have a j-bin histogram for every such patch. Each bin (representing some range of angles) will represent the magnitude of that directions gradient in that patch.

Figure 1.1 shows this representation. The length of the arrow describes the magintude of the gradient of that bin in the orientation that arrow is pointing to. This is a better representation of the direction of edges in a manner that is better than gradients. HOGs are very popular in detection of shape based objects in images particularly, human pedestrians.

These hand-crafted latent features were, for the most part, general enough that they were used for a variety of tasks. While this school of thought continues to be quite popular and some of these features have standardized implementations that are available for most researchers to plug and play for their tasks, they were not task-specific.

Hand-crafted features were designed to be useful feature representations of images that are capable of providing cues about certain aspects of images that are agreed upon to be important based on our prior knowledge. HOG and Shape Context for instance, were good shape-related information and were therefore used in tasks involving shape and structure, such as pedestrian detection. Features like color correlogram Venkatesan *et al.* (2012b); Chandakkar *et al.* (2013b, 2017) provided cues on color transitions and were therefore used in medical images and other problems where shape was not necessarily an informative feature.

In the next section we will study a popular technique used in *learning to create good features* and task-specific feature extractors. These feature extractors are learned directly from raw images with minimal and often no pre-processing. This is quite desirable as features developed using a massive dataset that encompasses most entropy found in the real-life can be reused for other tasks and other (smaller) datasets.

1.2 Deep Image Features

Multi-layer neural networks were long since viewed as a technique for learning and extracting hierarchical task-specific features. Ever since the early works of Rumelhart et al., Rumelhart *et al.* (1985) it was recognized that representations learned by back-propagation had the potential to learn fine-tuned features that were task-specific. Until recently, these methods were severely hampered by a dearth of large-scale data and large-scale parallel compute hardware to be leveraged sufficiently. This, in part, focused the creativity of computer vision scientists to develop the aforementioned general-purpose and hand-crafted feature representations. Recently, we developed datasets that are large enough and GPUs that are capable of large-scale parallel computations. This resulted in an explosion of neural image features and their usage. In this section we will survey some of these techniques.

1.2.1 Convolutional Neural Networks

There are many types of neural networks. A neural network is a graphical connection of computational neurons and can be mathematically represented as a composition of multiple functions Goodfellow *et al.* (2016). In the case of images, we are primarily concerned with the use of a convolutional neural network (CNN).

In a computational neuron, each neuron accepts a number of inputs and produces one output. These outputs can further be supplied to many other neurons in a deeper hierarchy. A typical computational neuron weighs all the inputs, sums all the weighted inputs and generates an output depending on the strength of the summed and weighted inputs. Neurons are typically organized in groups or hierarchies called as layers. Each layer typically receives input from one previous layer. Layers come in three varieties, each with a unique type of a neuron. These are, the dot-product,

the fully-connected layer, the convolutional layer or the pooling layer. Although one layer may contain different types of neurons, typically most layers have the same type of neurons.

The Dot-Product Layer

Consider a $1D$ vector of inputs $\mathbf{x} \in [x_0, x_1, \ldots x_d]$ of d dimensions. x could be a vectorized version of an image or could be the output of a preceding layer. Consider a dot-product layer containing n neurons. The j^{th} neuron of this layer will produce the output,

$$z_j = \alpha\left(\sum_{i=0}^{d-1} x_i \times w_i^j\right), \tag{1.7}$$

where, α is typically an element-wise monotonically-increasing function that scales the output of the dot-product. α is referred to as the activation function. The output of this neuron that has been processed by an activation is also referred to as a *neuron activity*. Inputs can be processed in batches or mini-batches. In these cases $\mathbf{x}$ is a mini-batch (matrix instead of a vector) in $\mathbb{R}^{b\times d}$, where b is the batch size. The vectorized output of the layer is the dot-product operation between the matrix of

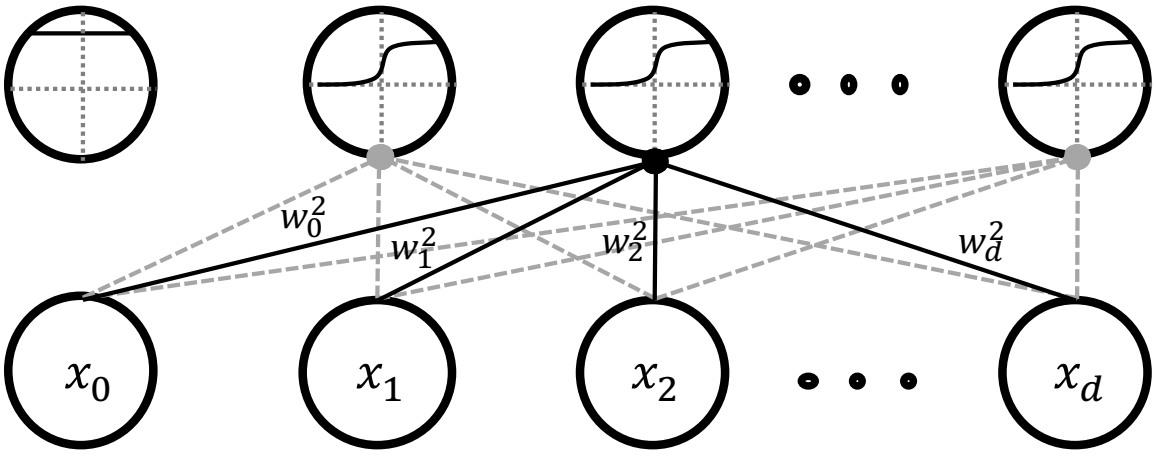


Figure 1.3: A Typical Dot-Product Layer

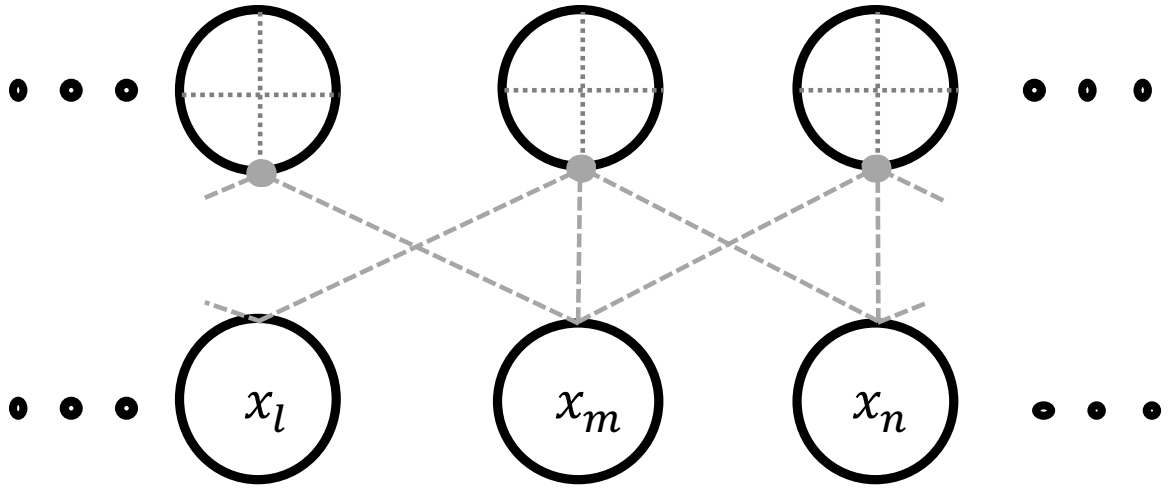


Figure 1.4: Locally-connected neurons with a receptive field of $r = 3$.

layer's weights and the input signal batch,

$$\mathbf{z} = \alpha(\mathbf{x} \cdot \mathbf{w}), \tag{1.8}$$

where, $\mathbf{z} \in \mathbb{R}^{b \times n}$, $\mathbf{w} \in \mathbb{R}^{d \times n}$ and the $(i, j)^{\text{th}}$ element of $\mathbf{z}$ is the output of the j^{th} neuron for the i^{th} sample of input. Figure 1.3 shows the connectivity of the dot-product layer. This layer takes in a d-dimensional input and produces a n dimensional output. The weights of these layers are typically *learned* using back-propagation and gradient descent Rumelhart *et al.* (1985).

The Convolution Layer

Fully-connected layers require a large amount of memory to store all their weights. They also involve a lot of computation. Therefore, they are not ideal for use as feature extractors for images. This is because, a dot product layer has a complete receptive field. The receptive field of a neuron is the spatial range of input flowing into the neuron. It is the range of the input that the neuron has access to to. In our definition of the dot-product layer, the receptive field of a neuron is essentially the length of

the entire signal. As we noticed earlier, most image features have small and local receptive fields that are typically a few tens of pixels such as 16×16. A convolution layer is a generalization of the fully-connected layer that has a small receptive field.

Locality arises in this connection as the input dimensions are organized spatially implying that adjacent dimensions of inputs are locally-related. Consider a neuron that has a reception of $r = 3$. Figure 1.4 demonstrates this connectivity. Each neuron in this arrangement, is capable of only detecting a pattern in an image that is local and small. While each location in an image could have spatially independent features, most often in images, we find that spatial independence doesn't hold. This implies that one feature learned from one location of the image ought to be reasonably assumed to be useful at all other locations.

Although Figure 1.4 shows the neurons are being independent, typically several neurons share weights. In the representation shown in Figure 1.4, all the neurons share the same weights . Even though we produce $n - r + 1$ outputs, we only use r unique weights. The convolutional layer shown here takes a $1D$ input and is therefore a $1D$ convolutional layer. Figure 1.5 shows a $2D$ convolutional layer. This figure does not show independent neurons and their connectivities. Instead, it shows a convolutional filter that slides over the entire image. In other words, the layer's weight is shared across all locations in the image. In cases where the input has more than one channel, we convolve along all channels independently and the outputs are summed location-wise.

The $2D$ convolution layer typically performs the following,

$$z(j, d_1, d_2) = \alpha \left[\sum_{c=0}^{C-1} \sum_{u=0}^{r-1} \sum_{v=0}^{r-1} x_{c,d_1+u,d_2+u} \times w_{u,v}^{j} \right], \\ \forall j \in [0, 1, \ldots n] \text{ and } \forall d_1, d_2 \in [0, 1, \ldots d], \quad (1.9)$$

where, the weights $\mathbf{w} \in \mathbb{R}^{j,r,r}$ are j sets of weights, each set being shared by several

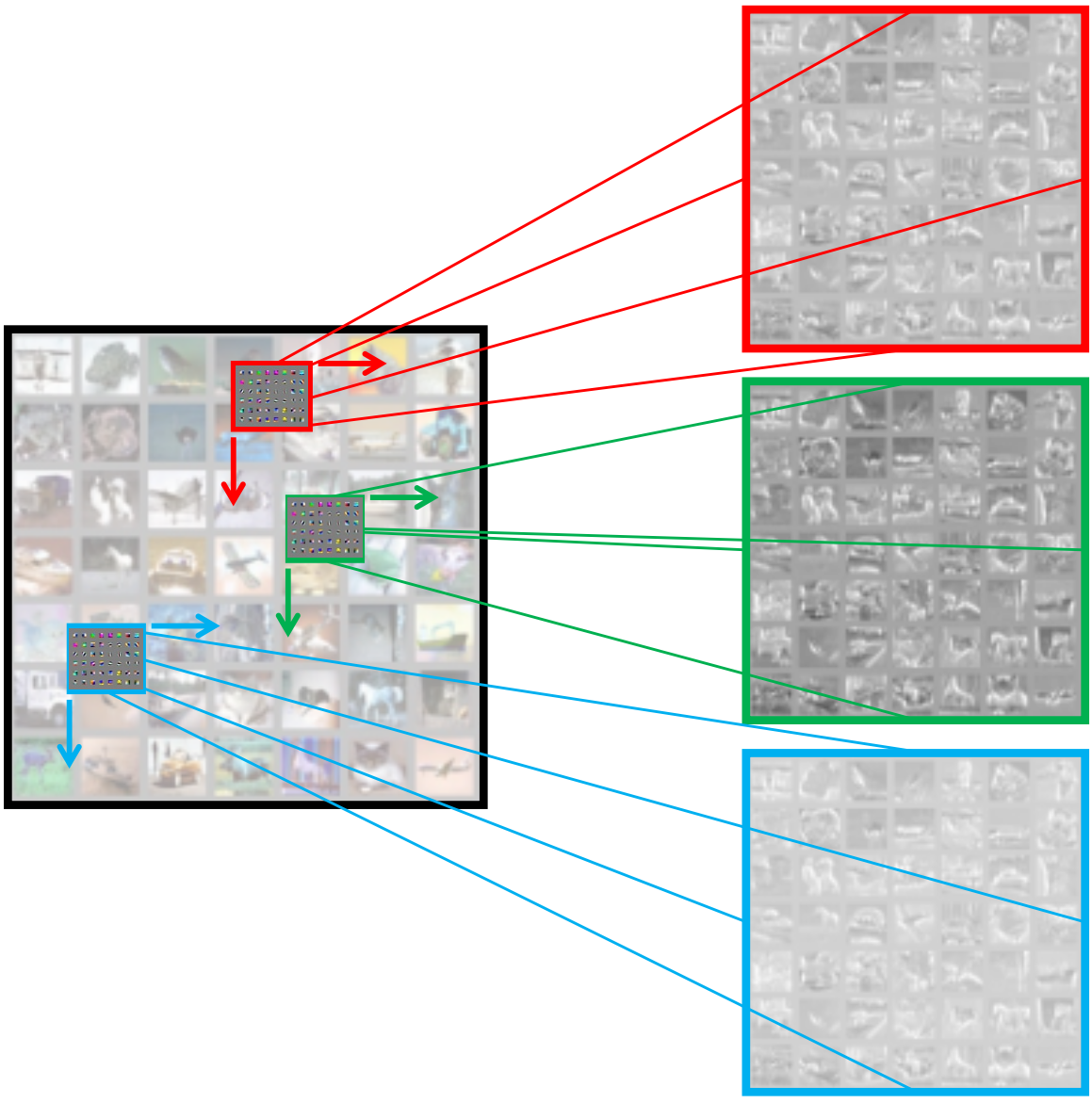


Figure 1.5: A Typical Convolution Layer

neurons, each with a receptive field of r working on an input $x \in \mathbb{R}^{d_1 \times d_2}$. Since we have j sets of weights shared by several neurons, we will produce j activations that are essentially *images* each of size $\mathbb{R}^{d-r+1 \times d-r+1}$.

In the context of convolution layers, the we also refer to the outputs as feature maps. Figure 1.5 shows feature maps being generated at the end of a typical layer. The convolutional layer's filters can also be *learned* by back-propagation and gradient descent. Once learned, these filters will work as pattern detectors, locally. The feature map is a spatial map of confidence values for the existence of the local pattern, the filter has adapted to detect, at each location.

The pooling layer

The convolution layer creates feature maps that are of size $d - r + 1$ on each axis. Activities that are spatially adjacent in each of these feature maps are related to each other. We want to avoid storing (and processing) all activations and use only the most prominent of these features.

This is typically accomplished by using a sub-sampling operation. This is also often referred to as pooling and is done using non-overlapping sliding windows, where each window will sample one activation. In CNNS, pooling by maximum (max-pooling) is typically preferred. Note that pooling by p (widow size of p) reduces the sizes of activations $p-$folds.

1.2.2 CNN Architecture Design

Akin to the design involved in hand-crafted features and in most of machine learning, CNNs also involve some level of skilled hand-crafting. Most of designing involves the the architecture of the network and parameters of optimization such as choosing the types and number of layers and types of activation functions and number of neurons in each layer. One critical design choice that arises particularly in image

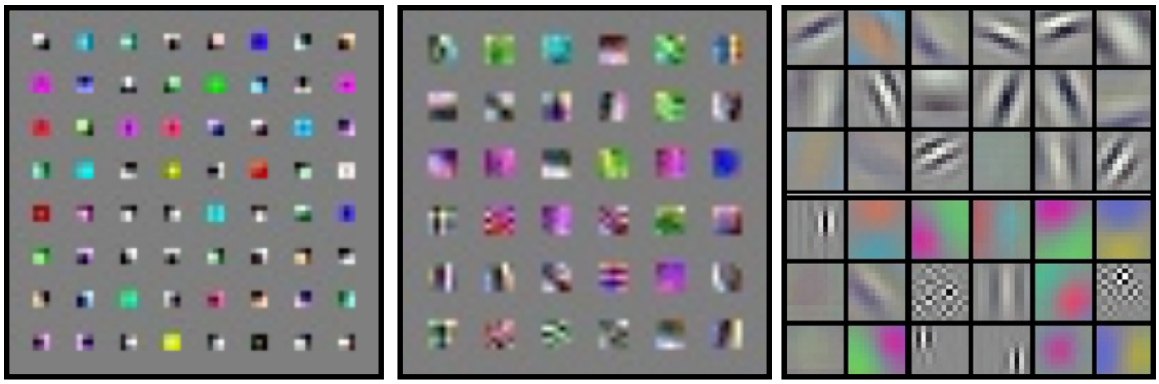


Figure 1.6: Some filters learned by various CNNs at the first layer that is closest to the image. From left to right are filters learned by VGG (3×3), a typical Yann LeCun style LeNet (5×5) and a AlexNet (11×11).

data and CNNs, is the choice of the receptive fields.

The receptive field can be controlled by the size of filters (weights) in each layer and by the use of pooling layers. The effective receptive field (in relation to the input image) increases after each layer as the area of the signal received from the input layer expands progressively. There are generally two philosophies guiding the choice of filter sizes and therefore to the receptive fields. The first was promoted by Yann LeCun et al., LeCun *et al.* (1998a, 1990) and was later re-introduced and widely preferred in modern day object categorization by Alex Krizhevsky et al., Krizhevsky *et al.* (2012). They employ a *relatively* large receptive field to begin with at the earlier layers and keep contining growing with the rate of growth reducing by a magnitude.

The second of these philosophies begins with a relatively small receptive field and increases it as minimally as possible. These were pioneered by the VGG group Simonyan and Zisserman (2014a).

Although we studied some popular network architecture design and philosophies, several other styles of networks also exists. Here we briefly only studied those that feed-forward from the input layer to the task layer (whatever that task might be) and there is only one path for gradients to flow during back-propagation. Several

networks such as the GoogleNet Szegedy *et al.* (2015) and the newer implementations of the inception layer Szegedy *et al.* (2016, 2017), Residual Net He *et al.* (2016) and Highway Nets Srivastava *et al.* (2015) have been proposed that create networks with multiple feed-forward paths allowing for more than one path to the target. One of these paths can even involve directly feeding forward the input signal. This therefore allows for the gradient to not vanish Bengio *et al.* (1994).

Chapter 2

NON-PARAMETRIC MULTIPLE INSTANCE LEARNING

2.1 Introduction

Multiple-instance learning (MIL) is a setting where labels are provided only for a collection of instances called bags. There are two types of instances: negative instances, which are found in either negative bags or positive bags, and positive instances, which are found only in positive bags. While a positive bag should contain at least one inherently positive instance, a negative bag must not contain any positive instances. In MIL, labels are not available at the instance level. It is interesting to note however that the label-space is the same for both at the bag level and at the instance level. One may attempt to learn instance-level labels during the training stage, thus reducing the problem to an instance-level supervised classification. Alternatively, one may also localize and prototype the positive instances in the feature space and rely on the proximity to these prototypes for subsequent classification.

MIL is an ideal set-up for many computer vision tasks and examples of its application include object tracking Babenko *et al.* (2011), image categorization Chen and Wang (2004) Wang *et al.* (2011a) Wang *et al.* (2012a) Felzenszwalb *et al.* (2010), scene categorization Maron and Ratan (1998) and content-based image retrieval Zhang *et al.* (2002). In particular, MIL can be an especially suitable model for medical image-based pathology classification and lesion detection-localization, where an image is labeled pathological just because of one or a few lesions localized to small portions of the image. Figure 2.1 illustrates such an example: color fundus images of eyes affected with different pathologies of diabetic retinopathy (DR). It is easy to notice

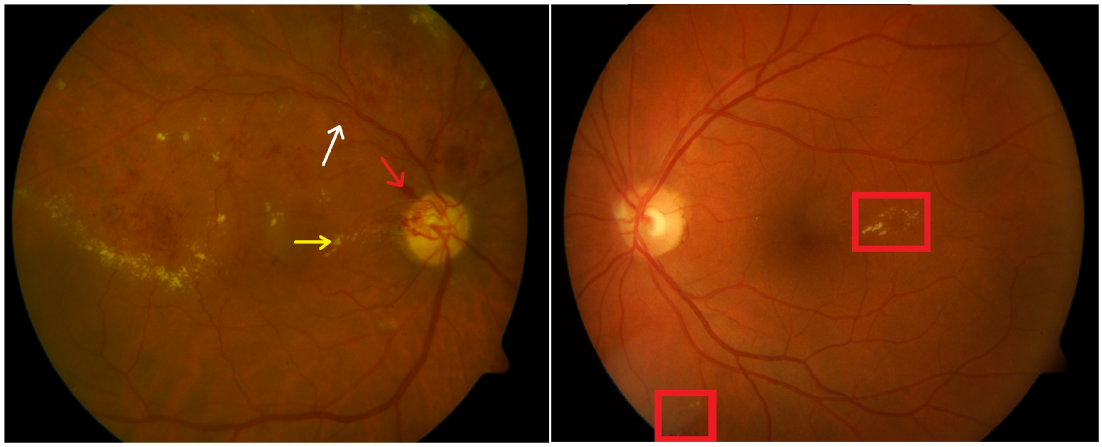


Figure 2.1: DR Image Classification as a MIL Problem. © 2015 IEEE.

that, although majority of the image looks normal, a small retinal landmark is enough to alter the label of the image from normal to pathological. In a MIL formulation for this problem, each image can be considered a bag and patches of images can be considered instances.

Over the years, many methods have been proposed to solve the MIL problem Dietterich *et al.* (1997) Wang and Zucker (2000) Chen *et al.* (2006) Andrews *et al.* (2002). The most fundamental one is the diverse density approach Maron and Lozano-Pérez (1998), which has been built upon by many variants Zhang and Goldman (2001) Rahmani *et al.* (2008) Chen and Wang (2004). Diverse density is in its basic sense, a function so defined over the feature space such that it is high at any point in the feature space that is close to instances from positive bags while being far away from instances from negative bags and vice-versa. The various local maximas in this function are positive instance prototypes and any instance that is closer to these prototypes are labeled inherently positive instances. Other types of methods also exist in this setting Bergeron *et al.* (2012) Antić and Ommer (2013) Wang *et al.* (2011b) Wang *et al.* (2012c).

MIL has many different variants and perspective to its definition and indeed most

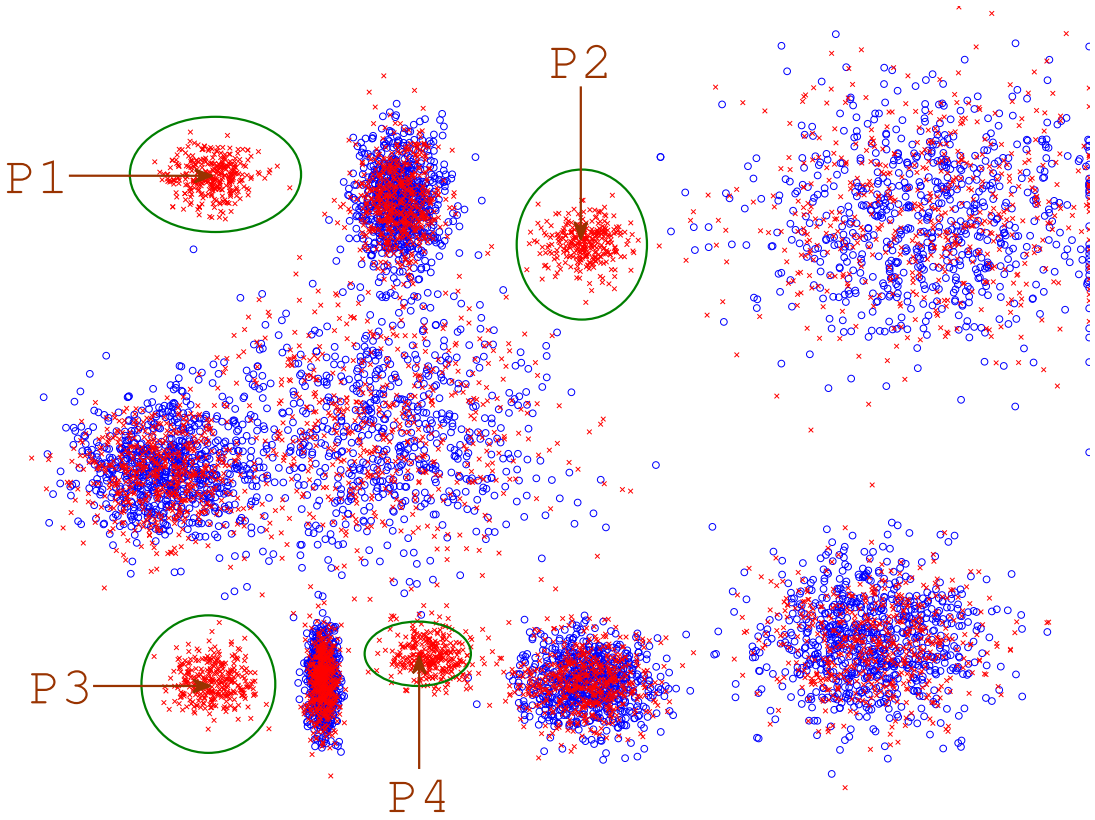


Figure 2.2: An illustrative feature space for multiple-instance setting. The 'x' in red represents all instances from positive bags and the 'o' in blue represents all instances from negative bags. © 2015 IEEE.

MIL solutions are application centric Amores (2013). This can be easily seen from table 2.1. *Earlier methods perform as good or better in the MUSK dataset than the ones published recently although the recent methods perform better on more complex tasks* but for certain exceptions. In this course of research while many particular and complicated solutions are sought after, MIL has never been sufficiently analyzed using traditional non-parametric learning methods. Despite the recent advances, MIL remains a challenging task as the feature space may be arbitrarily complex, the ratio of positive to negative instances can be arbitrarily low in a positive bag, and (by definition) no labeling information is directly available for positive instances.

To illustrate these factors, we simulate a typical MIL feature space as depicted in figure 2.2. Each instance belonging to a particular cluster is independently drawn from a normal distribution that defines the said cluster. While positive bags can draw a subset of random cardinality of instances from negative distributions, negative bags cannot draw any data from positive distributions. Every positive bag must have at least one instance sampled from a positive distribution (marked in green ellipses $P1$ through $P4$). The centroids of these clusters would be the ideal positive instance prototypes that a MIL algorithm should identify. With the help of this illustration, it is not difficult to imagine that, one or few noisy negative instances coming close to a true positive instance prototype could lower the diverse density drastically and thus lead to a dramatic decrease in performance, and herein lies a core argument to the MIL definition - the strictness of positive neighbourhood. We show that DD-based algorithms are not tolerant even to a single negative instance in an arbitrary positive instance neighbourhood. Such strict assumptions are not suitable for real-world (medical imaging) data wherein the feature space can be noisy.

2.2 Related Works

MIL was first introduced for the problem of drug activity prediction Dietterich *et al.* (1997), where axis-parallel hyper-rectangles (APR) were used to design three variants of enclosure algorithms. The APR algorithms tried to surround at least one positive instance from each positive bag while eliminating any negative instances inside it. Any test bag was classified positive as long as it had at least one instance within the APR. Conversely a bag was classified as negative when it had no instance represented within the APR.

The first density-based formulation of MIL was diverse density (DD) Maron and Lozano-Pérez (1998). DD is not a conventional density but is rather defined as the

intersection of the positive bags against the intersection of the negative bags. It is a measure that is high at any point on the feature space x if x is closer to positive instances *and* is farther away from negative instances. The local maxima of DD would yield a potential concept for the positive instances. Several local maxima can yield several prototypes of positive instances that can be far apart in the feature space. Some of these prototypes can be separated by other negative instances. The concept point of a diverse density in a MIL feature space was defined as,

$$\arg\max_{x} \prod_{i} Pr(x = t|B_i^+) \prod_{i} Pr(x = t|B_i^-). \tag{2.1}$$

These local maxima were termed as instance prototypes. A noisy-or model was used to intuitively maximize the DD in Equation 2.1. This was further developed to assume more complicated and disjoint concepts in EMDD and further developed by other methods including DD-SVM and Accio Zhang and Goldman (2001) Chen and Wang (2004) Rahmani *et al.* (2008). The major drawback of the diverse density arises in a situation where the distribution of negative instances is noisy. In other terms, if one instance prototype has a negative distribution closer to the prototype than the others, then its diverse density is largely lower than that of the others, as DD unfairly favours the distribution of positive instances that is farther away from negative instances than those that are relatively closer. This makes it hard to define that particular prototype in such situations. Even the presence of one noisy negative instance near the potential instance prototype can lower the DD drastically as we show in the later sections. In figure 2.2 the prototype $P4$ was the twenty second largest local maxima in the DD of the feature space. If there were a bag that contained only one positive instance near $P4$ but was still close enough to the negative instances, chances are that this bag will be misclassified as negative. DD defined in such a formulation provides a density-like function that is fickle and is easily affected by introducing even just one

negative sample closer to the positive prototype.

The maximization procedure for DD is started from initial guesses. An idea was put forward by Chen and Wang that the maximization should start from every instance in every positive bag (or at least a large sample of positive bags) so that unique local maxima in DD can be identified Chen and Wang (2004). A plethora of methods still use this DD formulation Chen and Wang (2004) Chen *et al.* (2006) Rahmani *et al.* (2008) Zhang and Goldman (2001) Li *et al.* (2013). The decision boundary of a DD system is a hyper-ellipsoid in the feature space. A kernel based maximum-margin approach would construct hyper-planar decision boundaries characterizing complex decision surfaces. The first formulation of a support vector machine (SVM) for MIL was proposed in 2002 Andrews *et al.* (2002). They devised an instance-level classifier *mi*-SVM and a bag-level classifier *MI*-SVM. In a way, *MI*-SVM maximized the margin between the most positive instances and the least negative instances in positive and negative bags respectively. The *MI*-SVM framework is now modified and re-christened as *latent*-SVM which plays a central role in the deformable-part models based object recognition algorithms Felzenszwalb *et al.* (2010). MILIS provided a similar SVM-based approach with a feedback loop to select instances that provided a higher training stage confidence Fu *et al.* (2011). This was an idea adapted from a previously existing related idea, MILES Chen *et al.* (2006).

The first distance-based non-parametric, lazy learning approach to MIL was taken by citation-k-NN Wang and Zucker (2000). Inter-bag distances were found using a *minimal* Hausdorff distance. A k-nearest neighbour approach was used along with this distance to classify a new bag or to retrieve closer bags. This did not always work in a MIL setting as k-NN uses a majority voting scheme. If a positive bag contains fewer number of inherently positive instances than inherently negative instances, majority of its neighbours are going to be negative and the algorithm was confused by the

false-positives it reported. Therefore the concept of *citers* was introduced. If k-NN *refereed* its neighbours, then its neighbours are *cited* by citers. Citers are the backward propagated references, in the sense that they refer back the considered instance. Though it was a generalized approach, citation k-NN did not work as well when positive instances were clustered and such clusters were separated by negative instances, in which case the citers and references did not always compliment each other.

This problem does not apply to all nearest neighbour based approaches. Nearest neighbour approaches should be used properly and their smart usage was discussed in Boiman *et al.* (2008). A novel concept of bag to class (B2C) distance learning was adopted for the use of k-NN. A complimentary idea was utilized in a MIL set-up by learning class to bag (C2B) distances by combining all training bags of a particular class to form a super-bag Wang *et al.* (2011a) Wang *et al.* (2012c). A similar instance specific distance learning approach was used in Wang *et al.* (2011b). On further study, this was reformulated as a $l_{2,1}$ *minimax* problem and was solved with some effort Wang *et al.* (2012a). A similar idea was implemented to group faces in an image by considering inter bag or bag to bag (B2B) distances in Guillaumin *et al.* (2010). A related bag to bag approach is used to quantify super-bags in Antić and Ommer (2013).

Most of the MIL algorithms presented above assume that the bags are independent. Though it is a reasonable assumption in a computer vision context, it might not be a general idea. Zhang et al., explored the MIL idea for structured data Zhang *et al.* (2011). A *data-dependent* mixture model approach was developed in Wang *et al.* (2012b). Another approach designed specifically for special data space is the fast bundle algorithm for MIL Bergeron *et al.* (2012). One important assumption in the early understanding of MIL is that every positive bag must contain at least one

positive instance. Chen et al. felt this was too restrictive and developed a feature mapping using instance selection that projects a MIL problem into a much simpler supervised learning problem using an instance similarity measure Chen *et al.* (2006). This counter-assumption was also used in a histopathology cancer image learning system using a multiple clustered instance learning approach Xu *et al.* (2012). Although in a MIL formulation bag level classification is sufficient and instance level classification though clever, is not required, many algorithms attempt to identify positive instances. A SVM was used to minimize the *hinge* loss (modeled as slack variables) to identify positive instances in Wu *et al.* (2009). The above methods cater to certain particular configurations of the MIL space and are suitable for particular domains.

2.3 The Non-parametric MIL Approach

Consider figure 2.2. Though not universal, this figure illustrates a typical MIL feature space. The instances arising from regions $P1$ to $P4$ are potentially inherently positive instances as they are farther away from negative instances while being closer to other positive instances. The instances from positive bags in other regions, along with negative instances are in reality, negative instances as they rub shoulders with negative instances from negative bags.

Suppose we have labeled data $\mathcal{D} = \begin{bmatrix} X^{(1)} & Y^{(1)} \\ X^{(2)} & Y^{(2)} \\ X^{(3)} & Y^{(3)} \\ \vdots & \vdots \\ X^{(n)} & Y^{(n)} \end{bmatrix}$ where $X^{(i)}$ is the i^{th} bag in the dataset and $Y^{(i)} \in \{0, 1\}$ is its label. Internally, each bag $X^{(i)}$ contains m_i (often is a constant m by design, particularly in image classification contexts) instances such that $X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots x_{m_i}^{(i)}\}$. Consider a small region R of volume V in this

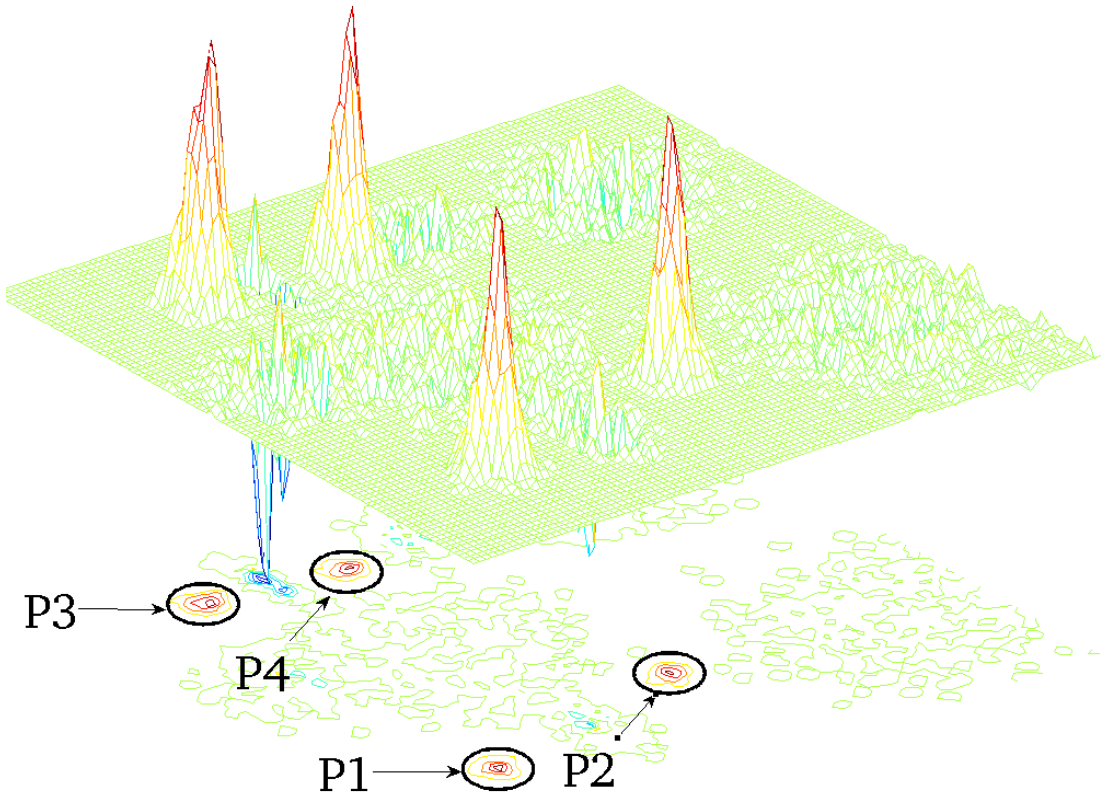


Figure 2.3: Parsing the MIL feature space with a Parzen window technique. It can be seen that this follows the properties of a MIL density-like. © 2015 IEEE.

feature space. The estimate for the density of instances from positive bags is given by $\frac{(|k^+|)/n}{V}$, where k^+ is the set of instances from positive bags in the region R and $|k^+|$ its cardinality, and n is the number of instances in all of the feature space. Similarly the estimate for the density of negative instances is given by $\frac{(|k^-|)/n}{V}$, where k^- is the set of instances from negative bags in the region R, $|k^-|$ is the number of negative instances in the region R.

Putting them together, $\frac{(|k^+|)/n}{V} - \frac{(|k^-|)/n}{V}$ is a measure that, will be high if the number of positives exceed the number of negatives in that region, will be low if the number of negatives exceed the number of positives in that region, and will be 0 if the number of positives equal the number of negatives within that region. Alternatively,

if one considers a (rectangular) Parzen window,

$$\phi(u) = \begin{cases} 1, & |u_j| \leq h \text{ where, } j = 1, 2, ...d, \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

the aforementioned measure can also be formulated as,

$$f_{parzen}(x) = \frac{1}{n} \sum_{i=1}^{|k_n^+|} \frac{1}{V} \phi(\frac{x - k_i^+}{h}) - \frac{1}{n} \sum_{i=1}^{|k_n^-|} \frac{1}{V} \phi(\frac{x - k_i^-}{h}) \tag{2.3}$$

where, x is any location on the feature space and k_i^+ and k_i^- are instances from positive and negative bags within that region respectively. Such a parsing of the MIL feature space of figure 2.2 is shown in figure 2.3. The properties of the function $f_{Parzen}(x)$ hold similar to that of DD and can be easily observed in figure 2.3. The choice of the size of the region (analogous to the selection of the variance for the Gaussian in the DD formulation) and the Parzen window functions are in line with that of a traditional Parzen window: if the size becomes too large, the measure will not have sufficient resolution. Picking a proper region-size would be a practical difficulty.

Instead of considering a region R of fixed size, let's limit to a fixed number of neighbours k. In this set-up, we start with a region of zero volume from x and grow two regions, one for positive instances and one for negative instances, until we just enclose for each of the regions, k points respectively. This enables us to have different sized regions for positive and negative estimates respectively. In fact, a direct application of nearest-neighbour voting technique will not work on a MIL space as was pointed out by Wang et. al, but the idea of nearest neighbour can still be modified and used to suit the MIL needs Wang and Zucker (2000). The vote contributions of positive and negative neighbours enclosed by the two regions are their respective kernelized distances to the point x, instead of a uniform majority

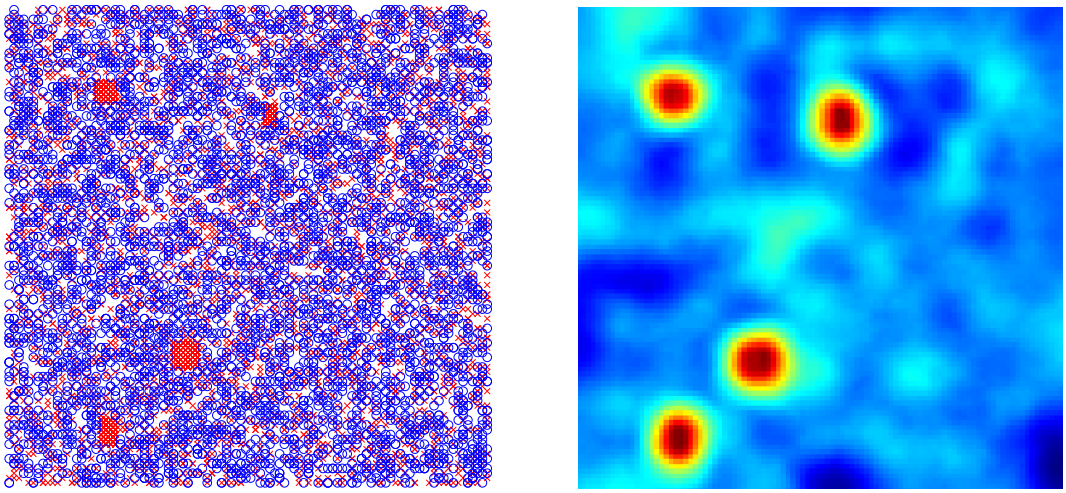


Figure 2.4: A region of a typical 2D MIL feature space and its parse using the k-NN measure. Red represents positive and blue represents negative. © 2015 IEEE.

vote. This aggregated vote can be formulated as,

$$f_{kNN}(x) = \sum_{i=1}^{|k^-|} \Psi(||x - k_i^-||) - \sum_{i=1}^{|k^+|} \Psi(||x - k_i^+||) \text{ such that, } |k^+| = |k^-| = k. \quad (2.4)$$

where, $\Psi(.)$ is a monotonically increasing sub-modular function, k is the number of neighbours considered, and k^+ and k^- are now the set of k instances from positive and negative bags that are the nearest to x respectively. $\Psi(.)$ is used as a way to scale distances when the featurespace is arbitrarily large. It can be considered as normalization. For all our experiments, it is typical to use $\Psi(x) = x$.

The advantage of fixing the number of neighbours is that in a region where there are no points or very few number of points, we will get a block of uniform measure and in a region where there is a high density of points, we will get a smoothly varying measure. Such a measure is shown in figure 2.4. The impact of the number of neighbours k is similar to that of the size of the region R in the Parzen window idea. If k is too small, the measure is going to give information about a very small local region and is thereby unreliable. If k is too large , the impact of proximity is going to be averaged out.

2.3.1 Learning Under This Formulation

Learning under this formulation is a straight forward threshold learning and this is done by maximizing the validation accuracy. An instance-level classifier using this measure can be constructed as,

$$h(x) = \mathbb{1}\{f_{kNN}(x) \geq T\} \tag{2.5}$$

This is an indicator function that outputs 1 if the measure is above a threshold T and 0 if the measure is below the threshold T. We can use this instance-level classifier to construct a bag-level classifier.

$$b(X) = \mathbb{1}\{\sum_{i=1}^{m} h(x_i) \geq a\} \forall x_1, x_2, \ldots, x_m \in X. \tag{2.6}$$

This is an indicator function that classifies the bag 1 if it has at least a instances classified as positive and 0 other-wise. Typically in most MIL settings $a = 1$, although this need not be the case generally. The aim of this non-parametric empirical risk minimization formulation is to minimize the training error,

$$\hat{\epsilon(b)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{b(X^{(i)}) \neq Y^{(i)}\}, \quad \forall (X, Y) \in D. \tag{2.7}$$

by estimating $\hat{T}$ that best minimizes $\hat{\epsilon(b)}$ as,

$$\hat{T} = \arg\min_{T} \hat{\epsilon(b)} \tag{2.8}$$

Once the threshold is learnt, classification is performed directly by using the bag-level classifier in equation 2.6 with the learnt threshold. Note that in MIL, it is not required, although possible in this case, to label each instance in the bag. The labeling of instances can be as follows:

$$y(x) = h(x)|_{T=\hat{T}} \tag{2.9}$$

This process equivalent to maximizing the equation 2.15 (or 2.3) for all points of feature space and considering the local maximas as instance prototypes, as was described by Chen et. al, for the DD formulation Chen and Wang (2004). This now enables comparison to prototyping-based methods. Such a formulation can now be re-written as,

$$\hat{x} = \arg\max_{x} \left[\sum_{i=1}^{|k^-|} \Psi(||x - k_i^-||) - \sum_{i=1}^{|k^+|} \Psi(||x - k_i^+||) \right], \text{such that,} |k^+| = |k^-| = k. \quad (2.10)$$

where $\hat{x}$ is a prototype positive instance. One advantage of using equation 2.10 is that once the prototypes are found, we neither need the entire dataset anymore nor do we need to calculate distances to all the points in the dataset. The prototypes easily divide the featurespace into probabilistic Voronoi tessellations such as in figure 2.4, or we could estimate a radius around every prototype to isolate hyper-spherical regions that are positive. We solve this optimization problem by using an idea similar to the one used in Chen and Wang (2004).

We start a local gradient ascent from every instance from every positive bag in the training dataset and find a local maxima. Since such maximas can only ever end in a high density region of true positive instances from positive bags and since we start each gradient ascent from every instance in every positive bag, each ascent is computationally tractable in small number of iterations. Indeed, often few well-chosen instances from positive bags make this convergence faster and such techniques can be found for maximizing the DD. Similar techniques can be applied here as well. All the local maximas are sorted (after non-maximal suppression) and top N are considered as instance prototypes. It is to be noted that for the dataset shown in figure 2.2, while the top 5 maximas were enough to find all four prototypes for our approach, it takes top 24 maximas for DD to find the four prototypes.

Since the number of positive instances in a bag is usually arbitrarily low as com-

pared to total number of points on the feature space, this is generally not too many iterations and is computationally tractable.

The k for k-NN is picked here by a typical elbow method. Once local maximas (instance prototypes) are found we can again maximize a validation accuracy jointly for all instance prototypes to find a threshold of classification for each prototype in terms of the distance to the prototype, hence creating a hyper-spherical decision regions around each prototype. Thus the decision boundaries of this method creates a tessellation of the feature space. The tessellation is a set of hyper-spherical regions around a positive prototype with varying radii.

2.4 Experiments and Results

In this section, details of the experiments will be provided along with the results from those experiments. We evaluated the method using three standard MIL datasets: the musk dataset, Andrew's datasets, the Corel datasets (both 1k and 2k), and our own dataset: the DR dataset. For all the results shown on all the datasets, we used the most common implementation methodologies, including data splits, cross validations and average over runs that were found in literature. This enabled us to compare against results that were published in the same. When results were not available or when the protocol doesn't match, we evaluated the results using the codes from CMU MIL toolbox [1] . In case of MILES, the results were obtained by using the author's original code [2] . The results provided were obtained for best parameter settings using grid search.

[1]CMU MIL toolbox: `http://www.cs.cmu.edu/~juny/MILL`

[2]MILES homepage: `http://www.cs.olemiss.edu/~ychen/MILES.html`

2.4.1 Musk Dataset

An popular evaluation dataset in the MIL literature is the musk dataset. The musk dataset is well-described in Dietterich *et al.* (1997). Musk dataset is a benchmark feature space used to predict drug activity. MUSK 1 contains 92 molecules with 47 musk and 45 non-musk molecules. MUSK 2 contains 102 molecules with 39 musk and 63 non-musk molecules. Each bag contains variable number of instances with 166 dimensional features and binary labels. We use the standard implementation specifications that is used in the original APR paper and other published literature: ten-fold crossvalidation over the entire dataset, since its easier to compare against a plethora of methods Dietterich *et al.* (1997). Table 2.1 compares the performance of various algorithms against the proposed method. It can be seen that the proposed method is best in MUSK 1 and among the high performing methods in MUSK 2. MUSK datasets are uni-concept datasets. For instance, in MUSK 1, among a total of 476 unique instances each with feature values ranging from -348 degrees to 336 degrees, there are only 633 unique feature values. In such a heavily quantized feature space that is 166 dimensional, detecting one potential instance prototype is easier for density based algorithms. Our method while being the best in the MUSK 1 data, is also among the better in the MUSK 2 data.

2.4.2 Andrew's Datasets

Andrews et. al, in their *mi*-SVM paper proposed the use of three classification datasets, *elephant*, *fox* and *tiger*, for the use of evaluating multiple-instance learning Andrews *et al.* (2002). These are now popular benchmark datasets in the MIL literature. We also test the algorithm on these datasets using the same specifications mentioned on the said article. Each dataset has 200 images with 100 positive and

Methods	MUSK 1	MUSK 2
DD Maron and Lozano-Pérez (1998)	88.9%	82.5%
EM-DD Zhang and Goldman (2001)	84.8%	84.9%
citation (k)-NN Wang and Zucker (2000)	**92.4%**	86.3%
mi-SVM Andrews *et al.* (2002)	87.4%	83.6%
MI-SVM Andrews *et al.* (2002)	77.9%	84.3%
DD-SVM Chen and Wang (2004)	85.8%	**91.3%**
MILES Chen *et al.* (2006)	86.3%	87.7%
MIforest Wang *et al.* (2012c)	85%	82%
MILIS Fu *et al.* (2011)	88.6%	91.1%
ISD Wang *et al.* (2011b)	85.3%	79.0%
ALP-SVM Antić and Ommer (2013)	87.9%	86.6%
MIC-Bundle Bergeron *et al.* (2012)	84%	85.2%
Ensemble Li *et al.* (2013)	89.22%	85.04%
Proposed	**92.4%**	86.4%

Table 2.1: Performance of Various MIL Algorithms On the Musk Dataset. © 2015 IEEE.

100 negative images. The number of instances in each category are 1391, 1320 and 1220 respectively with varying number of instances per bag. Each instance is a 230 dimensional feature vector. We train on a 2/3 random split of the data and test on the remaining 1/3 of the unseen data. The results are maximized over 15 runs of validation and The results are shown in table 2.2. Our result while being the best in the Elephant and Fox classes is almost as good as the best in the Tiger class. It is to be noted that we are significantly higher in the Fox class which is widely considered to be a notoriously noisy dataset for MIL ergo a strong indicator of the proposed

Methods	Elephant	Fox	Tiger
citation k-NN Wang and Zucker (2000)	79.2%	62.5%	82.6%
mi-SVM Andrews *et al.* (2002)	79.7%	62.9%	79%
MILES Chen *et al.* (2006)	70%	56%	62%
MIforest Wang *et al.* (2012c)	84%	64%	82%
ISD Wang *et al.* (2011b)	77.9%	63%	85.3%
ALP-SVM Antić and Ommer (2013)	84%	69%	**86%**
MIC-Bundle Bergeron *et al.* (2012)	80.5%	58.3%	79.11%
Ensemble Li *et al.* (2013)	84.25%	63.05%	79.30%
Proposed	**86%**	**73.94%**	85.7%

Table 2.2: Performance of Various MIL Algorithms on Andrew's Dataset. ©2015 IEEE.

method's adaptability.

2.4.3 Corel Dataset

Corel is another well known, image categorization dataset for MIL benchmarking. The Corel-2k dataset consists of 2000 images. There are 20 classes and each class consists of 100 images. The Corel-1k dataset is a subset of this dataset with the first 10 difficult categories. Table 2.3 shows the performance of the proposed approach in the corel dataset. It is to be noted that we are producing the best results in the Corel dataset. Training-testing data is again a $2/3 - 1/3$ split.

2.4.4 A DR Dataset

As was briefly discussed in section 2.1, DR image classification is an application especially suitable for MIL. In practice, the difficulty in this problem arises from the fact

Methods	Corel-1k	Corel-2k
mi-SVM Andrews *et al.* (2002)	76.4%	53.7%
MI-SVM Andrews *et al.* (2002)	75.1%	55.1%
MILES Chen *et al.* (2006)	82.3%	68.7%
DD-SVM Chen and Wang (2004)	81.5%	67.5%
MILIS Fu *et al.* (2011)	83.8%	70.1%
Proposed	**87.3%**	**71.9%**

Table 2.3: Performance of Various MIL Algorithms on Corel Dataset. ©2015 IEEE.

that the physical and observable difference between a normal eye and a pathological eye can be very small, localizing to regions with slightly different characteristics. This can be seen in figure 2.1. A variety of classification and retrieval schemes have been tried on DR images. Structural Analysis of the Retina (STARE) is one of the earliest attempts to solve the DR conundrum McCormick and Goldbaum (1975) Goldbaum *et al.* (1989). STARE performs automated diagnosis and comparison of images to search for images similar in content. Recently other learning approaches were developed to identify relevant patterns using local relevance scores Quellec *et al.* (2012b). Application of MIL approaches to DR is gaining interest in recent years Quellec *et al.* (2012a). In this study, we consider the auto color correlogram (AuoCC) as a color feature, which is well-studied in the medical imaging literature Huang *et al.* (1997). A modified and quantized 64-bin AutoCC feature is extracted for each instance in an image. We neglect the black regions and sample 48 non-overlapping instances from every image. We use a high-quality color fundus image database of 425 images comprising 160 normal images, and 265 affected images to test the algorithm on. This dataset was constructed from publicly available databases including DiabRetDB0 et al. (2005), DiabRetDB1 Kauppi *et al.* (2007), STARE McCormick and

Goldbaum (1975) and Messidor [3] and has been used in some existing studies Chandakkar *et al.* (2013a) Venkatesan *et al.* (2012a). The balance of the database is more towards the positive bags and this makes it more challenging for a MIL algorithm. The results were all evaluated using a $2/3 - 1/3$ train-test split.

Prototyping DR Instances

In the prototyping sense, each prototype of positive instances should roughly correspond to one type of lesion. As we use color features this is easily possible. We estimated a total of about 35 different types of lesion prototypes using the algorithm and verified it with EM-DD's prototypes. EM-DD had its maximum accuracy at about 40 prototypes. It is reasonable to assume from this information that there is somewhere between 35-40 different positive prototypes, each of which in the feature space might correspond to a unique lesion type or character. In this feature space, the negative instances are of three types: normal skin, nerves and the optical disk. This is a reasonably noisy datasets and often has only one or two instances among 48 instances that are positive in a positive bag. Though the distribution of the optic disk might be noisy, and the number of true positive instances are very low, the proposed algorithm has the potential to adjust to it. Table 2.4 shows the results of the proposed approach on the DR dataset, where the proposed method stands best.

2.4.5 Sensitivity to Labeling Error

Although not an implicit feature of the proposal, we perform the experiments to demonstrate the proposed method's sensitivity to labeling error, exactly similar to the one described in Chen *et al.* (2006). We deliberately flip the labels for a range of percentages of labels randomly on our training split and test the trained model

[3]Kindly provided to us by the messidor program partners. Visit `http://messidor.crihan.fr`

Methods	Accuracy
DD Maron and Lozano-Pérez (1998)	61.29%
EM-DD Zhang and Goldman (2001)	73.5%
citation k-NN Wang and Zucker (2000)	78.7%
mi-SVM Andrews *et al.* (2002)	70.32%
MILES Chen *et al.* (2006)	71%
Proposed	**81.3%**

Table 2.4: Performance of various MIL Algorithms on DR Dataset. ©2015 IEEE.

on the original labels in the testing split. The split was $2/3 - 1/3$. The accuracies of the proposed method on various datasets are shown in figure 2.5. After about 20% of labels are corrupted, the proposed method still loses only about 5% accuracy and only when about one-third of the labels are corrupted, the proposed method loses about 10% accuracy. The average drop in accuracy for both the proposed method and MILES are compared in figure 2.6. It is clear that MILES and the proposed algorithm follow the exact same trend. This trend is clearly indicative that the proposed method is *as good as* MILES and is often times better, when it comes to sensitivity to labeling noise. It is noteworthy that MILES is considered the state-of-the-art benchmark for sensitivity to labeling error out of all MIL methods published and that was one of its core contributions.

2.5 A Simple Case Study Describing the Effectiveness of the Proposed Method

In this section we demonstrate by a case-study the strictness of a DD positive neighbourhood.

Consider two bags B^+ and B^- being positive and negative labeled respectively. Consider the instances in the bags as such: $B^+ = \{p, \alpha\}$ and $B^- = \{\alpha, \alpha\}$; such

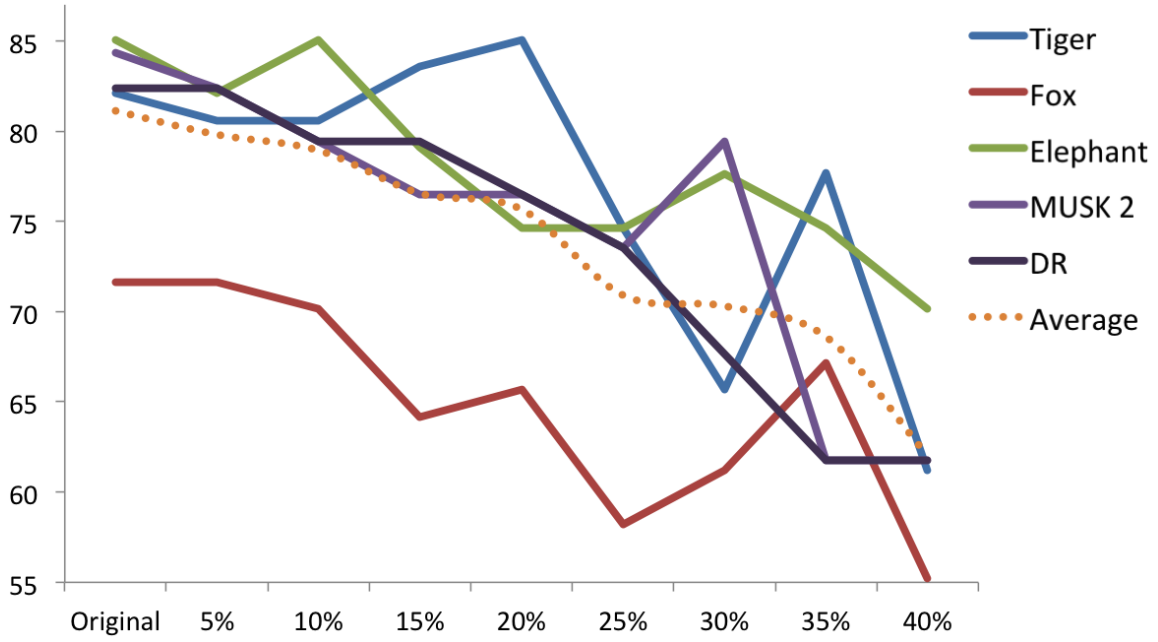


Figure 2.5: Accuracy vs Percentage of Labels Flipped for the Proposed Method. Flatter curve is good. ©2015 IEEE.

that α be any instance that is so far away form p so that $||p - \alpha|| = \Phi$ where Φ is a large constant and $e^{(-||p-\alpha||)} = 0$. Any instance prototype for a positive instance is therefore at p.

Diverse density at any point x is defined by $f_{DD}(x)$,

$$f_{DD}(x) = \prod_{i=1}^{n} Pr(x = t|B_i^+) \prod_{i=1}^{m} Pr(x = t|B_i^-) \tag{2.11}$$

for n positive and m negative bags in the data space. Assuming independence between instances, and using the noisy-or model, equation 2.11 can be decomposed to:

$$f_{DD}(x) = [1 - (1 - Pr(x = t|p))(1 - Pr(x = t|\alpha))](1 - (Pr(x = t|\alpha))^2 \tag{2.12}$$

DD models the probability $Pr(x = t|i)$ where i be any instance as, $e^{(-||i-x||^2)}$. On the original data space, equation 2.12 becomes,

$$f_{DD}(x) = [1 - (1 - e^{(-||p-x||)})(1 - e^{(-||\alpha-x||)})](1 - e^{(-||\alpha-x||)})^2 \tag{2.13}$$

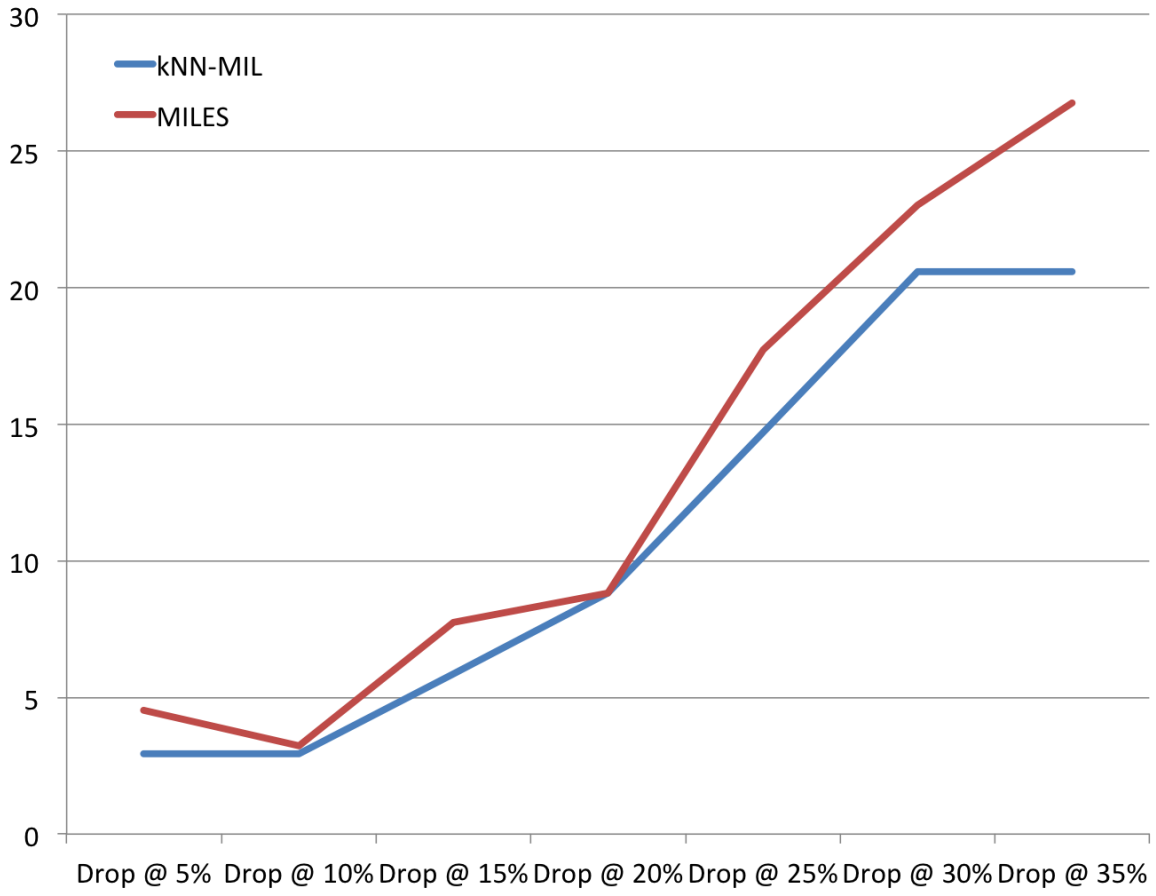


Figure 2.6: Drop in accuracy at various noise levels for proposed and MILES on the DR dataset. The lower the value the b

This equation can be solved at $x = p$ and at $x = \alpha$. At $x = p$ equation 2.13 becomes, $f_{DD}(p) = [1-(1-1)(1-0)](1-0)^2 = 1$. For the case of N positive and M negative bags with each positive bag containing only one positive instance each, the above measure will be, $f_{DD}(p) = M$ which is also the same case if there were M instances in the one negative bag. Note that DD is not a true density measure. Similarly at $x = \alpha$, $f_{DD}(\alpha) = [1-(1-0)(1-1)](1-1)^2 = 0$, which is also true for the many bags situation. Therefore unless $M = 0$ (no negative bags at all) DD will still be maxima at positive instance prototype.

Adding one additional negative bag with only one instance ($B^* = \{\beta\}$) to the

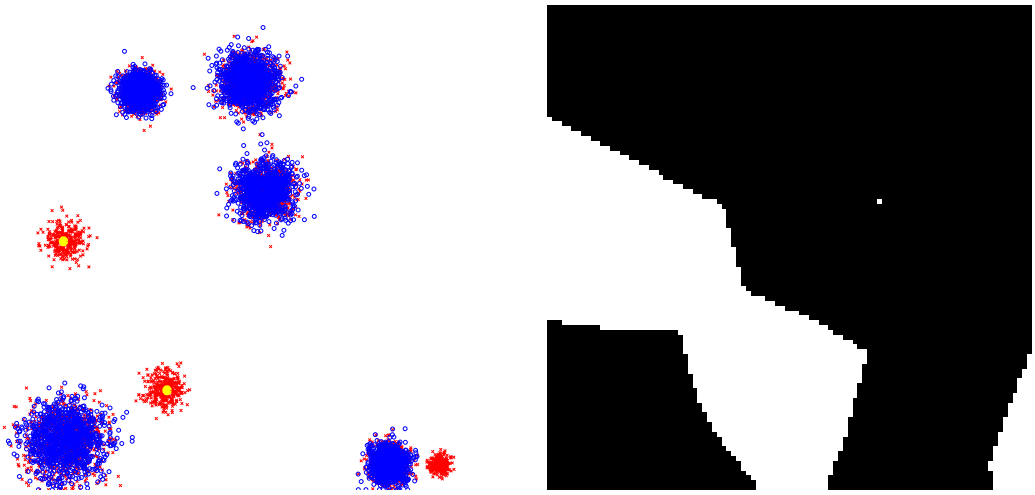


Figure 2.7: While the EMDD algorithm fails to capture one of the prototypes(left), the proposed method does and classifies that region as positive. Tessellation of the feature space by the proposed method is shown in the right. The accuracy for the proposed method is 100% while the accuracy of EMDD is 77.4%.

existing database, equation 2.13 becomes,

$$f^*_{DD}(x) = [1 - (1 - e^{(-||p-x||)})(1 - e^{(-||\alpha-x||)})] \\ (1 - e^{(-||\alpha-x||)})^2(1 - e^{(-||\beta-x||^2)}) \quad (2.14)$$

At $x = p$, this equation yields,$f^*_{DD}(p) = 1 - e^{(-||\beta-p||^2)}$.

This is a function that is exponentially decreasing in the order of the distance between β and p. The closer the β is to p, the exponentially lower the function is going to become and less the difference will be between, $f^*_{DD}(p)$ and $f^*_{DD}(\alpha)$ as at $x = \alpha$, the equation still remains at $f^*_{DD}(\alpha) = 0$. We also find that $\lim_{\beta\to p} f^*_{DD}(p) = 0$ thus nullifying the prototype as $f^*_{DD}(\alpha)$ is also 0. Although in the strict definition of MIL, such a point is not to be considered a MIL prototype, the belligerent instance could have been noisy. This is truly the case in figure 2.7 for instance, where due to the presence of a large cluster of negative points next to a positive cluster which contains just one negative point, EMDD cannot identify the positive cluster at all [4] .

[4]We use EMDD to maximize the DD for instance prototypes in this case.

The proposed MIL formulation on the other hand, learns by threshold learning a function that can also be applied for similar analysis purpose at any point x and can be defined by $f_{kNN}(x)$ as

$$f_{kNN}(x) = \sum_{i=1}^{|k^-|} \Psi(||x - k_i^-||) - \sum_{i=1}^{|k^+|} \Psi(||x - k_i^+||)$$

$$\text{such that, } |k^+| = |k^-| = k. \quad (2.15)$$

For the two bag case with $k = 1$, and for $\Psi(a) = a$ the equation becomes, $f_{kNN}(x) = ||x - \alpha|| - ||x - p||$. This equation can be solved at $x = p$ and at $x = \alpha$. At $x = p$, we get $f_{kNN}(p) = ||p - \alpha|| - ||p - p||$, $= \Phi - 0 = \Phi$, where Φ is the large distance measured between the positive instance p and the negative instance α. For the case of N positive and M negative bags with each positive bag containing only one positive instance each, the above measure will be, $f_{kNN}(p) = (2M - N)\Phi$ which is also the same case if there were M instances in the one negative bag. Notice how unlike DD, where for the case of M negative bags and N negative bags, the value of N didn't feature in $f_{DD}(p)$, the formulation is still dependent on N.

Similarly, at $x = \alpha$, $f_{kNN}(\alpha) = ||\alpha - \alpha|| - ||\alpha - p|| = -\Phi$, a large negative value. For the case of N positive and M negative bags with each positive bag containing only 1 positive instance each, the above measure will be, $f_{kNN}(p) = -N\Phi$. Therefore unless $M = 0$ (no negative bags at all) the proposed approach will still be maxima at positive instance prototype.

Introducing the third bag into the dataset, we get$f_{kNN}(x) = ||x - \alpha|| - ||x - p|| + ||x - \beta||$. At $x = p$, this yields,$f_{kNN}(p) = ||p - \alpha|| - ||p - p|| + ||p - \beta|| = \Phi - 0 + ||p - \beta||$. One can notice here, that as $\lim_{\beta \to p} f_{kNN}(p) = \Phi$ which is the same as the previous case without the negative bag. On the other hand,at $x = \alpha$, the function becomes, $f_{kNN}(\alpha) = 0 - \Phi + ||\alpha - \beta||$. As $\lim_{\beta \to p} f_{kNN}(\alpha) = 0$ which is still Φ, a large value, away from the prototype. Thereby the prototype is still not nullified unlike in the

case of DD.

2.6 Analogical Difference Between DD and the Proposed Formulation.

It is easy to wonder if the proposed formulation is the non-parametric analogy to DD and thus be decomposable from one to another. In this section we attempt to show the fundamental analogical differences between the two formulations and thereby elucidate the philosophical differences between the two.

Consider DD,

$$f_{DD}(x) = \prod_{i=1}^{n} Pr(x = t|B_i^+) \prod_{j=1}^{m} Pr(x = t|B_j^-) \tag{2.16}$$

where n is the number of positive bags in the dataset and m is the number of negative bags in the dataset. Taking log we get,

$$f_{lDD}(x) = \log \left[\prod_{i=1}^{n} Pr(x = t|B_i^+) \prod_{j=1}^{m} Pr(x = t|B_j^-) \right] \tag{2.17}$$

$$= \sum_{i=1}^{n} \log[Pr(x = t|B_i^+)] + \sum_{j=1}^{m} \log[Pr(x = t|B_j^-)] \tag{2.18}$$

by making an independence (iid) assumption for all the instances in each bag (as is done all through the MIL literature and first introduced in DD itself Maron and Lozano-Pérez (1998))

$$= \sum_{i=1}^{n} \log[\prod_{k=1}^{|B_i^+|} Pr(x = t|b_{i,k}^+)] + \sum_{j=1}^{m} \log[\prod_{l=1}^{|B_i^-|} Pr(x = t|b_{j,l}^-)] \tag{2.19}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{|B_i^+|} \log[Pr(x = t|b_{i,k}^+)] + \sum_{j=1}^{m} \sum_{l=1}^{|B_i^-|} \log[Pr(x = t|b_{j,l}^-)] \tag{2.20}$$

where an instance $b_{i,j}^{+/-}$ is the j^{th} instance from the i^{th} bag and the $+/-$ represents the bag being positive or negative and $|B_r^+|$ and $|B_r^-|$ represents the cardinality of the

r^{th} positive and negative bags respectively. Making the substitution $Pr(x = t|a) = \exp(\Psi(||x - a||))$ in the above equation for any instance represented here by a, we get,

$$f_{lDD}(x) = \sum_{i=1}^{n} \sum_{k=1}^{|B_i^+|} \Psi(||x - b_{i,k}^+||) + \sum_{j=1}^{m} \sum_{l=1}^{|B_i^-|} \Psi(||x - b_{j,l}^-||) \tag{2.21}$$

This equation sums up all the values of all the instances in all the bags, both positive and negative and weights them exponentially. In essence this equation is the energy distribution of all instances positive and negative with respect to x. This equation cannot decompose into equation 2.15, where negative instances are weighted additively and positive instances are weighted subtractively so as to find regions that are closer to positive instances and farther away from negative instances. The essence of the proposed method is to get away from the DD formulation using bags and to get into the instance space and perform a NN-like instance space tessellation and therein lies the analogical difference between the proposed method and DD.

2.7 Computational Complexity

From the above discussion on decomposing DD into sums, one can observe that computationally, to estimate functional value at each x (abstracting out the procedure of optimization), the DD form takes $O(no + mq)$ where o is the expected value of number of instances present in a positive bag and q is the expected value of number of instances present in a negative bag. For the k-NN method that complexity is only $O(2k)$, where $k << n$ and $k << m$.

Since estimating time taken for training and testing depends on coding methodologies, choice of optimization solvers, the authors couldn't provide timing information. Considering the fact that the proposed method always achieves better accuracy with

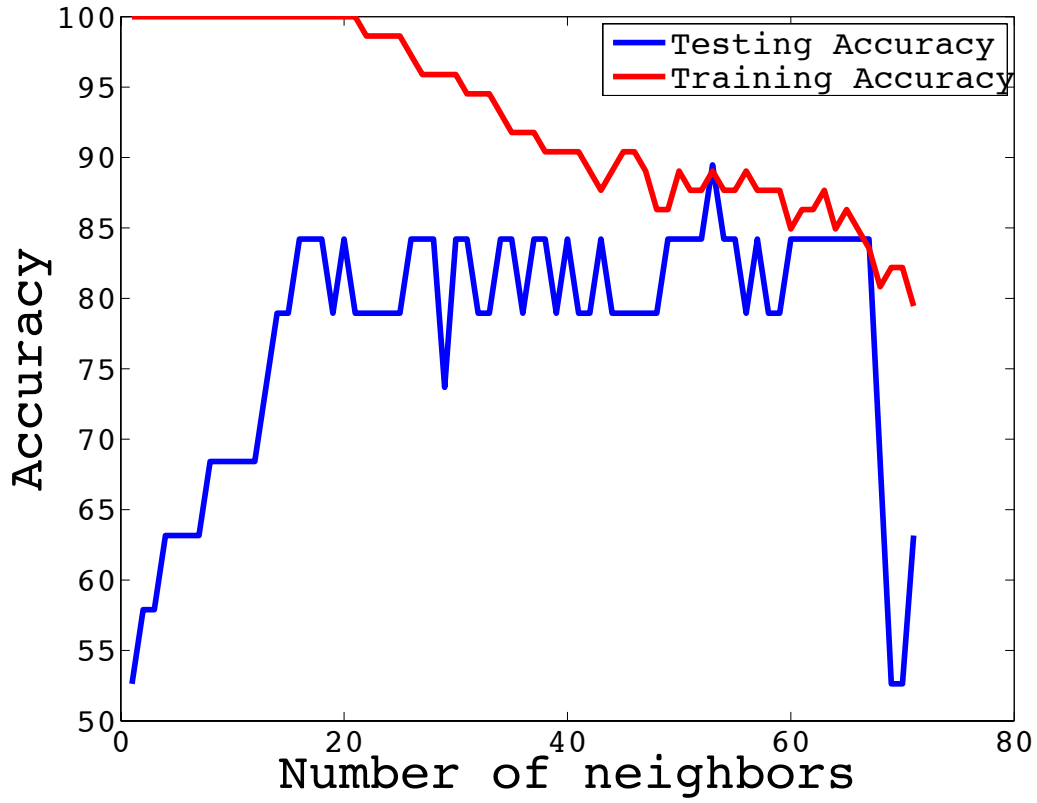


Figure 2.8: Accuracy vs k. It can be noted that accuracy stabilizes.

comparatively lesser number of prototypes than EMDD, just the fact that one has to deal with less number of prototypes also significantly increases the speed during both testing and training time. It is also noteworthy that once after training, we find the prototypes and the radii associated with them, we no longer need to calculate a points' distances using equation 4 all the time, we could simply use the threshold. In this case we only require the prototypes.

2.8 Sensitivity to k

The number of neighbors k doesn't have as strong an influence on generalization unless the number is too less or too high. The accuracy usually plateaus in a large range of k. All discussions that have been made on choosing a good k for traditional

kNN also apply to this formulation as well. We use the "elbow method" to fix a k manually, as mentioned in the paper. We performed an experiment on the MUSK2 dataset by varying k over a large range and plotted the accuracy vs k. The plot is shown in figure 2.8.

A rule of thumb for picking k is the intuition that you need as many members as half the noisy instances you want to allow around a positive prototype. In this intuition, one may think of choosing k analogous to choosing *slack* in a support vector machine.

2.9 Conclusion

In this section, it was postulate whether lazy learning ideas can be carried over from traditional non-parametric methods for supervised learning to a MIL setup. The proposal was a simple, yet novel usage of non-parametric learning philosophy to the MIL problem. In particular, the analysis focused on the MIL feature space using a k- NN philosophy and proposed a new formulation based on distances to k-nearest neighbours. The new formulation was compared and contrasted with the widely used DD formulation. The proposed approach was tested on the musk datasets, Andrews dataset and the corel datasets, and was found to be effective. The algorithm was used to solve the DR image classification problem and was found to be the best among other algorithms. It can therefore be concluded that a non-parametric learning philosophy to MIL not only makes intuitive sense but can also be quite a powerful tool for most general cases. The material from this chapter was published at the international conference on computer vision, 2015 Venkatesan *et al.* (2015).

Chapter 3

NEURAL DATASET GENERALITY

3.1 Introduction

Neural networks, particularly CNNs have broken all records recently in the computer vision research area. The growth of CNNs focused initially on the recognition of characters. Fukushima and LeCun were the initial pioneers. Independently they developed CNN based systems, some of which are still being used widely Fukushima and Wake (1991); LeCun *et al.* (1989). Large networks are often trained with large number of data samples to achieve good accuracies Szegedy *et al.* (2014); Krizhevsky *et al.* (2012). Still, scepticism over CNNs among the modern day computer vision scientists stems from the fact that one does not have a clear understanding of its inner working. Some studies show that a few ($< 1\%$) nodes are all that are actively contributing to classification Escorcia *et al.* (2015). They also suggest that large networks often overfit, but since the data is too large over-fitting often works as an advantage Nguyen *et al.* (2015). While it is reasonable to expect edge detectors and Gabor-like features in the lower-level filters and more sophisticated concepts at the higher levels, it is not clear as to why these filters adapt themselves in this manner. What is fairly clear though is that different datasets result in different sets of filters that are similar if the datasets are similar. It is only natural to ask, what role does the data itself play in such filters being learnt and how they compare with filters learnt from another dataset. Let us take the view that the filters learnt by networks when trained using a particular dataset represent the detectors for some *atomic structure* in the data itself. In which case each layer is a mapping form the previous layer to

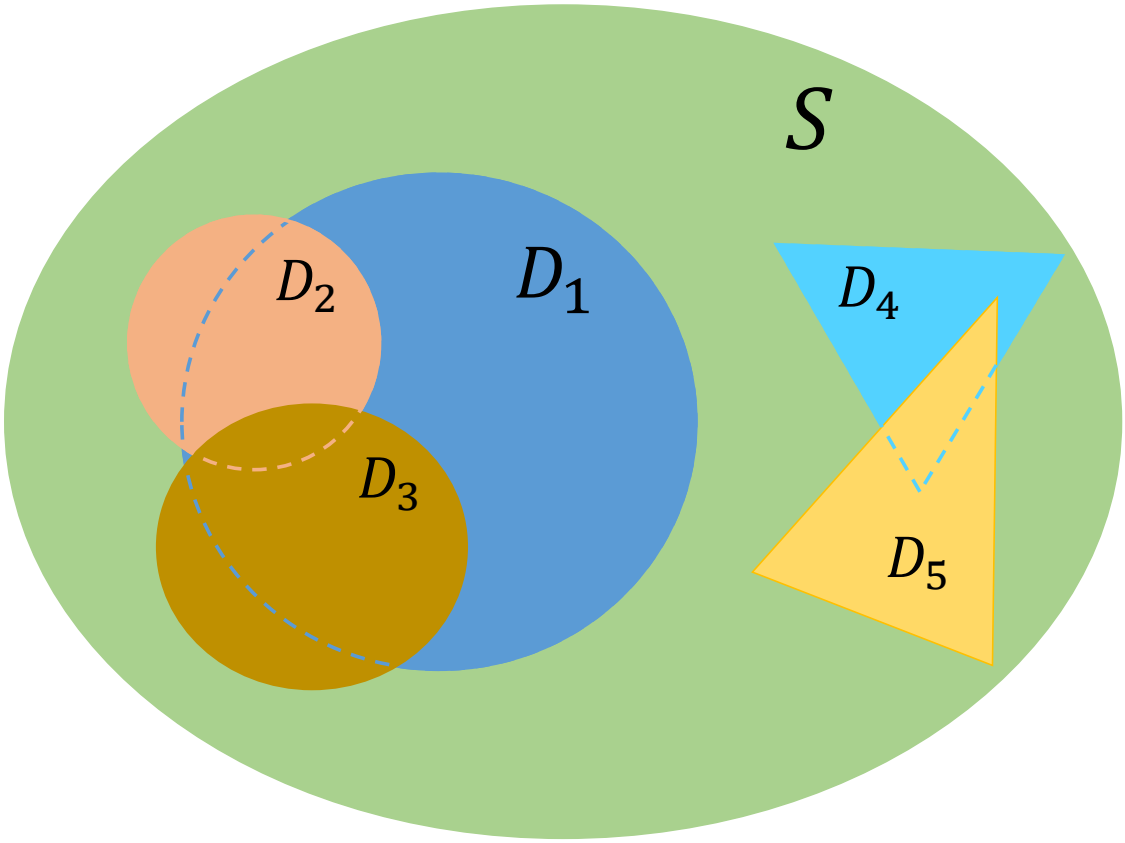


Figure 3.1: Thought experiment to describe the dataset generality. S is the space of all possible atomic structures, $D_1 - D_5$ are the atomic structures present in respective datasets. ©2016 IEEE.

the next layer that is constructed using combinations of these atomic structures in the first layer in order to minimize a cost.

Let us first define *atomic structures* to be the forms that CNN filters take by virtue of the entropy of the dataset it is learning on, analogous to dictionary atoms. Complex datasets have more and varied atomic structures. Consider the following thought experiment: Let's assume that all possible atomic structures reside in an universe S. Suppose we have a set of three datasets $D = \{D_1, D_2, D_3\}$ and $D \in S$. Consider the system in figure 3.1. The figure describes the configuration of the elements of D. One would now recognize that D_1 is a more general dataset with respect to D_2

and D_3. It is so because, while D_1 *contains* most of the atomic structures of D_2 and D_3, the latter do not contain as many atomic structures of D_1. While this analysis is simplified for one layer, in typical CNNs, co-adaptation plays a major role in the learning of these atomic structures. Therefore, generality as defined by the overlap of areas in a layer-wise Venn diagram is impractical to obtain.

We postulate that, the generalization performances of CNNs on one dataset re-trained on a network initialized by training using another, could be used to derive generality. We call this process of pre-training as *prejudicing*. By prejudicing on the first dataset, we froze and unfroze layers and retrained the networks on the second dataset. By freezing layers we are making a network more obstinate and we call this process obstination [1] . The more the layers are frozen, the more obstinate the feature extractor is, therefore the harder the classifier has to work. If the prejudice was general enough, the classifier shall still generalize fairly well enough. What this means is that if the prejudicing dataset is more general than the re-train dataset, the classifier can generalize better than vice versa. Let us develop, a generality metric by comparing the gain in performances of networks of various obstination. Using a generality such as the one proposed, it becomes clearer as to what kind of datasets are to be used to prejudice CNNs with during transfer learning. We even discovered that samples with particular labels within a dataset alone are general enough. So, if we begin by prejudicing the network on only those and then moved on to the rest of the labels, we were able to learn the rest of the dataset with considerably less training samples while achieving comparable generalization performances.

Off-the-shelf networks such as VGG, overfeat and various published Caffe model weights are trained on large scale image datasets such as Imagenet or PASCAL Si-

[1]Obstinate layer or freezing implies that the weights were not changed during backprop. The layer remains prejudiced.

monyan and Zisserman (2014b); Jia *et al.* (2014); Girshick *et al.* (2014); Russakovsky *et al.* (2015); Everingham *et al.* (2010). For instance, while these may work on applications such as human pose recognition or vehicle detection, they do not necessarily work on tasks involving medical images. This is because the datasets on which they are trained are not general enough to adapt to the representational requirements of medical images, which is on a manifold unique and disjoint form the manifolds of natural images. This is visualized in D_4 and D_5 from figure 3.1. Even a large collection of natural images is not general enough to have networks trained that are suitable to medical images. In these cases, the prejudiced network often fails. For instance, on the Colonoscopy dataset discussed later a 22 layer deep overfeat features, trained with a logistic regression performs poorer than a 3 layer deep CNN trained from random initialization, which is in turn outperformed when initialized by a network trained on an endoscopy dataset.

In this section we considered popular offline character recognition datasets and arrived at some interesting analysis and generalities. We also show that within the MNIST dataset, classes $[4, 5, 8]$ are general enough that we could learn the other classes with very few (even just one) samples, when prejudiced with networks trained on $[4, 5, 8]$. We also considered more sophisticated datasets such as Cifar 10 and Caltech 101 against some medical image datasets for colonoscopy video quality Krizhevsky and Hinton (2009). This study led us to two major research insights:

1. If one has very few data to learn from, which other dataset is better to prejudice the network with? The answer is particularly helpful when dealing with medical image datasets where data is very scarce and one can't simply use a network trained on VOC datasets as feature extractors as discussed above.

2. Among the various classes during the training procedure, if we prejudice with a

certain *general* set of classes first and then move on to others later, generalization to all classes, even for those with few samples is better. This is particularly significant if the dataset has a lot of samples in certain classes and not as much of others.

The rest of the paper is organized as follows: section 3.2 discusses related works, section 3.3 presents the design of our experiments, section 3.4 shows some results on the core-experiment and section 3.5 provides concluding remarks.

3.2 Related Work

One related work that this article shares with is the work by Yosinski et al Yosinski *et al.* (2014). In that article, the authors considered two tasks A and B that were essentially 500 classes each from the Imagenet dataset Russakovsky *et al.* (2015). They trained an 8 layer network on one of the tasks (say A). They then initialized a new network carrying over the first n layers from the previous job while randomly initializing the others. This new network was used to retrain task B. Such a network was AnB^+. They experimented by obstination of the carried over layers. Such a network was AnB. They also studied the specificity of each layer and their contributions to the overall performance. They also showed that networks working on similar tasks had a high memorability and that co-adaptation of layers increased the generalization performance.

While this analysis is interesting, it was performed on only one dataset: Imagenet. By design, the networks were forced to learn very general filters, so as to be best transferable. Since the images were all *natural images*, one would expect the layers to be more Gabor-like at earlier layers and have more label specific features at later layers, which was what was observed. Also, the paper analysed the transferability of the feature extractors from the perspective of the networks in terms of their fall in gen-

eralization performance. This analysis was not catered to the dataset's perspective, which is that the filters learned are a property of the dataset being trained on. This was not a problem for the authors as their datasets for tasks A and B occupied similar manifolds. This analysis also didn't explore re-training using the same network but rather went with re-initializing so that they could learn new co-adaptations. This is not interesting to the study of generality as we want to observe the effect of filters transferred from one dataset on another. The more *general* a dataset, the more variety of atomic structures it offers to the network to learn. We used this idea to define a generality metric between two datasets. To do so, we cannot follow the techniques used by Yosinski et al.

Another closely related work is the work on *dark knowledge* by Hinton et al, Hinton *et al.* (2015). Here the authors suggest that among the various classes in a dataset, there exists some amount of generalization knowledge that could be transferred. The authors construct a large network that learns all its classes. They then go on to train a smaller network with the same dataset (or with a dataset that is missing some of the classes altogether). While training this smaller network though, instead of using the the hard labels, they also use the softmax output from the large network also for backprop. This creates an effect of the larger network *guiding* the smaller network to not just generalize to the dataset, but also to generalize to unseen classes. This is because, as the argument goes, "the network learns the relationship between the classes" and "all the knowledge is among the relative probabilities or softmaxes that the network is almost certain is wrong" Hinton *et al.* (2015).

Although the author retrains an entire network that is randomly initialized using the softmax outputs from a trained network and uses this as prejudice, no information is actually being transferred in terms of actual filters. Ergo, this work, while interesting, also doesn't help in understanding generality of the data itself in a more

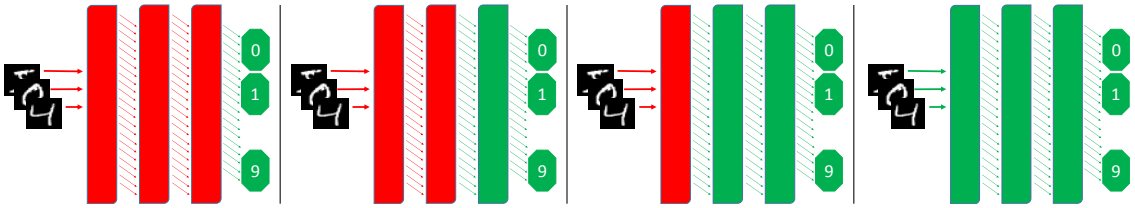


Figure 3.2: Protocol of obstination: From left to right, all layers frozen, one, two and three layers unfrozen. Green represent unfrozen and red represent frozen. Note that the layers are always unfrozen from the end and that the softmax layer is always unfrozen and randomly initialized. This should be generalized similarly for more than three layers also. ©2015 IEEE.

direct manner. Some of the claims made by this article though were indirectly and independently verified by us through our generality results. The basic claim of their work is that among only a handful of classes, there is enough knowledge to generalize to other classes. Unless there exists some generality between classes, training on particular classes will not have been representational enough for the other classes to learn on. We directly verify this by showing that some classes alone have a high generalization to the rest of the dataset and make a similar conclusion from an entirely independent direction of research.

3.3 Design of Experiments

Consider figure 3.3. Among the various datasets shown, it is natural to expect any network trained on MNIST to contain simpler filters than MNIST-rotated. This is because, while MNIST-rotated contains many structures from MNIST, due to the rotations, MNIST-rotated will contain additional structures that require the learning of more *complicated* filters. A network trained on MNIST-rotated on its first layers will be expected to additionally have filters for detecting sophisticated oriented edges than for MNIST. This would mean that prejudicing a network with MNIST to then retrain MNIST-rotated is much less helpful than vice versa. A network prejudiced with a general enough dataset is better to be retrained for it generalizes easily. A prejudice

must come from a more general dataset if a prejudice transfers positive knowledge as shown in their generalization performances. We use this simple intuition to argue that MNIST-rotated is a more general dataset with respect to MNIST.

Our basic experiment is conducted between pairs of datasets D_i and D_j. Firstly, we train (prejudice) a randomly initialized network with dataset D_i. We call this network $n(D_i|r)$ or the base network (r implies random initialization). We then proceed to retrain $n(D_i|r)$ as per any of the setup shown in figure 3.2. $n_k(D_j|D_i)$ would imply that there are k degrees of freedom, or to be precise, k layers of filters that are allowed to learn by dataset Dj that is prejudiced by the filters of $n(D_i|r)$. $n_k(D_j|D_i)$ has $N - k$ obstinate layers that carries the prejudice of dataset D_i, where N is the total number of layers. Note that more degrees of freedom implies that the network is less obstinate to learn. Also note that these layers can be both convolutional or fully connected neural layers. Any idea expressed here can be extended to any type of parametrized layers. In fact while we perform operations such as batch normalizations, we even freeze and unfreeze the α and β of batch norm Ioffe and Szegedy (2015). Obstination also includes the bias parameters. Layers learn in two facets. They learn some components that are purely their own and some that are co-adapted from previous layers that are allowed to learn as well. By freezing some layers we are making those layers a fixed functional transformation. Note that the performance gain from $n_k(D_j|D_i)$ and $n_{k+1}(D_j|D_i)$ is not because of just the new layer $k+1$ being allowed to learn, but of the combination of all $k+1$ layers allowed to learn. Figure 3.2 shows the setup of our experiments and explains degrees of freedom. These are our obstination protocols. Notice that in all the various setup, the softmax layer remains non-obstinate. In fact the softmax layer is always randomly re-initialized because not all dataset pairs have the same number of labels. Also notice that the unfreezing of layers happen from the rear. We cannot unfreeze a layer that feeds into a frozen layer.

This is because, while the unfrozen layer learns a new filter and therefore represents the image on new distributed domains, the latter layer is not adapting to such a transformation. When there are two layers unfrozen, the two layers should be able to co-adapt together and must finally feed into an unfrozen classifier layer.

3.3.1 Dataset Generality

Suppose the generalization performance of $n(D_j|r)$ is $\Psi(D_j|r)$ and the generalization performance of $n_k(D_j|D_i)$ is $\Psi_k(D_j|D_i)$. First order dataset generality or simply dataset generality of D_i with respect to D_j at the layer k is given by,

$$g_k(D_i, D_j) = \frac{\Psi_k(D_j|D_i)}{\Psi(D_j|r)} \tag{3.1}$$

This indicates the level of performance that is achieved by D_j using $N - k$ layers worth of prejudice from D_i and k layers worth of features from D_i combined with k layers of novel knowledge from D_j together. Note that the generality is calculated for the base dataset as a measure of how the re-train performs with the prejudice of the base dataset. $g_k(D_i, D_j) > g_k(D_i, D_l)$ indicates that at k layers, D_i provides more general features to D_j than to D_l. Conversely, when initialized by $n(D_i|r)$, D_j has an advantage in learning than D_l.

Note that, $g_k(D_i, D_i) \geq 1 \ \ \forall k.\ g_k(D_i, D_j)$ for $i \neq j$ might or might not be greater than 1. If $g_k(D_i, D_j) \geq 1$ for $i \neq j$, it indicates that D_j is at least very similar to D_i (such as the case considered by Yosinski et al.) and at most a perfect generalizer of D_i Yosinski *et al.* (2014).

3.3.2 Class Generality

D_i and D_j need not be entire datasets but can also be just disjoint class instances of the same dataset that is split in two. These generalities will tell us if particular

classes are themselves more general than others. For instance, we divided the MNIST dataset into two parts. The first part contained the classes $[4, 5, 8]$, the rest were contained by the second part [2] . We performed the generality experiments with MNIST$[4, 5, 8]$ as base, which was trained over a random initialization. We re-trained this prejudiced network using the second part with the same experiment design as above. We defined class generality as the generality, of a class or a set of classes, retrained on the prejudice of the other mutually exclusive classes.

We repeated this experiment several times with decreasing number of training samples per-class in the retrain dataset of MNIST $[0, 1, 2, 3, 6, 7, 9]$. All the while, the testing set remained the same size. This implies that the prejudiced network retrains on a much smaller dataset and tests on a much larger dataset. The re-train dataset had 7 classes. We created seven such datasets with $7p,\ p \in [1, 3, 5, 10, 20, 30, 50]$ samples each. We now define sub-class generality as the generality of these sub-sampled datasets (in each class we only consider a small random sample), retrained on the base of other mutually exclusive classes (MNIST$[4, 5, 8]$). . Initializing a network that was trained on only a small sub-set of well-chosen classes can significantly improve generalization performance on all classes, even if trained with arbitrarily few samples, even at the extreme case of one-shot learning.

3.3.3 Datasets Used

We designed these experiments across three board categories of datasets: 1. Character datasets that included MNIST LeCun *et al.* (1998a), MNIST-rotated Larochelle *et al.* (2007), MNIST-random-background Larochelle *et al.* (2007), MNIST-rotated-background Larochelle *et al.* (2007), Google street view house numbers Netzer *et al.*

[2] We chose this combination of classes strategically after trail and error as these are the most general among the classes and exaggerate the effect.

(2011), Char 74k English de Campos *et al.* (2009) and Char 74k Kannada de Campos *et al.* (2009) 2. Natural image datasets that includes Cifar 10 and Caltech 101 Krizhevsky and Hinton (2009); Fei-Fei *et al.* (2007) and 3. Natural images against medical images that included in addition to Caltech 101 a Colonoscopy video qualitty dataset. We leave it to the reader to find for themselves details about the datasets from the original articles. Although we chose only a handful of datasets, the intention of this article was only to show that such generality measures could be made. The scope of this article was not to benchmark various publicly available popular datasets. Neither was it to make suggestions specific to types of datasets.

3.3.4 Network Architecture and Learning

We used one standard network architecture for all character datasets and experiments, one for Cifar 10 vs. Caltech 101 and another standard for Caltech 101 vs. Colonoscopy.

The network architectures, learning rates and other details are provided below. The experiments were conducted on a Macbook Pro Laptop using an Nvidia GT 750M GPU, for character datasets and on an Nvidia Tesla K40 GPU for the others, with cuDNN v3 and Nvidia CUDA v7.

No pre-processing were done on the images themselves except for cropping, resizing, normalizing. The images were all normalized to lie in $[0, 1]$. The character recognition datasets were all of a constant $28X28$ grayscale, the Caltech 101 vs. Cifar 10 experiments were performed ar $32X32$, RGB and the Caltech 101 vs. Colonsoscopy were at $128X128$, RGB. It is to be noted that the aim of the authors was not to set up the networks to achieve state-of-the-art. The authors did although try to achieve satisfactory performances on all base datasets involved before proceeding with the experimentation.

Character Datasets

Our networks had three convolutional layers with $20, 20$ and 50 kernels respectively. All the filters were 5 X 5 and were all stride 1 convolutions. The first layer didn't have any pooling. The second and the third layer maxpool by 2 subsampled. All the layers used rectified linear units ($ReLU$) activations Nair and Hinton (2010). The classifier layer was a softmax layer and we didn't use any fully connected layers. We used a dropout of 0.5 only from the last convolutional layer to the softmax layer Srivastava *et al.* (2014). We optimized a categorical cross-entropy loss using an rmsprop gradient descent algorithm Dauphin *et al.* (2015). For acceleration we used Polyak Momentum that linearly increases in range $[0.5, 1]$ from start to 100 epochs Polyak (1964). Unless early terminated, we ran 200 epochs. We also used a constant L_1 and L_2 regularizer co-efficients of 0.0001. Our learning rate was a 0.01 with a multiplicative decay of 0.0998.

CIFAR10 vs. Caltech101 and Caltech 101 vs Colonoscopy

For this task, the networks had five convolutional layers with $20, 20$, 50, 50 and 50 kernels respectively. We also had a last fully connected layer of 1800 nodes, which also had a dropout of 0.5. All the filters were 5 X 5 and were all stride 1 convolutions. Only the last layer maxpool by 2 subsampled. All the layers used rectified linear units ($ReLU$) activations Nair and Hinton (2010). All CNN and MLP layers were also batch normalized Ioffe and Szegedy (2015).The classifier layer was a softmax layer and we didn't use any fully connected layers. We used a dropout of 0.5 only from the last convolutional layer to the softmax layer Srivastava *et al.* (2014). We optimized a categorical cross-entropy loss using an rmsprop gradient descent algorithm Dauphin *et al.* (2015). For acceleration we used Polyak Momentum that linearly increases in

range $[0.5, 0.85]$ from start to 100 epochs Polyak (1964). We use a learning rate of 0.001 for the first 150 epochs and then fine tune with a learning rate of 0.0001 for an additional 50 epochs unless early-terminated. Our learning decay of was subtractive 0.0005. Figure 3.4 shows more generality curves.

3.4 Results and observations

3.4.1 Character Datasets

Figure 3.4 shows the generalities of MNIST-rotated-bg and Kannada prejudiced by all other the character datasets. For reference each plot also shows the generalization performance of a randomly initialized base convolutional network. The following are some observations of interest:

While no dataset is qualitatively the most general, it is quite clear that *MNIST dataset is the most specific*. Rather, MNIST dataset is one that is generalized by all datasets very highly at all layers. Surprisingly, MNIST dataset actually gives better accuracy when prejudiced with other datasets, rather than when initialized with random, if all layers were allowed to learn. This is a strong indicator that *all datasets contain all atomic structures of MNIST*.

NIST, Char74-English and Char74-Kannada follow similar generalization trends with almost all the datasets. With no degrees of freedom they all generalize rather poorly, but their generalities shoot up once one or many layers of the base networks are unfrozen. This indicates two properties: Firstly, these three datasets have similar manifolds. Secondly this also indicates that the last layers of the base datasets are extracting some particular quality of atomic structures that are present in the these datasets alone. Similarly, SVHN does not generalize in the first layer to most datasets, it generalizes much better in the latter layers. This is particularly noticeable in

MNIST and Kannada. This further exemplifies the results.

While initially one would have assumed that Kannada would be a general dataset, we observed the contrary. SVHN, Char74-English and Nist generalizes better to Kannada than even Kannada itself does. *English characters seem to be a more general set than Kananda.* While counter-intuitive, this result is immediately obvious when one pays close attention to the filers that are learnt and the dataset itself. Kannada is dominated by predominantly curved edges only, whereas even MNIST has a multitude of unique atomic structures.

Figure 3.6 demonstrates some interesting phenomenon that we discovered often. *The gain in performance achieved, constantly decreases with increase in degrees of freedom.* Through the epochs, unfreezing only the classifier layer, quickly converges. But while unfreezing, all layers converge at about the same number of epochs. We also observe, that MNIST retrained over MNIST-rotated-background, with *the last degree of freedom does not learn antything at all.* The error rate is within the statistical margin of error. This is a testament to the generality of MNIST-rotated-background among the MNIST datasets. One might expect this because MNIST-rotated-background contains smooth background images (similar to natural image set) and MNIST characters that are rotated. These conditions provide for a good generality.

For the intra-class experiment described in the section 3.3.2 above, table 3.1 shows the accuracies. From the table one can observe that even with one-sample per class, a 7-way classifier could achieve 22% more accuracy than a randomly initialized network. It is note worthy that the last row of table 3.1 still has 100 times less data than the full dataset and it already achieves close to state-of-the-art accuracy even when no layer is allowed to change. This is a remarkably strong indicator that the classes $[4, 5, 8]$ generalizes the entire dataset.

p	base	$k=0$	$k=1$	$k=2$	$k=3$
1	Random	-	-	-	55.61
	MNIST[458]	73.07	73.91	76.37	77.52
3	Random	-	-	-	73.34
	MNIST[458]	83.61	87.2	85.7	87.6
5	Random	-	-	-	83.32
	MNIST[458]	90.98	92.98	92.6	92.07
10	Random	-	-	-	81.31
	MNIST[458]	91.55	93.71	93.82	95.08
20	Random	-	-	-	87.77
	MNIST[458]	95.52	95.52	97.07	96.78
30	Random	-	-	-	88.62
	MNIST[458]	96.5	97.34	97.35	97.45
50	Random	-	-	-	90.78
	MNIST[458]	96.38	97.40	97.71	97.38

Table 3.1: Sub-sample experiment and its generalization accuracies for different layers of freezing. The re-train network was MNIST$[0, 1, 2, 3, 6, 7, 9]$. For obvious reasons random initializations are trained only with all layers unfrozen, hence the missing values.

Figure 3.7 mimics the same. We also observed that once initialized with a general enough subset of classes from within the same dataset, the generalities didn't vary among the layers like it did when we initialized with data from outside the mother dataset. We also observed that the more the data we used, more stable the generalities remained. Point of take away from this experiment is that if the classes are general enough, one may now initialize the network with only those classes and then learn the rest of the dataset even with very small number of samples.

3.4.2 CIFAR 10 vs. Caltech 101

From figure 3.4 we observe that Caltech 101 doesn't generalize to Cifar 10, which is surprising because Caltech 101 has a lot more classes. One would expect it to be more general. Its quite the opposite because Caltech 101 although has a lot of classes, the variability of each class is not as much as the variability in the Cifar 10 dataset. But it is altogether a serendipitous result that *Cifar 10 is more general than Caltech 101 on the lower layers*. However after three layers of obstination, we find that when the generalities crosses 1, the effect nullifies and reverses slightly. Even though the low-level features are more general in Cifar 10, Caltech 101 generalizes more on higher layers.

3.4.3 Caltech 101 vs. Colonoscopy

The colonoscopy dataset's labels identify if a image is deemed to be of a quality that is good enough so as to make a diagnosis on the pathology of that particular image. Figure 3.5 show the filters learnt by Caltech 101 base network and Colonoscopy base network for the exact same architecture from random initialization. Two things are immediately apparent from the learnt filters that while Caltech 101 learns more structured and organized shape features, Colonoscopy dataset learns at first sight what appears to be unstructured blob detectors and detectors for dark colors. These features still produce state-of-the-art accuracy on the dataset. On observation of the activations produced after the first layer, and from observations of images and their labels, one can immediately recognize that what the network is learning is indeed changes in brightness patterns.

Most often the video quality in colonoscopy is affected because of saturation when too much light is thrown at a scene. The quality is also affected due to light reflection

from bodily fluids that is also noticeable in the activations. As also can be noticed that most of the filter colors are yellowish or blueish. On an colonoscopy video most often the video is also labelled poor quality when these colors are present, as these colors are often present mostly because of scattering and reflections. Having made these observations one would arrive at the obvious conclusion that neither dataset generalizes the other. This was indeed the result observed from figure 3.4. Although, Caltech 101 seem to generalize a bit better for even though it predominantly learns shapes, it learns some color features also.

3.4.4 Summary of Results

From all these results and observations, we could summarize that one should prefer to initialize with a general dataset that might have a lot of variability or rather generality in data, when attempting to train with very few number of samples. Whenever possible one must initialize the network trained by a general dataset as this always boosts generalization performance. When there are biased datasets with large number of samples in some classes and fewer in others, one should train the most general classes first. Once the network is well-prejudiced one should start introducing the classes with fewer number of and less general samples, provided the general class is general enough.

3.5 Conclusions

In this chapter, we used the performance of CNNs on a dataset when initialized with the filters from other datasets as a tool to measure generality. We proposed a generality metric using these generalization performances. We used the proposed metric to compare popular character recognition datasets and found some interesting patterns and generality assumptions that add to the knowledge-base of these datasets.

In particular, we noticed that MNIST data is one of the most specific dataset. We also found that Char74k Kannada is less general than English datasets. We also calculated generality on class-level within a dataset and conclude that a few well-chosen classes used as pre-training could build a network that is well-initialized that even with 100 times less samples, we could learn the other classes. We also provided some practical guidelines for a CNN engineer to adopt. After performing similar experiments on popular imaging datasets and medical datasets, we made similar serendipitous observations. The material from this chapter was published at the International Conference on Image Processing, 2016 Venkatesan *et al.* (2016a) and a longer version of it was published in arXiv Venkatesan *et al.* (2016b).

Figure 3.3: Samples of some of the datasets that we used in this analysis. From top to bottom: MNIST LeCun *et al.* (1998a), MNIST-rotated Larochelle *et al.* (2007), MNIST-random-background Larochelle *et al.* (2007), MNIST-rotated-background Larochelle *et al.* (2007), Google street view house numbers Netzer *et al.* (2011), Char 74k English de Campos *et al.* (2009), Char 74k Kannada de Campos *et al.* (2009). Last two rows, first five from left are CIFAR 10 and the rest are Caltech101 Krizhevsky and Hinton (2009); Fei-Fei *et al.* (2007). The bottom row is the colonoscopy dataset.

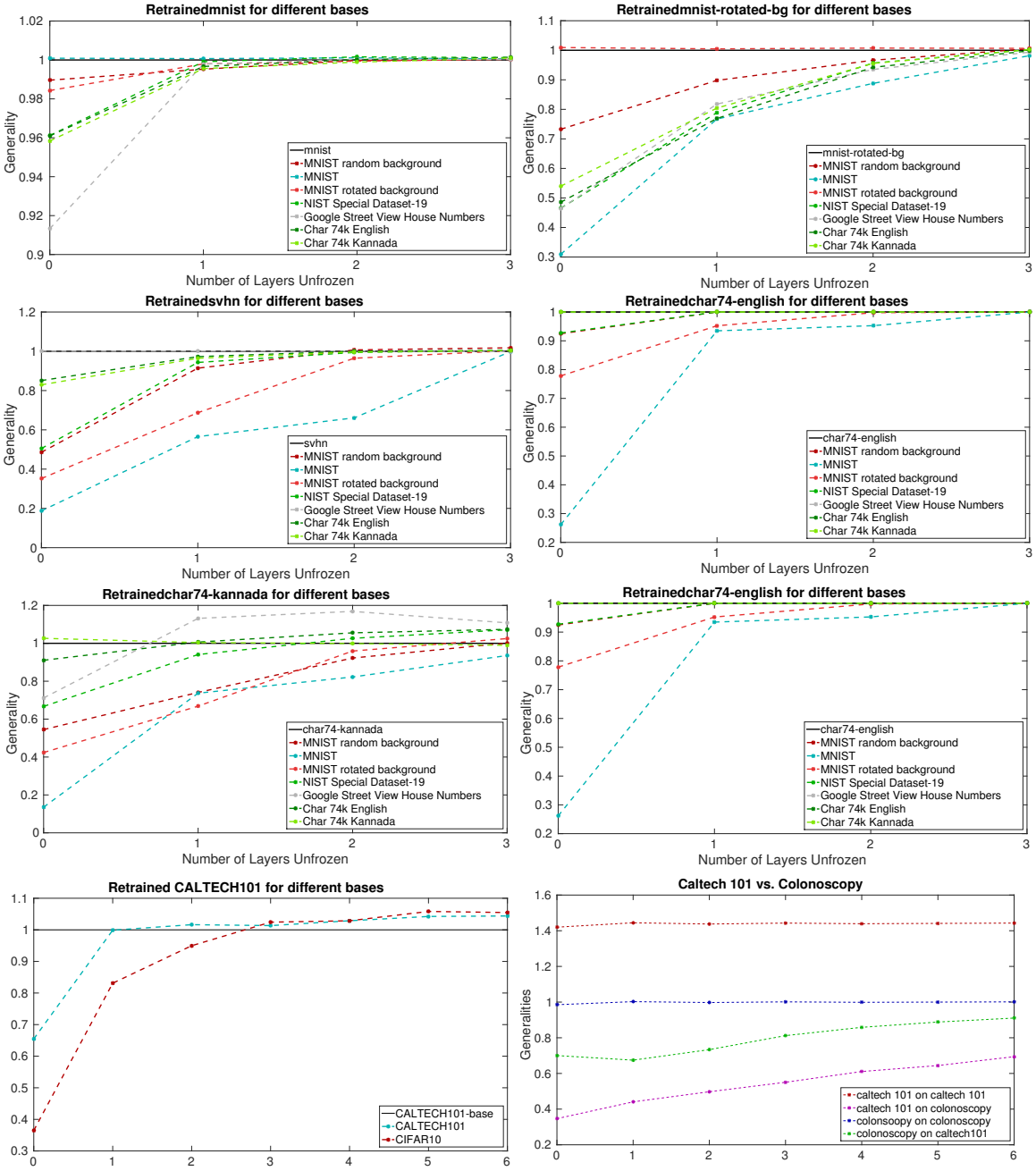


Figure 3.4: Generalities of datasets not shown in the actual paper. The dark line represents the accuracy of $n(D|r)$. Please zoom on a computer monitor for closer inspection. ©2016 IEEE.

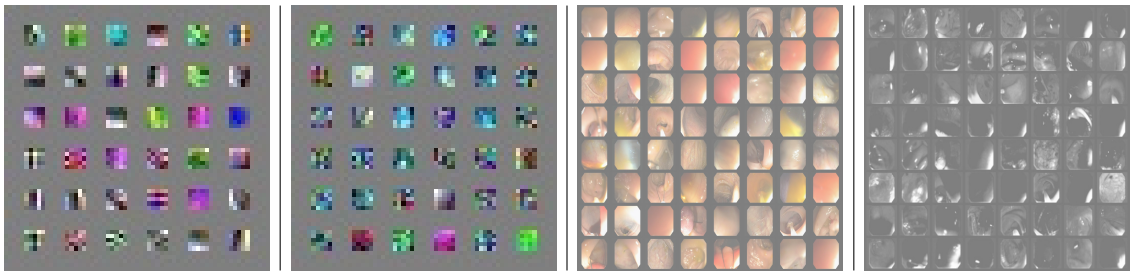


Figure 3.5: From left to right, separated by a line are filters learnt by a base Caltech101 base colonoscopy, sample images from the colonoscopy dataset and their first activation for a filter that detects smooth areas of brightness.

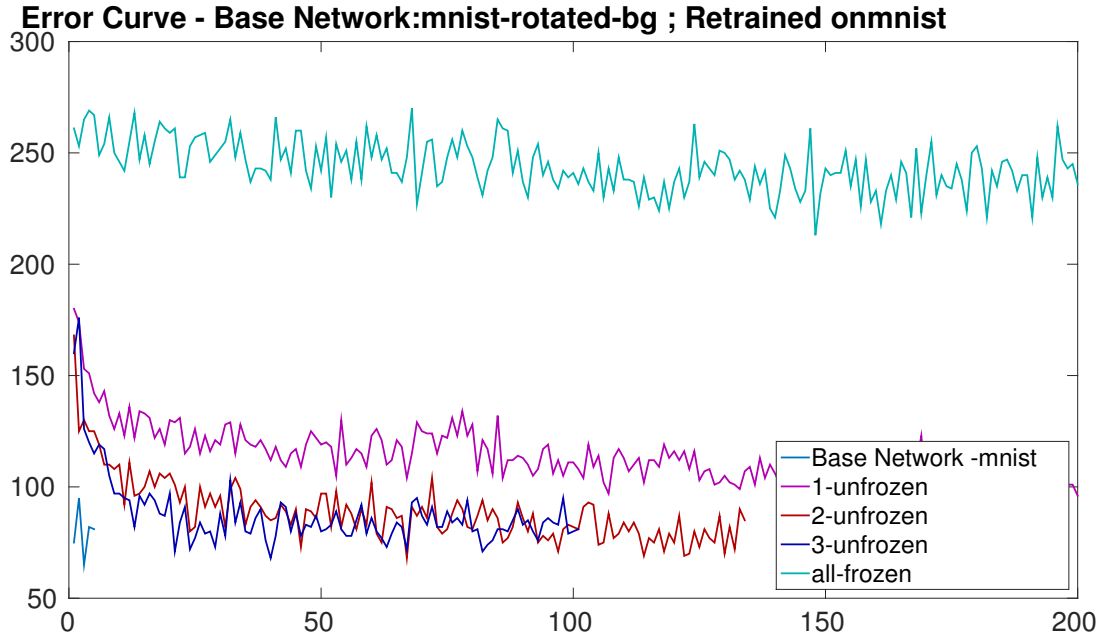


Figure 3.6: Validation Errors vs Epoch Number for Base-MNIST-rotated-bg Retrained on MNIST

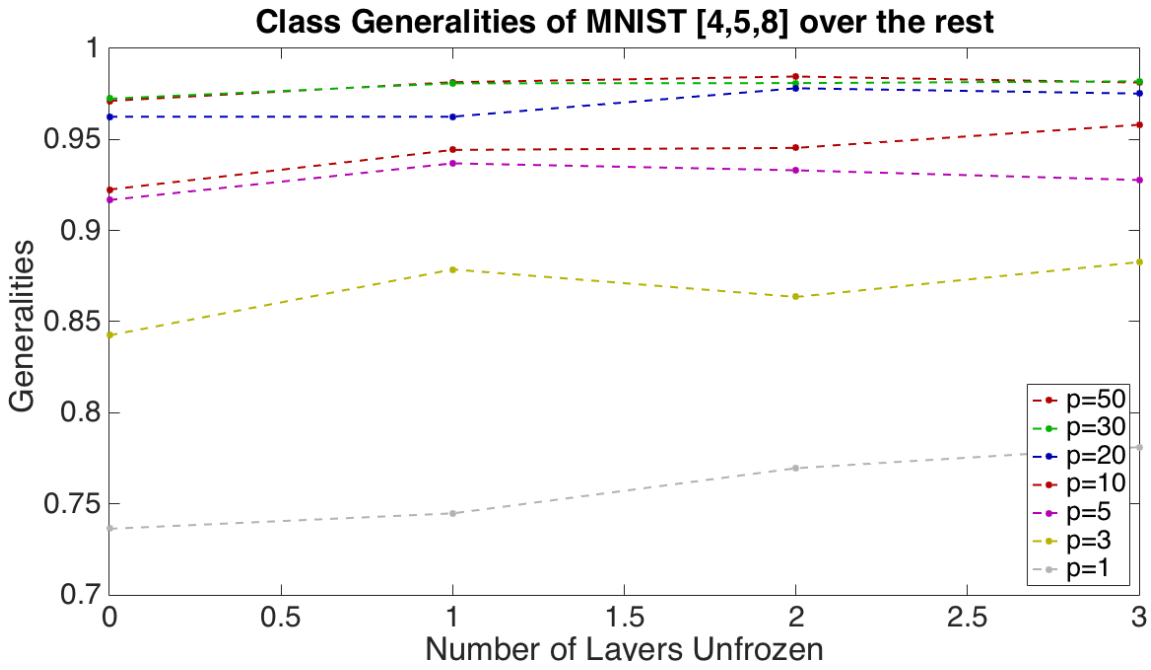


Figure 3.7: Sub-class Generalities for MNIST $[4, 5, 8]$

Chapter 4

MENTEE NETS

4.1 Introduction

With the proliferation of off-the-shelf, downloadable networks such as VGG-19, overfeat, R-CNN and several others in the caffe model zoo, it has become common practice in the computer vision community to simply fine-tune one of these networks for any task Simonyan and Zisserman (2014b); Jia *et al.* (2014); Girshick *et al.* (2014). These networks are usually trained on a large dataset such as Imagenet and Pascal Russakovsky *et al.* (2015); Everingham *et al.* (2010). The proponents of these networks argue that these networks have learnt image representations that are pertinent for most datasets that deal with natural images. Under the assumption that all these datasets are *natural images* and are derived from a similar distribution this might as well be true. Even with such networks, features that are unique to each datasets do matter. While fine-tuning of an already trained network works to a certain extent, these features are not *learnt* in a traditional manner on the target dataset but are simply copied. There is also no guarantee that these features are the best representations for the target dataset, although there is some validity in expecting that such a representation might work well, since after all it was learnt from a large enough dataset.

Most computer vision scientists do not attempt to train a new architecture from scratch (random initializations). Training even a mid-sized deep network with a small dataset is a notoriously difficult task. Training a deep network, even those with mid-level depth require a lot of supervision in order to avoid weight explosion. On

most imaging datasets, with image sizes being $224X224$, the memory insufficiency of a typical GPU restricts the mini-batches to less than 100. Using small mini-batches and small datasets lead to very noisy and untrustworthy gradients. This leads to weight explosions unless the learning rates are made sufficiently smaller. With smaller learning rates, learning is slowed down. With smaller mini-batches learning is unstable. One way to avoid such problems is by using regularization. By regularizing we can penalize the gradients for trying to make the weights go higher and higher. Batch Normalization is another technique that is quite commonly used to keep weight explosion under check Ioffe and Szegedy (2015). Even with these regularization techniques, the difficulty of training a deep network from scratch leads most computer vision scientists to use pre-trained networks.

There are also several reasons why one might favour a smaller or a mid-sized network even though there might be a better solution available using these large pre-trained networks. Large pre-trained networks are computationally intensive and often have a depth in excess of 20 layers. The computational requirement of these networks do not make them easily portable. Most of these networks require state-of-the-art GPUs to work even in simple feed forward modes. The impracticality of using pre-trained networks on smaller computational form factors necessitates the need to learn smaller network architectures. The quandary now is that smaller networks architectures cannot produce powerful enough representations.

Many methods have been recently proposed to draw additional supervision from large well-trained networks to regularize a new network while learning from scratch Wang *et al.* (2015); Romero *et al.* (2014); Balan *et al.* (2015); Chan *et al.* (2015). All of these works were inspired from the Dark Knowledge (DK) approach Hinton *et al.* (2014). All these techniques use at most one layer of supervision on top of the soft-max supervision and try to use this technique to learn more deeper networks better.

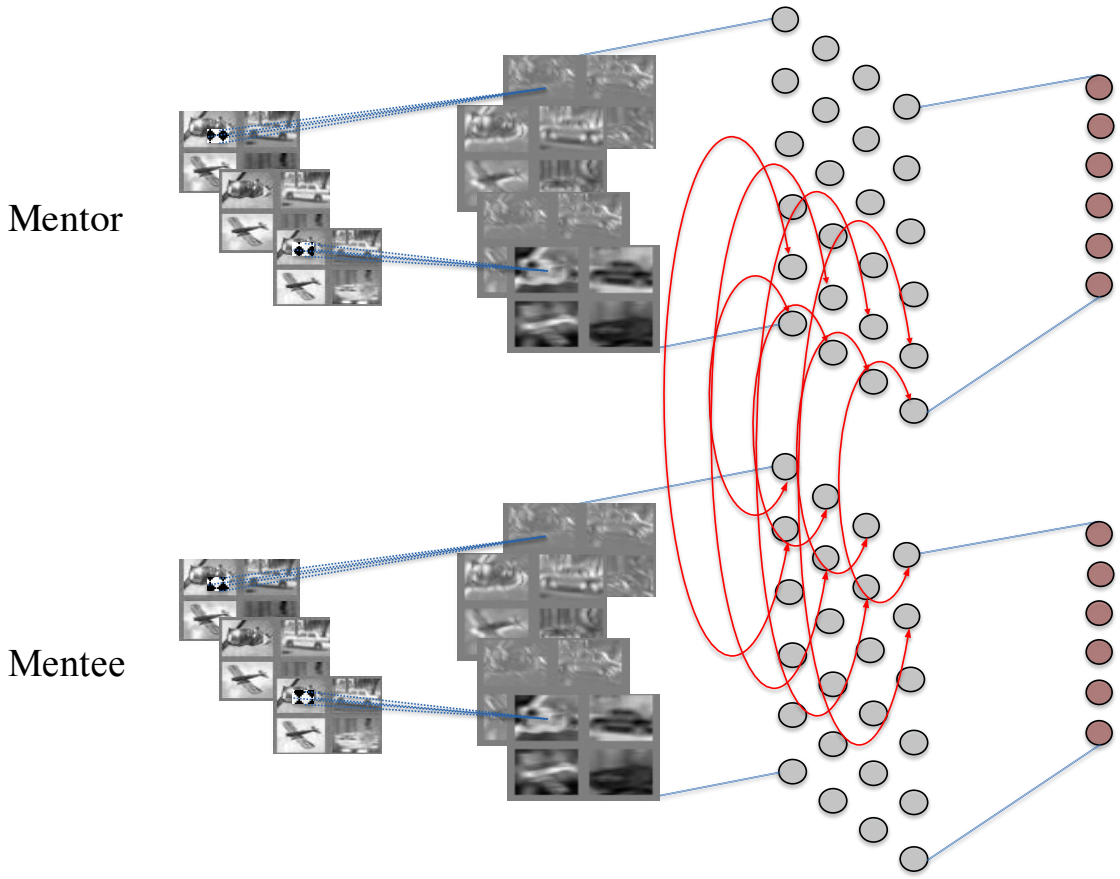


Figure 4.1: Mentor Mentoring Mentee on the Second Hidden Layer.

Figure 4.1 shows a conceptualization of this idea. In this chapter, we try and make a shallower mentee network learn the same representation as a larger, well-trained mentor network at various depths of its network hierarchy. Mentorship happens by tagging on to the loss of the mentee network, a dissimilarity loss for each layer that we want mentored. To the best of our knowledge, there hasn't been any work that has regularized more than one layer this way. There also hasn't been any work that has trained a mid-sized network from a larger and deeper network from scratch. We study some idiosyncratic properties for some novel configurations of mentee networks. We argue that such mentoring avoids weight explosion. Even while using smaller mini-batches, mentee networks get ample supervision and are capable of stable learning

even at high learning rates. We show that mentee networks produce a better generalization performance than an independently learnt baseline network. We also show that mentee networks are better transferable than the independently learnt baselines and are also a good initializer. We also show that mentee networks can learn good representations from very little data and sometimes even without supervision from a dataset in an unsupervised fashion.

The rest of the chapter is organized as follows: section 4.2 discusses related works, section 4.3 details the mentored learning, section 4.4 discusses designs for experiments, section 4.5 produces results and section 4.6 provides concluding remarks.

4.2 Related Works

Hinton et al., tried to make networks portable by learning the softmax outputs of a larger well-trained network along with the label costs Hinton *et al.* (2014). This was previously explored using logits by Caruana et al., Ba and Caruana (2014); Bucilu *et al.* (2006). By directly learning the softmax layers, they were forcing the softmax layer of a smaller network to mimic the same mapping as that of a larger network onto the label space. In a way they tried to learn a better second and third guesses. They called this *dark knowledge*, as the knowledge so learnt is only available to the larger network. By attempting to learn the softmax layer, they were able to transfer or *distil* knowledge between the two networks. The drawback of this work is that it only works as long as the larger network is already well-trained and stable. They relied upon the network's predictive softmax layer being learnt perfectly on the target dataset and propagate that knowledge. This also assumes that there are relationships between classes to be exploited. While this may work in cases where this is true, such as in character recognition or in voice recognition, it doesn't work in most object detection datasets where the relationship between classes is not a given in terms of its appear-

ance features [1] . They also distil only the softmax labels and not the representational space itself. This also requires that the smaller network is capable of training in a stable manner. Dark knowledge is extended upon by several previous works Wang *et al.* (2015); Romero *et al.* (2014); Balan *et al.* (2015); Chan *et al.* (2015). One extension of this work that we generalize in this article is using layer-wise knowledge transfer for one layer in the middle of the network. This was used to show that thinner and deeper network can be trained with better regularization Romero *et al.* (2014). Another method uses a similar one-layer regularizer as knowledge transfer between a RNN and a CNN Chan *et al.* (2015). Mentored training has also been shown to be extremely useful when training LSTMs and RNNs with an independent mentor supervision Wang *et al.* (2015). All these methods discussed above are essentially the same technique as the dark-knowledge method extended beyond just the softmax layer. All of these methods have fixed one-layer regularizations and although trivial, we generalize this for many layers. Their mentee networks are typically much deeper and complex than their mentors and they use these as a means to build more complex models (albeit thinner as in the case of FitNets Romero *et al.* (2014)). There has been no study to the best of our knowledge that builds less complex (both thinner and shallower) models with the same capability as larger models. Also, neither has there been a study that studies various properties of these networks nor those that show the transferability and generality of these networks.

4.3 Generalized Mentored Learning

Let us first generalize all of the methods that use this knowledge transfer as follows: Consider a large mentor network with n layers $\mathcal{M}_n$. Suppose we represent the k^{th} neuron activations of the i^{th} layer in the network as $\mathcal{M}_n(i,k)$. Consider a smaller

[1] We tried this approach on Caltech101 and couldn't get reliable results.

mentee network with m [2] layers $\mathcal{S}_m$. Suppose that $\mathcal{M}$ is already well-trained and stable on a general enough dataset $\mathcal{D}$. Now consider that we are using $\mathcal{S}$ to learn classification [3] on a newer dataset d_1 which is less general and much smaller than $\mathcal{D}$ as determined a priori. Although this is not a constraint, having a smaller and less general dataset emphasizes the core need where such mentored learning is most useful.

$\forall \;\; l \leq n$ and $j \leq m$, we can define a probe as an error measure between $\mathcal{M}_n(l)$ and $\mathcal{S}_m(j)$. This error can be modelled as an RMSE error as follows,

$$\Psi(l,j) = \sqrt{\frac{1}{a}\sum_{i=0}^{a}(\mathcal{M}_n(l,i) - \mathcal{S}_n(j,i))^2}, \tag{4.1}$$

where a is the minimum number of neurons between $\mathcal{M}_n(l,.)$ and $\mathcal{S}(j,.)$. If the neurons were convolutional, we consider element-wise errors between filters. By adding this cost to the label cost of the network $\mathcal{S}$ during back propagation, we learn not just a discriminative enough representation for the samples and labels of d_1, but also for layers j in a pre-determined set of layers, a representation closer to the one produced by $\mathcal{M}$. Some implementations of such loses in literature tend to learn a regressor instead of simply adding the loss, but we concluded from our experiments that the computational requirements of such regressors do not justify their contributions. Adding a regressor would involve embedding the activations of the mentor and the mentee onto a common space and minimizing the distances between those embeddings. We quite simply circumvent that and consider the minimum number of matching neurons. This enables us to have a slimmer, fatter or same sized mentee. Suppose d_1^b is the b^{th} mini-batch of data from the dataset d_1 and suppose we have a

[2] Although we adhere strictly to $m < n$, without losing any generality, we could have any m or n. In fact $m > n$ with only one probe would be the special case of FitNets Romero *et al.* (2014).

[3] Although we only consider the task of classification, the methods proposed are applicable to many forms of learning.

pre-determined set of probes B, which is a set of tuples of layers from ($\mathcal{M}$, $\mathcal{S}$). The overall network cost is,

$$e = \alpha_t \mathcal{L}_s(d_1^b) + \beta_t \sum_{\forall (l,j) \in B} \Psi(l,j) + \gamma_t \Psi(n,m), \tag{4.2}$$

where $\mathcal{L}(.)_s$ is the network loss of that mini-batch, α_t and β_t weighs the two losses together to enable balanced training and γ_t is the weight of the probe between the two (temperature) softmax layers. $\alpha_t = g_\alpha(t)$, $\beta_t = g_\beta(t)$ and $\gamma_t = g_\gamma(t)$ are annealing functions parametrized by the iteration t under progress. Although most methods in the literature use constants for α_t, β_t and γ_t, we found it preferable to retain $g_\alpha(t) = 1, \forall t$ throughout and anneal β and γ linearly. We discuss the value and the need for these parameters in detail further.

Since $\mathcal{M}$ is pre-trained and stable, the second and third terms of equation 4.2 are penalties for the activations of those layers in $\mathcal{S}$ not resembling the activations of the probed layer from $\mathcal{M}$ respectively. These losses as defined by equation 4.1 are functions of the weights of those layers from $\mathcal{S}$ only. They restrict the weights within a proximity or region, that produces activations that are known for the mentor to be better activations. This restricting behaviour acts as a guided regularization process, allowing the weights to explore in a direction that the mentor thinks is a good direction, while still not letting the gradients to explode or vanish.

For a particular weight $w \in \mathcal{S}$ at any layer, a typical update rule without the probe is,

$$w^{t+1} = w^t - \eta \frac{\partial}{\partial w} \mathcal{L}_s, \tag{4.3}$$

where t is some iteration number, η is the learning rate and assuming $\alpha_t = 1, \forall t$. The update rule with mentored probes is,

$$w^{t+1} = w^t - \eta \Big[\alpha_t \frac{\partial}{\partial w} \mathcal{L}_s + \beta_t \sum_{\forall (l,j) \in B} \frac{\partial}{\partial w} \Psi(l,j) + \gamma_t \frac{\partial}{\partial w} \Psi(n,m) \Big]. \tag{4.4}$$

The last two terms add a guided version of a *noise* that decreases with each iteration. While at earlier stages of training, this allows the weights to explore the space, it also restricts the weights from exploding because the direction that the weights are allowed to explore is controlled by the mentor. The freedom to explore tightens up as the as learning proceeds, provided $g_\beta(t)$ is a monotonically annealing function with respect to t. Note that even though to calculate these error gradients we need one forward propagation through $\mathcal{M}$, we do not back propagate through $\mathcal{M}$. This is a penalty on the weights, even though we are using the activations to penalize the weights indirectly. Although mentee networks can be further regularized with l_2, l_1, dropouts and batch normalizations, it is recommended that the mentee networks imposes additional regularizations mirroring the mentor networks for better learning.

Different Configurations of Mentee Networks

Different combinations of α, β and γ produces different characteristics of mentee networks. Equation 4.4 can be seen as learning with three different learning rates, $\alpha * \eta$, $\beta * \eta$ and $\gamma * \eta$. We can simulate using these three parameters, two idiosyncratic personalities of mentee networks: an obedient network and an adamant network. An obedient network is a network that focuses on learning the representation more than the label costs at the beginning stages and once a good representation is learnt, it focuses on learning the label space. It tends towards being over-regularized and its regularization relaxes with epochs. An adamant network is a network that focuses almost immediately on the labels as much as learning the representation, but its focus is positively towards learning the label only. The learning rates of these personalities are shown in figure 4.2.

An independent network can be considered as a special case of the adamant network where probe weights are ignored ($\beta = 0,\ \gamma = 0,\ \forall t$). The other extreme case

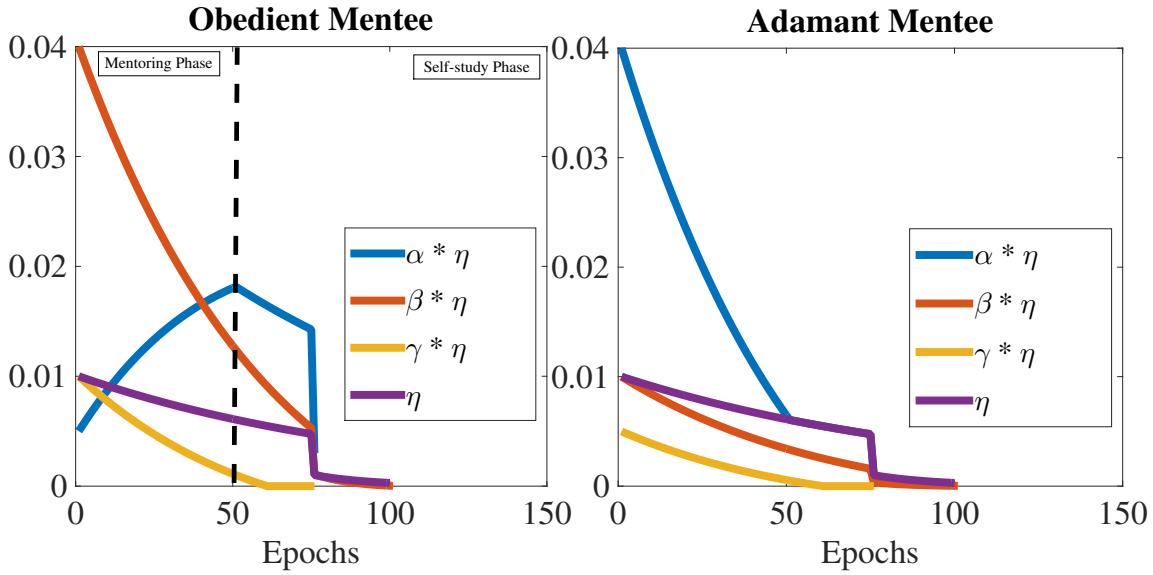


Figure 4.2: Annealing α, β and η while learning for an obedient and an adamant network.

of an obedient network is perhaps a gullible network that learns just the embedding space of the mentor. Gullible networks are also a good way to initialize a network in an unsupervised mentoring fashion. Consider a dataset d_2, that does not have any labels. Neither the mentor nor the mentee could potentially learn any discriminative features. Using just the probes we could build an error function that could make the smaller mentee network still learn a good representation for the dataset. We use the information from the parent network to learn a good representation for d_2 by simply back propagating the second term of equation 4.2 alone. These gullible mentees come in really handy when the dataset has considerably less samples to be supervised with. Unsupervised mentoring is also an aggressive way to initialize a network and is often helpful in learning large networks in a stable manner with a stable initialization.

Typically the deeper one goes, the more difficult it becomes to learn the activations and the costs saturate quickly. The softmax layer is the most difficult to learn. To our surprise we find that probe costs converge much sooner than the label costs, leading us to believe that the representations being mentored are indeed relevant as long as

the datasets share common characteristics. There is a plethora of such configurations that could be tried and many unique characteristics discovered. In this article we limit ourselves to only those that enable us stability during learning and focus on those that help us with better generalizations.

For learning large networks we prefer the use of obedient networks as obedient networks are heavily regularized at the beginning leading to careful initialization and stabilization of the network before learning of labels takes over. We call the stabilization phase as the mentoring phase and the rest, self-study phase. During the mentoring phase learning is slow but steady. In most cases, $\alpha * \eta$ is an increasing function due to the aggressive climb of α. The annealing of these rates for a typical obedient mentee and an adamant mentee are shown in figure 4.2. We also find that typically the later layers are more stubborn in being mentored than earlier layers. Although this is typically to be expected, more obedience may be enforced by choosing higher β values for layers that are deeper in the network.

4.4 Design of Experiments

We evaluate the effectiveness of mentorship through the following experiment designs:

4.4.1 Effectiveness

To demonstrate the effectiveness of learning, we first train a larger network on a dataset. Using this network as a mentor, we train the mentee network on the same dataset. Unlike those in literature, we choose mentee networks that are generally much smaller than the mentor. We show that this generalizes at least as well as an independent network of the exact same architecture regularized not by mentor, but by batch normalization, l_2 and l_1 norms and dropouts. Training mid-sized networks

on small datasets are often difficult. To our best knowledge we have provided our best effort in meticulously learning all the networks. For learning an independent network often we spent additional effort in adjusting the learning rates at the opportune moments. We show that mentee networks outperforms the independent networks and even at the worst case performs as well as the independent networks.

4.4.2 Generality of the Learnt Representations

To demonstrate that the network learns a more general representation, we gather a pair of datasets of seemingly similar characteristics with one more general or larger than the other. We train the mentor with the more general dataset first and then fine tune it on the less general dataset. We then train both the independent and the mentee nets on the less general dataset and demonstrate again that at worst the mentee net performs the same as the independent net.

We then proceed to fine tune the classifier layer of both the mentee net and the independent net using the more general dataset but since the other layers are not allowed to change, the mentee net does not have any additional supervision. This tests the quality of the features learnt by these networks on a more general and more difficult dataset. For the sake of our experiments we consider the pairing of (Cifar-10 - Cifar-100) and (Caltech-101 - Caltech-256) Krizhevsky and Hinton (2009); Griffin *et al.* (2007). We assume that Cifar-100 is more general than Cifar-10 and Caltech-256 is more general than Caltech-101.

Additionally, we conduct another experiment where we try to learn from a mentor network trained with the full MNIST dataset, a mentee network that only has supervision from a part of the dataset LeCun *et al.* (1998a). The independent network also in this case, learns with the same redacted dataset. We redact the dataset by only having p samples for each class in the dataset where $p \in \{500, 250, 100, 50, 10, 1\}$.

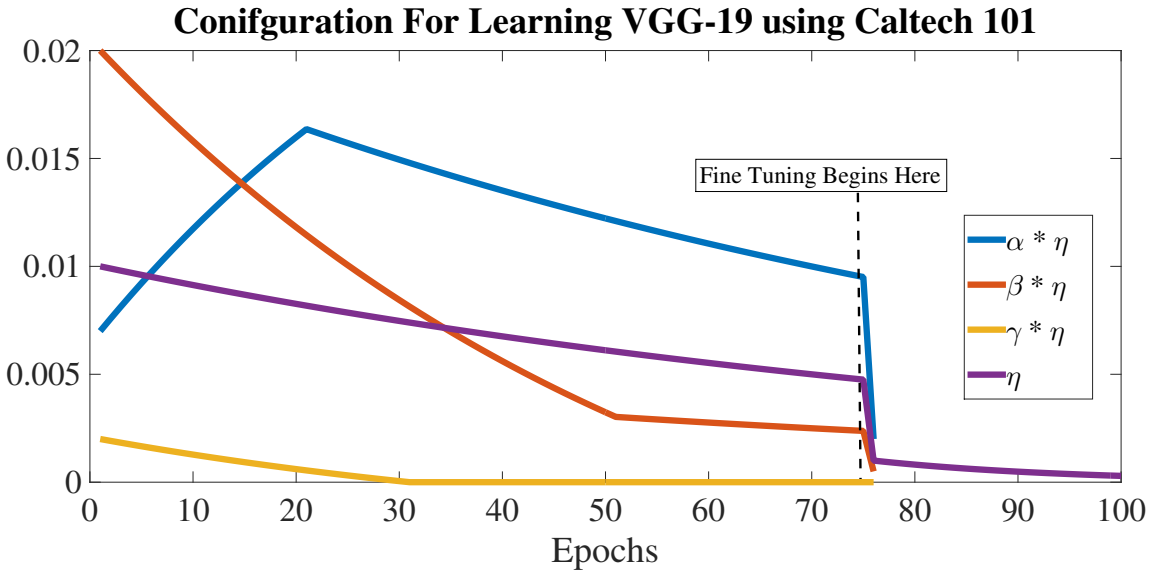


Figure 4.3: Annealing α, β and η while learning VGG-19 space for Caltech-101. We used an obedient network.

$p = 1$ is essentially an ambitious goal of 1-shot learning from scratch using a deep network. We also try this with a mentee network that is initialized by unsupervised mentoring from the same mentor network. We acknowledge that the comparison with unsupervised mentoring is unfair because the mentee net is initialized by the mentor with information filtered from data that is unavailable for the independent network. The latter results are to demonstrate that unsupervised mentoring could learn an effective feature space even without labels and with very less samples.

4.4.3 Learning the VGG-19 Representation

In particular, while learning classification on the Caltech101 dataset, we try to learn the same representation as the popularly used VGG-19 network at various levels of the network hierarchy Simonyan and Zisserman (2014b). VGG-19 network's 4096 dimensional representation is one of the most coveted and iconic image features in computer vision at the time of the writing of this article. The VGG-19 network has 16 convolutional layers and 2 fully-connected layers the last of which produces the

4096 dimensions of features upon which many other works have been built.

We try to learn the same 4096 dimensional representation of the VGG-19 network using ambitiously less number of layers. For the (Caltech-101-Caltech256) dataset pairs in all our experiments, there is no explicit mentor network that we learnt. We simply set $g_\gamma(t) = 0, \forall t$ and learnt with probes without retraining the VGG-19 network. In a way we are attempting to learn VGG-19's view of the Caltech-101 dataset and are probing into the representational frame of the VGG-19 network. We used a relatively obedient student as shown in figure 4.3 for this case.

4.4.4 Implementation Details

The independent networks were all regularized with a l_1 and l_2 penalties with a weight of $1e^{-4}$, which seems to give the best results. On all networks we also applied parametrized batch norm for both fully connected and convolutional layers and dropouts with rate of $p = 0.5$ for the fully connected layers Ioffe and Szegedy (2015); Srivastava *et al.* (2014). We find that dropout and bath norm together help in avoiding over-fitting. All our activation functions were rectified linear units Nair and Hinton (2010). For learning the mentee network we start with learning rates as high as 0.5, for the larger independent networks we are forced a learning rate of 0.001, while for the smaller experiments we were able to go as high as 0.01, since the batch sizes were larger. During training, if ever we ran into exploding gradients or NaNs, we reduce the learning rate by ten times, reset the parameters back to one epoch ago and continue training. We train until 75 epochs after which we reduce the learning rate by a hundred times and continue fine-tuning until early stopping. Unless early stopped, we train for 150 epochs. All our initializations were from a 0-mean Gaussian distribution, except the biases which were initialized with zeros. The experiment set-up was designed using Theano v0.8 and the programs were written

by ourselves [4] Bastien *et al.* (2012). The experiments with MNIST datasets were conducted on a Nvidia GT 750M GPU, the others on an Nvidia Tesla K40c GPU, with cuDNN 3007 and Nvidia CUDA v7.5. The mini-batch sizes for all the MNIST and cifar experiments were 500 (unless forced by small dataset size in which case we performed batch descent instead of the usual stochastic descent). The mini-batch sizes for all Caltech experiments were 36, with images resized to $224X224$ so as to the fit the VGG-19 requirement. Apart from normalization and mean-subtraction, no other pre-processing were applied to any of the images. For the Caltech experiments we used Adagrad with Polyak's momentum Polyak (1964); Green *et al.* (2013). For the experiments that were smaller networks we used RMSprop with Nesterov's accelerated gradient Dauphin *et al.* (2015); Nesterov (1983). It is to be noted that we chose to use vanilla networks that are as simple as possible so as to enable us to compare against a baseline which is also vanilla. Since our aim is not to achieve state-of-the-art accuracies on any datasets, we didn't implement several techniques that are commonly applied to boost the network performances in modern day computer vision. The purpose of these experiments is to unequivocally demonstrate that among networks that learn from scratch, one that is mentored can perform better and learn more general features than one that is not.

4.5 Results

The results are split across two tables based on the network architectures. The smaller experiments on a 5 layer network are shown in figure 4.4 and the larger ones in figure 4.5. The $\rightarrow$ symbol shows which layers are probed and from where.

In figure 4.4, the results clearly demonstrate the strong performance of the mentee networks over the independent networks. In the cifar experiments we under-weighted

[4] Code is available at our GitHub page.

Network		Mentor	Mentee	Independent
Architecture	Convolutional Layers Activation: ReLU Stride: 1 Max Pooling: 2	Kernel Size: 5 Neurons: 20	Kernel Size: 5 Neurons: 20	Kernel Size: 5 Neurons: 20
		Kernel Size: 3 Neurons: 50		
	Fully Connected Layers Activation: ReLU Dropout Input Rate: 0.5	Neurons: 800	Neurons: 800	Neurons: 800
		Neurons: 800	Neurons: 800	Neurons: 800
	Output Layer	Neurons: 10/100	Neurons: 10/100	Neurons: 10/100
Accuracies	Trained from scratch Cifar 10	79.36 %	68.5 %	68.58 %
	Fine-tuned last layer only Cifar 100	41.21%	33.2 %	26.67%
	MNIST - 500	MNIST 99.59%	97.73% unsupervised mentoring 98.2%	97.71%
	MNIST-250		97.47 % unsupervised mentoring 97.88%	96.89 %
	MNIST-100		97.42% unsupervised mentoring 96.01%	95.12%
	MNIST-50		92.95% unsupervised mentoring 96.80%	90.96%
	MNIST-10		78.5 % unsupervised mentoring 96.7%	75.3%
	MNIST - 1		48.5% unsupervised mentoring 96.7%	41.5%

Figure 4.4: Architecture and Results for the Experiments with CIFAR and MNIST Datasets.

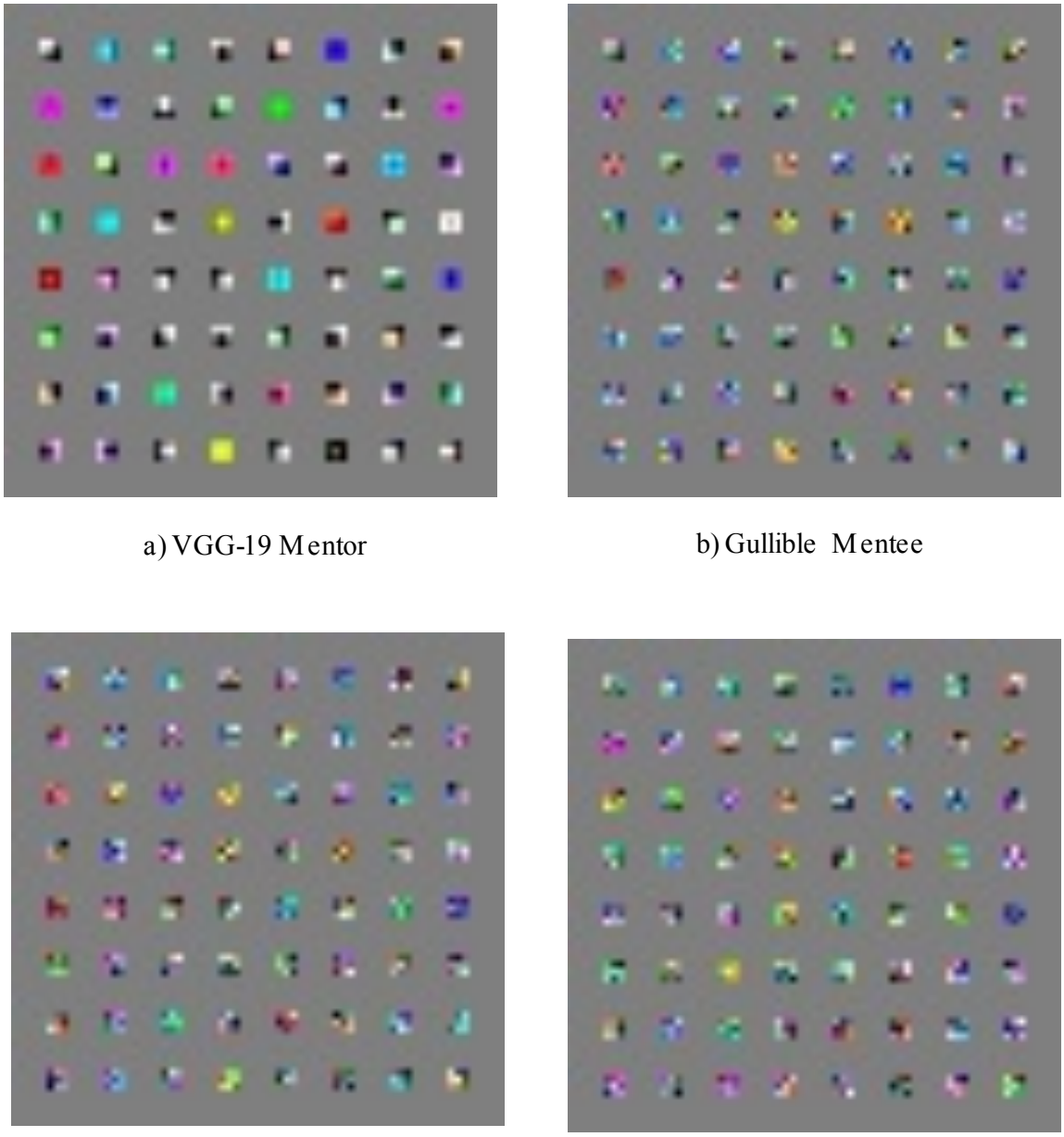
γ purposely as we didn't want to propagate the 20% of error from the mentor network on to the mentee network. The results on Cifar 10 from scratch seem to indicate that both networks have reached the best possible performance for that architecture. We believe with the amount of supervision already provided from the 40,000 training images, mentoring is not as effective. When there is already ample supervision, mentoring is ineffective, or rather unwanted, albeit it doesn't hurt. While fine-tuning on cifar 100, we find that there are great gains to be made.

We find a similar trend with the MNIST experiments also. The less data there is, the higher the gain of the mentee networks. Note that even though mentee networks are regularized, care was taken to ensure that they both go through the exact same number of iterations at the exact same learning rate. We also found that unsuper-

Network		**Mentor (VGG-19)**	**Mentee**	**Independent**
Architecture	Convolutional Layers Stride: 1 Kernel Size: 3 Activation ReLU	Neurons: 64 Max Pooling: 1	→	
		Neurons: 64 Max Pooling: 2	Neurons: 64 Max Pooling: 2	Neurons: 64 Max Pooling: 2
		Neurons: 128 Max Pooling: 1		
		Neurons: 128 Max Pooling: 2	Neurons: 128 Max Pooling: 2	Neurons: 128 Max Pooling: 2
		Neurons: 256 Max Pooling: 1		
		Neurons: 256 Max Pooling: 1		
		Neurons: 256 Max Pooling: 1	Neurons: 256 Max Pooling: 2	Neurons: 256 Max Pooling: 2
		Neurons: 256 Max Pooling: 2		
		Neurons: 512 Max Pooling: 1		
		Neurons: 512 Max Pooling: 1		
		Neurons: 512 Max Pooling: 1	Neurons: 512 Max Pooling: 2	Neurons: 512 Max Pooling: 2
		Neurons: 512 Max Pooling: 2		
		Neurons: 512 Max Pooling: 1		
		Neurons: 512 Max Pooling: 1	Neurons: 512 Max Pooling: 2	Neurons: 512 Max Pooling: 2
		Neurons: 512 Max Pooling: 1		
		Neurons: 512 Max Pooling: 2	Neurons: 512 Max Pooling: 2	Neurons: 512 Max Pooling: 2
	Fully Connected Layers Activation: ReLU Dropout Input Rate: 0.5	Neurons: 4096	→	
			Neurons: 4096	Neurons: 4096
		Neurons: 4096	→	
			Neurons: 4096	Neurons: 4096
	Softmax Layer		Neurons: 102/256	Neurons: 102/256
Accuracies	Trained from scratch on Caltech 101	N/A	56.16%	45.46%
	Fine-tuned last layer only for Caltech 256	N/A	66.12 %	55.45%

Figure 4.5: Architecture and results for the Experiments with Caltech Datasets.

vised mentoring always keeps the learning at a very high standard although as was discussed in section 4.4.2 there was additional supervision on the entire dataset from the unsupervised mentoring, which is unfair.



a) VGG-19 Mentor

b) Gullible Mentee

c) Obedient Mentee

d) Adamant Mentee

Figure 4.6: VGG-19 first layer filters and filters probed using Caltech101 for a Gullible, Obedient and an Adamant mentee after only one epoch of training. We recommend viewing this image on a computer monitor.

In the experiments with the Caltech101 datasets, we find that the mentee networks perform better than the vanilla network. The mentee network was also able

to perform significantly better than the independent network when only the classifier/mlp sections were allowed to learn the Caltech256 dataset with representation learnt from Caltech101. This proves the generality of the feature space learnt. With an even obedient student, we were able to learn the feature space of the VGG-19 network to a remarkable degree. While with the first convolutional layer we were able to learn to a minimum rmse or 0.0023 from 6.54 at random. With the last two layers we were able to learn upto a rmse of 2.04 from 12.76 at random.

Figure 4.6 shows the filters learnt after one epoch for a gullible network, an obedient network and an adamant network. All these networks were initialized with same random values at their inception. We can easily notice that the gullible network already sway towards the VGG-19 filters. In obedient mentee, we notice that most corner detector features are already swaying towards the mentee network but more complex features are not swaying as much as the gullible network. To our surprise we notice that even in an adamant network corner detectors are swaying towards VGG-19. This shows that even with low weights, the first layer features are learning the VGG-19's representation. It is to be noted that we are not learning the weights directly, but are learning the activations produced by the VGG-19 network for the Caltech101 dataset that leads us to learn the same filters as the VGG-19. This implies that corner features are more general among the Imagenet dataset, which VGG-19 was trained on, and the Caltech101 dataset, which explains why they are learnt earlier than others.

4.6 Conclusions

While the use of large pre-trained networks will continue to remain popular, because of the ease in just copying a network and fine-tuning the last layers, we believe that there is still a need for learning small and mid-sized networks from scratch. We

also recognize the difficulty involved in reliably training deep networks with very few data samples. One way to meet the best of both worlds is by using a mentored learning approach. In our study, we find that a shallower mentee network was able to learn a new representation from scratch while being regularized by the mentor network's activations for the same input samples. We found that such mentoring provided much stabler training even at higher learning rates. We noted some special cases of these networks and recognize some idiosyncratic personalities. We extended one of these to be able to perform as an unsupervised initialization technique. We showed through compelling experiments, the strong performance and generality of mentor networks.

Chapter 5

INCREMENTAL LEARNING

5.1 Introduction

Animals and humans learn incrementally. A child grows its vocabulary of identifiable concepts as different concepts are presented, without forgetting the concepts with which they are already familiar. Antithetically, most supervised learning systems work under the omniscience of the existence of all classes to be learned, prior to training. This is crucial for learning systems that produce an inference as a conditional probability distribution over all known categories.

Incremental supervised learning though reasonably studied, lacks a formal and structured definition. One of the earliest formalization of incremental learning comes from the work of Jantke Jantke (1993). In this article the author defines incremental learning roughly as systems that *"have no permission to look back at the whole history of information presented during the learning process"*. Immediately following this statement though is the relaxation of the definition: *"Operationally incremental learning algorithms may have permission to look back, but they are not allowed to use information of the past in some effective way"*, with the terms *information* and *effective* not being sufficiently well-defined. Subsequently, other studies made conforming or divergent assumptions and relaxations thereby adopting their own characteristic definitions. Following suit, we redefine a more fundamental and rigorous incremental learning system using two fundamental philosophies: data membrane and domain agnosticism.

Consider there are two sites: the base site $\mathcal{S}_b$ and the incremental site $\mathcal{S}_i$ each

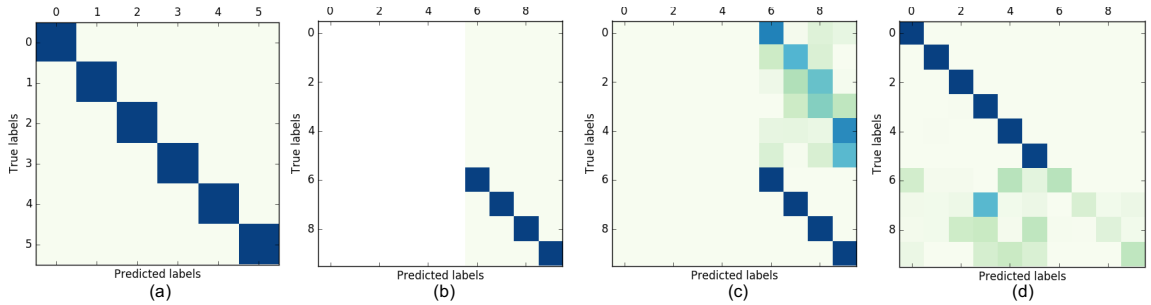


Figure 5.1: Catastrophic forgetting: Figure (a) is the confusion matrix of a network N_b, trained and tested on data from a subset containing only samples of labels $0 \ldots 5$. Figure (b) is the confusion matrix of a network initialized with the weights of trained N_b, re-trained with data from classes $6 \ldots 9$ and tested on the same label space. No testing samples were provided for the classes $0 \ldots 5$. Figure (c) is the same network as (b) tested on the entire label space. Figure (d) is similar to (c) but trained with a much lower learning rate. These confusion matrices demonstrate that a neural network retrained on new labels without supplying it data from the old data subset, forgets the previous data, unless the learning rate is very measured and slow as was the case in (d). If the learning rate were slow, though the old labels are not forgotten, new labels are not effectively learned.

with ample computational resources. $\mathcal{S}_b$ possesses the base dataset $\mathcal{D}_b = \{(x_l^b, y_l^b), l \in \{1, 2, \ldots n\}\}$, where $x_l^b \in \mathbb{R}^d, \forall l$ and $y_l^b \in \{1, 2, \ldots j\}, \forall l$. $\mathcal{S}_i$ possesses the increment dataset $\mathcal{D}_i = \{(x_l^i, y_l^i), l \in \{1, 2, \ldots m\}\}$, where $x_l^i \in \mathbb{R}^d, \forall l$ and $y_l^i \in \{j+1, j+2, \ldots c\}, \forall l$ and $y_l^i \notin \{0, 1, \ldots j\}, \forall l$.

Property 1. *$\mathcal{D}_b$ is only available at $\mathcal{S}_b$ and $\mathcal{D}_i$ is only available at $\mathcal{S}_i$. Neither set can be transferred either directly or as features extracted by any deterministic encoder, either in whole or in part to the other site, respectively.*

$\mathcal{S}_b$ is allowed to train a discriminative learner N_b using $\mathcal{D}_b$ and make N_b available to the world. Once broadcast, $\mathcal{S}_b$ does not maintain N_b and will therefore not support queries regarding N_b. Property 1 is referred to as the *data membrane*. Data membrane ensures that $\mathcal{S}_i$ does not query $\mathcal{S}_b$ and that no data is transferred either in original form or in any encoded fashion (say as feature vectors). The generalization set at $\mathcal{S}_i$ contains labels in the space of $y \in \{1 \ldots c\}$. This implies that though $\mathcal{S}_i$, has no

data for training the labels $1 \dots j$, the discriminator N_i trained at $\mathcal{S}_i$ with $\mathcal{D}_i$ alone is expected to generalize on the combined label space in the range $1 \dots c$. $\mathcal{S}_i$ can acquire N_b and other models from $\mathcal{S}_b$ and infer the existence of the classes $y \in \{1, 2, \dots j\}$ that N_b can distinguish. Therefore incremental learning differs from the problem of zero-shot novel class identification.

A second property of multi-class incremental learning is domain agnosticism, which can be defined as follows:

Property 2. *No priors shall be established as to the dependencies of classes or domains between $\mathcal{D}_b$ and $\mathcal{D}_i$.*

Property 2 implies that we cannot presume to gain any knowledge about the label space of $\mathcal{D}_b$ ($\{0 \dots j\}$) by simply studying the behaviour of N_b using $\mathcal{D}_i$. In other words, the predictions of the network N_b does not provide us meaningful enough information regarding $\mathcal{D}_i$. This implies that the conditional probability distribution across the labels in $y \in \{0 \dots j\}$, $P_{N_b}(y|x)$ for $(x, y) \in \mathcal{D}_i$ produced by N_b, cannot provide any meaningful inference to the conditional probability distribution across the labels $y \in \{j + 1 \dots c\}$ when generalizing on the incremental data. For any samples $x \in \mathcal{D}_i$, the conditional probability over the labels of classes $y \in \{0 \dots j\}$ are meaningless. Property (2) is called *domain agnosticism*.

From the above definition it is implied that sites must train independently. The training at $\mathcal{S}_i$ of labels $y \in \{j + 1 \dots c\}$ could be at any state when $\mathcal{S}_b$ triggers site $\mathcal{S}_i$ by publishing its models, which marks the beginning of incremental training at $\mathcal{S}_i$. To keep experiments and discussions simpler, we assume the worst case scenario where the site 2 does not begin training by itself, but we will generalize to all chronology in the later sections.

We live in a world of data abundance. Even in this environment of data affluence,

we may still encounter cases of scarcity of data. Data is a valuable commodity and is often jealously guarded by those who posses it. Most large institutions and organizations that deploy trained models, do not share the data with which the models are trained. A consumer who wants to add additional capability is faced with an incremental learning problem as defined. In other cases, such as in military or medicine, data may be protected by legal, intellectual property and privacy restrictions. A medical facility that wants to add the capability of diagnosing a related-but-different pathology to an already purchased model also faces a similar problem and often has to expend large sums of money to purchase an instrument with this incrementally additional capability. All these scenarios are plausible contenders for strict incremental learning following the above definition. The data membrane property ensures that even if data could be transferred, we are restricted by means other than technological, be it legal or privacy-related that prevents the sharing of data across sites. The domain agnosticism property implies that we should be able to add the capability of predicting labels to the network, without making any assumptions that the new labels may or may not hold any tangible relationship to the old labels.

A trivial baseline: Given this formalism, the most trivial incremental training protocol would be to train a machine at $\mathcal{S}_b$ with $\mathcal{D}_b$, transfer this machine (make it available in some fashion) to $\mathcal{S}_i$. At $\mathcal{S}_i$, initialize a new machine with the parameters of the transferred machine, while alerting the new machine to the existence of classes $j+1, \ldots c$ and simply teach it to model an updated conditional probability distribution over classes $\{1, 2, \ldots c\}$. A quick experiment can demonstrate to us that such a system is afflicted by a well-studied problem called catastrophic forgetting. Figure 5.1 demonstrates this effect using neural networks. This demonstrates that without supplying samples from $\mathcal{D}_b$, incremental training without catastrophic forgetting at $\mathcal{S}_i$ is difficult without relaxing our definition.

To avoid this, we propose that the use of generative models trained at $\mathcal{S}_b$, be deployed at $\mathcal{S}_i$ to hallucinate samples from $\mathcal{D}_b$. The one-time broadcast from $\mathcal{S}_b$ could include this generator along with the initializer machine that is transferred. While this system could generate samples-on-demand, we still do not have targets for the generated samples to learn classification with. To solve this problem, we propose the generation of supervision from the initializer network itself using a temperature-raised softmax. A temperature-raised softmax was previously proposed as a means of distilling knowledge in the context of neural network compression Hinton *et al.* (2015). Not only does this provide supervision for generated samples, but will also serve as a regularizer while training a machine at $\mathcal{S}_i$, similar to the fashion described in Hinton *et al.* (2015).

In summary this chapter provides two major contributions: 1. A novel, uncompromising and practical definition of incremental learning and 2. a strategy to attack the defined paradigm through a novel sampling process called *phantom sampling*. The rest of this chapter is organized as follows: section 5.2 outlines the proposed method, section 5.3 discusses related works on the basis of the properties we have presented, section 5.4 presents the design of our experiments along with the results and section 5.6 provides concluding remarks.

5.2 Proposed Method

Our design begins at $\mathcal{S}_b$. Although $\mathcal{S}_b$ and $\mathcal{S}_i$ may train at various speeds and begin at various times, in this presentation we focus on the systems that mimic the following chronology of events:

1. $\mathcal{S}_b$ trains a generative model G_b and a discriminative model N_b for $P(x^b)$ and $P_{N_b}(y|x^b)$ using $(x^b, y^b) \in \mathcal{D}_b$, respectively.

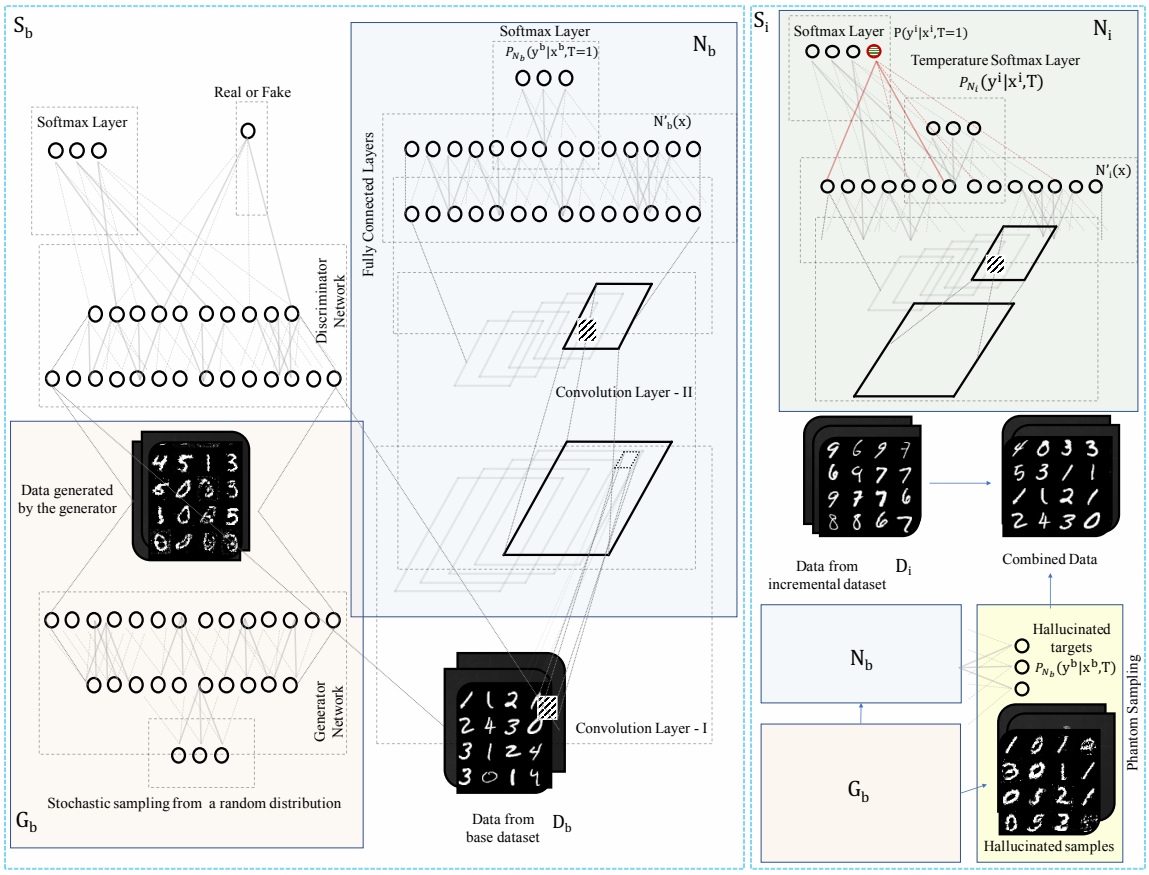


Figure 5.2: Sites $\mathcal{S}_b$, $\mathcal{S}_i$ and the networks that they train respectively. The networks G_b and N_b are transferred from $\mathcal{S}_b$ to $\mathcal{S}_i$ and work in feed-forward mode only at $\mathcal{S}_i$. In this illustration using MNIST dataset, $j = 5$. Classes $[0 \ldots 5]$ are in $\mathcal{D}_b$ and classes $[6 \ldots 9]$ are available in $\mathcal{D}_i$

2. $\mathcal{S}_b$ broadcasts G_b and N_b.

3. $\mathcal{S}_i$ collects the models G_b and N_b and initializes new model N_i with the parameters of N_b adding new random parameters as appropriate. Expansion using new random parameters is required since, N_i should make predictions on a larger range of labels.

4. Using $\mathcal{D}_i$ together with phantom sampling from G_b and N_b, $\mathcal{S}_i$ trains the model N_i until convergence.

This is an asymptotic special case of the definition established in the previous section and is therefore considered. Other designs could also be established and we will describe briefly a generalized approach in the latter part of this section. While the strategy we propose could be generalized to any discriminatory multi-class classifier, for the sake of clarity and being precise, in this article we restrict our discussions to the context of deep neural networks.

The generative model, G_b models $P(x|\mathcal{D}_b)$. In this article we considered networks that are trained as simple generative adversarial networks (GAN) for our generative models. GANs have recently become very popular for approximating and sampling from distributions of data. GAN was originally proposed by Goodfellow et. al, in 2014 and has since seen many advances Goodfellow *et al.* (2014). We consider the GANs proposed in the original article by Goodfellow et. al, for the sake of convenience. We use a simple convolutional neural network model as the discriminator N_b. Figure 5.2 shows the overall architecture of our strategy with G_b and N_b within the $\mathcal{S}_b$ capsule. As can be seen, G_b attempts to produce samples that are similar to the data and N_b learns a classifier using the softmax layer that is capable of producing $P_{N_b}(y^b|x^b)$ as follows:

$$\begin{bmatrix} P_{N_b}(y=1|x^b) \\ \vdots \\ P_{N_b}(y=j|x^b) \end{bmatrix} = \frac{1}{\sum_{p=1}^{j} e^{w_b^{(p)} N_b'(x)}} \begin{bmatrix} e^{w_b^{(1)} N_b'(x)} \\ \vdots \\ e^{w_b^{(j)} N_b'(x)} \end{bmatrix}, \tag{5.1}$$

where, w_b is the weight matrix of the last softmax layer with $w_b^{(p)}$ representing the weight vector that produces the output of the class p and $N_b'(x)$ is the output of the layer in N_b, immediately preceding the softmax layer. Once this network is trained, $\mathcal{S}_b$ broadcasts these models.

At $\mathcal{S}_i$, a new discriminative model N_i is initialized with the parameters of N_b. N_b is trained (and has the ability) to only make predictions on the label space of $\mathcal{D}_b$, i.e.

$\{1 \dots j\}$. The incremental learner model N_i therefore, cannot be initialized with the same weights in the softmax layer of N_b alone. Along with the weights for the first j classes, N_i should also be initialized with random parameters as necessary to allow for the prediction on a combined incremental label space of $\{1 \dots c\}$. We can simply do the following assignment to get the desired arrangement:

$$w_i^{(p)} = \begin{cases} w_b^{(p)}, & \text{if } \; p \in \{1 \dots j\} \\ \mathcal{N}(0,1), & \text{if } \; p \in \{j+1 \dots c\} \end{cases} . \tag{5.2}$$

Equation 5.2 describes a simple strategy where the weight vectors are carried over to the first j classes and random weight vectors are assigned to the rest of the $c - j$ classes. In figure 5.2, the gray weights in N_i represent those that are copied and the red weights represent the newly initialized weights.

We now have at $\mathcal{S}_i$, a network that will generate samples from the distribution of $P(x^b)$ and an initialized network N_i whose layers are setup with the weights from N_b. To train this network on $\mathcal{D}_i$, if we simply ignore G_b and train the network with samples $(x^i, y^i) \in \mathcal{D}_i$, we will run into the catastrophic forgetting problem as discussed in figure 5.1. To avoid this, we can use samples queried from G_b (such samples are notationally represented as $G_b(z)$ to indicate sampling using a random vector z) and use these samples to avoid forgetting. However we do not have targets for these samples to estimate an error with. Phantom sampling will help us to acquire targets.

Definition 1. *A phantom sampler is a process of the following form:*

$$\mathcal{P} : (z, T, N_b, G_b) \rightarrow \{G_b(z), P_{N_b}(y|G_b(z), T)\}. \tag{5.3}$$

where, $y \in \{0 \dots j\}$ and T is a temperature parameter which will be described below. Using N_b and G_b, we can use this sampling process to generate sets of sample-target pairs that simulate samples from the dataset $\mathcal{D}_b$. Simply using $P_{N_b}(y^b|x^b)$ is not

possible as we do not have access to x^b at $\mathcal{S}_i$, and $\mathcal{S}_i$ is not allowed to communicate with $\mathcal{S}_b$ regarding the data due to the data membrane condition described in property 1. We can however replace x^b with $G_b(z)$ and use the generated samples to produce targets from this network for the generated samples itself. This is justifiable since $G_b(z)$ is learnt to hallucinate samples from $P(x^b)$. However, given that we only use a simple GAN and that the samples are expected to be noisy, we might get corrupted and untrustworthy targets. GANs have not advanced sufficiently to a degree where perfect sampling is possible at the image level, at the moment of writing this article. As GAN technology improves, much better sampling could be achieved using this process.

Given that GANs (and any other similar generative models) are imperfect, often samples can have properties that are blended from two or more classes. In these cases, the targets generated from N_b might also be too high for only one of these classes, which is not optimal. To avoid this problem, we use a replacement for the softmax layer of N_b with a new temperature-raised softmax layer,

$$\begin{bmatrix} P_{N_b}(y=1|x^b,T) \\ \vdots \\ P_{N_b}(y=j|x^b,T) \end{bmatrix} = \frac{1}{\sum_{p=1}^{j} e^{\frac{w_b^{(p)} N_b'(x)}{T}}} \begin{bmatrix} e^{\frac{w_b^{(1)} N_b'(x)}{T}} \\ \vdots \\ e^{\frac{w_b^{(j)} N_b'(x)}{T}} \end{bmatrix}. \tag{5.4}$$

This temperature-raised softmax for $T > 1$ ($T = 1$ is simply the softmax described in equation 5.1) provides a softer target which is smoother across the labels. It reduces the probability of the most probable label and provides rewards for the second and third most probable labels also, by equalizing the distribution. Soft targets such as the one described and their use in producing ambiguous targets exemplifying the relationships between classes were proposed in Hinton *et al.* (2015). In this context, the use of soft targets for $G_b(z)$ helps us get appropriate labels for the samples that may be poorly generated. For instance, a generated sample could be in between

classes 8 and 0. The soft target for this will not be a strict 8 or a strict 0, but a smoother probability distribution over the two (all the) classes.

While learning N_i, with a batch of samples from D_i, we may simply use a negative log-likelihood with the softmax layer for the labels. To be able to back-propagate samples from phantom sampling, we require a temperature softmax layer at N_i as well. For this, we simply create a temperature softmax layer that share the weights w_i, of the softmax layer of N_i, just as we did for N_b. This implies that N_i will have $c - j + 1$ additional units for which we would not have targets as phantom sampling will only provide us with targets for the first j classes. Given that the samples themselves are hallucinated from $G_b(z)$, the optimal targets to assign for the output units $[j + 1 \dots c]$ of the temperature softmax layer are zero. Equivalently, we could simply avoid sharing the extra weights. Therefore along with the phantom sample's targets, we concatenate a zero vector of length $[j+1 \dots c]$. This way, we could simply back-propagate the errors for the phantom samples also. The error for data from $\mathcal{D}_i$ is,

$$e(w_i, x^i \in \mathcal{D}_i) = \mathcal{L}(y^i, \arg\max_y P_{N_i}(y|x^i)), \tag{5.5}$$

where, $\mathcal{L}$ represents an error function. The error for phantom samples is,

$$e(w_i, G_b(z)) = \mathcal{L}(P_{N_b}(y|G_b(z), T), P_{N_i}(y|G_b(z), T)). \tag{5.6}$$

Typically, we use a categorical-cross-entropy for learning labels and a root mean-squared error for learning soft-targets.

While both samples from $\mathcal{D}_i$ and from the phantom sampler are fed-forward through the same network, the weights are updated for two different errors. If the samples come from the phantom sampler, we estimate the error from the temperature softmax layer and if the samples come from $\mathcal{D}_i$, we estimate the errors from the soft-max layer. For every k iterations of $\mathcal{D}_b$, we train with 1 iteration of phantom samples

$G(z)$. k is decided based on the number of classes that are in each set $\mathcal{D}_b$ and $\mathcal{D}_i$.

Thus far we have assumed a certain chronology of events where $\mathcal{S}_i$ begins training only after $\mathcal{S}_b$ is finished training. We could generalize this strategy of using phantom sampling when $\mathcal{S}_i$ is already, partially trained by the time $\mathcal{S}_b$ finishes and triggers the incremental learning. In this case, we will not be able to re-initialize the network N_i with new weights, but as long as we have phantom samples, we can use a technique similar to mentor nets or fitnets, using embeded losses between N_b and N_i and transfer knowledge about D_b to N_i Romero *et al.* (2014) Venkatesan and Li (2016). This strategy could also be extended to more than one increment of data in a straightforward manner. Using the same phantom sampling technique we could continue training the GAN to update it with the distributions of the new classes. Once trained, we can pass on this GAN and the newly trained net N_i to the next incremental site.

5.3 Related Work

Catastrophic Forgetting: Early works by McCloskey, French and Robins outlines this issue McCloskey and Cohen (1989); French (1993); Robins (1995). In recent years, this problem has been tackled using special activation functions and dropout regularization. Srivastava et al. demonstrated that the choice of activation function affects catastrophic forgetting and introduced the *Hard Winner Take All* (HWTA) activation Srivastava *et al.* (2013). Goodfellow et al. argued that increased dropout works better at minimizing catastrophic forgetting compared to activation functions Goodfellow *et al.* (2013a). All these studies were made in regards to unavailability of data for particular classes, rather than in terms of incremental learning.

We find that most previous works in incremental learning, relaxes or violates the rigorous constraints that we have proposed for an incremental learner. While this may satisfy certain case studies, pertaining to each article, we find no work that has

addressed our definition sufficiently. In this section, we organize our survey of existing literature in terms of the conditions they violate.

Relaxing the data membrane: The following approaches relax property (1) to varying degrees. Mensink et al. develop a metric learning method to estimate the similarity (distance) between test samples and the *nearest class mean* (NCM) Mensink *et al.* (2012, 2013). The class mean vectors represent the centers of data samples belonging to different classes. The learned model is a collection class center vectors and a metric for distance measurement that is determined using the training data. The NCM approach has also been successfully applied to random forest based models for incremental learning in Ristin *et al.* (2014). The nodes and leaves of the trees in the NCM forest are dynamically grown and updated when trained with data from new classes. A tree of deep convolutional networks (DCNN) for incremental learning was proposed by Xiao et al. Xiao *et al.* (2014). The leaves of this tree are CNNs with a subset of class outputs and the nodes of the tree are CNNs which split the classes. With the input of new data and classes, the DCNN grows hierarchically to accommodate the new classes. The clustering of classes, branching and tree growth is guided by an error-driven preview process and their results indicate that the incremental learning strategy performs better than a network trained from scratch.

The Learn++ is an ensemble based approach for incremental learning Polikar *et al.* (2001) Muhlbaier *et al.* (2009). Based on the Adaboost, the algorithm weights the samples to achieve incremental learning. The procedure, however requires every data batch to have examples from all the previously seen classes. In Kuzborskij *et al.* (2013), Kuzborskij et al. develop a least squares SVM approach to incrementally update a N-category classifier to recognize N+1 classes. The results indicate that the model performs well only when the N+1 classifier model is also trained with some data samples from the previous N classes.

iCaRL is an incremental representation based learning method by Rebuffi et al. Rebuffi *et al.* (2017). It progressively learns to recognize classes from a stream of labeled data with a limited budget for storing exemplars. The iCaRL classification is based on the nearest-mean-of-exemplars. The number of exemplars for each class is determined by a budget and the best representation for the exemplars is updated with existing exemplars and newly input data. The exemplars are chosen based on a herding mechanism that creates a representative set of samples based on a distribution Welling (2009). This method while being very successful, violates the membrane property by transferring well-chosen exemplar samples. In our results section we address this idea by demonstrating that significant amount of (randomly chosen) samples are required to out-perform our strategy, which violates the *budget* criteria of the iCaRL methods.

Relaxing data agnosticism: Incremental learning procedures that draw inference regarding previously trained data based on current batch of training data, can be viewed as violating this constraint. Li et al. use the base classifier N_b to estimate the conditional probabilities $P(\hat{y}|x)$ for $x : (x, y) \in \mathcal{D}_i$. When training N_i with $\mathcal{D}_i$, they use these conditional probabilities to guide the output probabilities for classes $y \in [1, \ldots, j]$ Li and Hoiem (2016). In essence, the procedure assumes that if N_i is trained in such a manner that $P(\hat{y}|x)$ for $x : (x, y) \in \mathcal{D}_i$ is the same for both classifier N_b and N_i, this ensures that $P(\hat{y}|x)$ for $x : (x, y) \in \mathcal{D}_b$ will also be the same. This is a strong assumption relating $\mathcal{D}_b$ and $\mathcal{D}_i$ violating agnosticism. The authors Furlanello et al. develop a closely related procedure to in Furlanello *et al.* (2016). They train neural networks for the incremental classifier N_i by making sure the conditional probabilities $P(\hat{y}|x)$ for $x : (x, y) \in \mathcal{D}_i$ is the same for both N_b and N_i. The only difference compared to Li and Hoiem (2016) is in the regularization of network parameters using weight decay and the network initialization. In another procedure based on the

same principles, Jung et al. constrain the feature representations for $\mathcal{D}_i$ to be similar to the feature representations for $\mathcal{D}_b$ Jung *et al.* (2016).

Other models assume that the parameters of the classifiers w_b for N_b and w_i for N_i are related. Kirkpatrick et al. model the probability $P(w_b|\mathcal{D}_b)$ and get an estimate for the important parameters in w_b Kirkpatrick *et al.* (2017). When training N_i initialized with parameters w_b, they make sure not to offset the important parameters in w_b. This compromises the training of N_i under the assumption that important parameters in w_b for $\mathcal{D}_b$ are not important for $\mathcal{D}_i$.

Closely related to the previous idea is *pseudo-rehearsal* proposed by Robins in 1995 Robins (1995). Neuro-biological underpinnings of this work was also studied by French et. al, French (1997). This method is a special case of ours if, the GAN was untrained and produces random samples. In other words, they used N_b to produce targets for random samples $G_b(z) = z \rightarrow \mathcal{N}(0, 1)$, instead of using a generative model, similar to phantom sampling. This might partly be due to the fact that sophisticated generative models were not available at the time. This article also does not use soft targets such as those that we use because, for samples that are generated randomly, $T = 1$ is a better target. This article does not violate any of the properties that we required for our uncompromising incremental learner.

5.4 Experiments and Results

To demonstrate the effectiveness of our strategy we conduct thorough experiments using three benchmark datasets: MNIST dataset of handwritten character recognition, Street view housing numbers (SVHN) dataset and the CIFAR10 10-class visual object categorization dataset LeCun *et al.* (1998b); Netzer *et al.* (2011); Krizhevsky

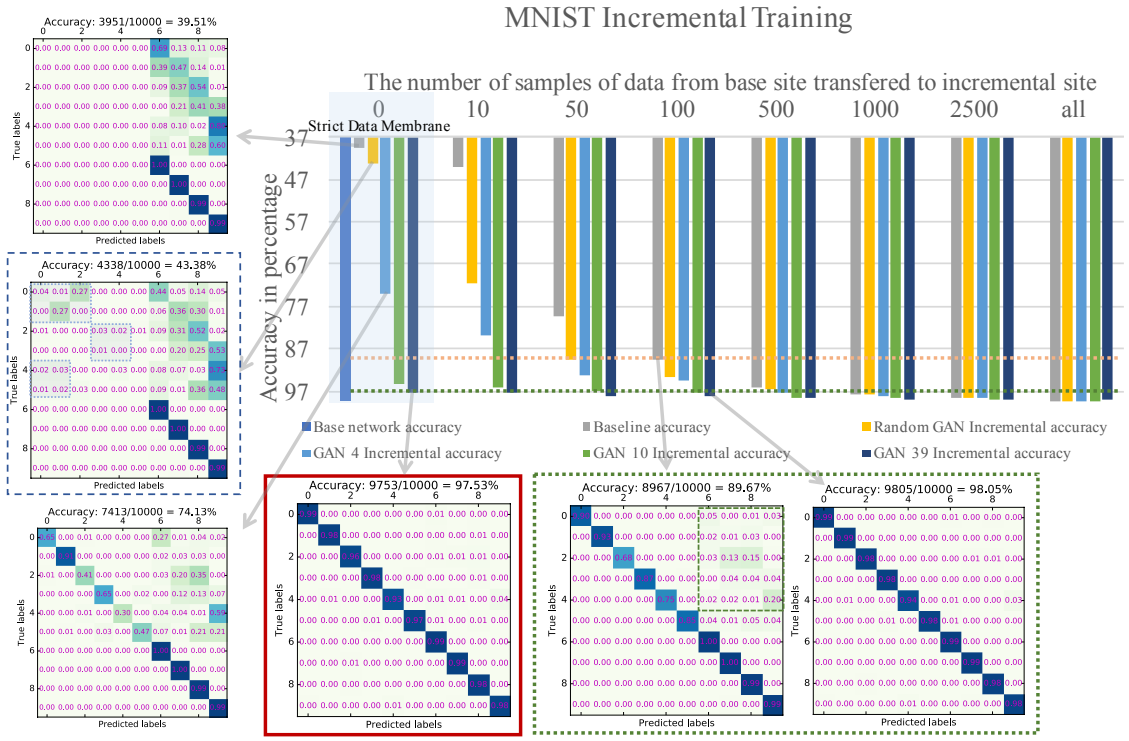


Figure 5.3: Results for the MNIST Dataset.

and Hinton (2009). In all our experiments [1] we train the $\mathcal{S}_b$'s GAN, G_b and base networks N_b independently. The network parameters of all these models are written to drive, which simulates broadcasting the networks. Once trained, the datasets that are used to train and test these methods are deleted, simulating the data membrane and the processes are killed.

We then begin $\mathcal{S}_i$ as an independent process in keeping with site independence. This uses a new dataset which is setup in accordance with property 1. Networks G_b and N_b's parameters are loaded but only in their feed-forward operations. Two identical copies of networks N_i^σ and N_i^T that share weights are built. These are initialized with the parameters of N_b, $N_i^\sigma$with without temperate and N_i^T with temperature

[1] Our implementations are in theano and our code is available at `https://github.com/ragavvenkatesan/Incremental-GAN`.

softmax layers. By virtue of the way they are setup, updating the weights on one, updates both the networks. We feed forward k mini batches of data from $\mathcal{D}_i$ through the column that connects to the softmax layer and use the error generated here to update the weights for each mini batch. For every k updates of weights from the data, we update one mini batch of phantom samples from $(G_b(z), P_{N_b}(y|G_b(z), T))$. This is run until early termination or until a pre-determined number of epochs. Since we save the parameters of G_b after every epoch, we can load the corresponding GAN for our experiments. We use the same learning rate schedules, optimizers and momentums across all the architectures. We fix our temperature values using a simple grid search. We conducted several experiments using the above protocol to demonstrate the effectiveness of our strategy. The following sections discuss these experiments.

5.4.1 Single Dataset Experiments

MNIST: For the MNIST dataset, we used a GAN G_b that samples 10 image generations from a uniform 0-mean Gaussian. The generator part of the network has three fully-connected layers of 1200, 1200 and 784 neurons with ReLU activations for the first two and tanh activation for the last layers, respectively Nair and Hinton (2010). The discriminator part of G_b has two layers of 240 maxout-by-5 neurons Goodfellow *et al.* (2013b). This architecture that mimics the one used by Goodfellow et. al, closely Goodfellow *et al.* (2014). All our discriminator networks across both sites $\mathcal{S}_b$ and $\mathcal{S}_i$ are the same architecture which for the MNIST dataset is, two convolutional layers of 20 and 50 neurons each with filter sizes of 5×5 and 3×3 respectively, with max pooling by 2 on both layers. These are followed by two full-connected layers of 800 neurons each. All the layers in the discriminators are trained with batch normalization and weight decay with the fully-connected layers trained with a dropout of 0.5 Srivastava *et al.* (2014); Ioffe and Szegedy (2015).

Results of the MNIST dataset are discussed in figure 5.3. The bar graph is divided into many factions $p = [0, 10, \ldots \text{all}]$, each representing the performance having p samples per class transmitted between $\mathcal{S}_b$ to $\mathcal{S}_i$. Within each faction are five bars, except $p = 0$ that has six bars. The first bar at $p = 0$ represents the state-of-the-art accuracy with the (base) network trained on the entire dataset ($\mathcal{D}_b \cup \mathcal{D}_i$, for the given hypothesis. This is the upper-bound on the accuracies, given the architecture. The first bar on the left (second for $p = 0$) represents the accuracy of a *baseline* network that is learnt without using our strategy. A baseline network does not use a phantom sampler and is therefore prone to catastrophic forgetting. The other four bars represent the performance of networks learnt using our strategy. From left to right, the G_b for each network is trained for $e = [0, 4, 10, 39]$ epochs, respectively. Confusion matrices are shown wherever appropriate.

The central result of this experiment is the block of accuracies highlighted within the blue-shaded box ($p = 0$), which show the performances while maintaining a strict data membrane. The confusion matrix in the top-left corner shows the performance of the base network with $p = 0$, which is similar to (c) from figure 5.1, demonstrating catastrophic forgetting. The next confusion matrix that is marked with blue dashed line depicts the accuracy of N_i with G_b producing random noise. This setup is the same as in the work by Robins Robins (1995). It can be observed that even when using a phantom sampler that samples pure noise, we achieve a noticeable boost in recognition performance, significantly limiting catastrophic forgetting. The confusion matrix in the bottom-left corner is the performance using G_b trained for only 4 epochs. This shows that even with a poorly trained GAN, we achieve a marked increase in performance. The best result of this faction is the confusion matrix highlighted in the red square. This is the result of a network learnt with phantom sampling with a GAN G_b that is trained closest to convergence at 39 epochs. It can be clearly noticed

that the phantom sampling strategy helps in avoiding catastrophic forgetting, going so far as to achieve nearly state-of-the-art base accuracy.

The rest of the factions in this experiment make a strong case against the relaxation of the data membrane. Consider, for instance, the pair of confusion matrices at the bottom right, highlighted within the green dotted lines. These represent the performance of *baseline* and $e = 39$ networks, when $p = 100$ samples per-class were transmitted through the membrane. A *baseline* network that was trained carefully without overfitting produced an accuracy of 89.67% and still retained a lot of confusion (shown in green dashed lines within the confusion matrix). The network trained with phantom sampling significantly outperforms this. In fact (refer the orange dotted line among the bars), this relaxation is outperformed by a phantom sampling trained network even with a poorly trained GAN (with just 10 epochs) while adhering to a strict data membrane ($p = 0$). It is only when $p = 1000$ samples per-class (which is 20%) of the data are being transferred, does the baseline even match the phantom sampling network with $p = 0$ (as demonstrated by the blue dotted line among the bars). All these results conclusively demonstrate the significance of phantom sampling and demonstrate the nonnecessity of the relaxation of the data membrane. An uncompromising incremental learner was thereby achieved using our strategy.

SVHN and CIFAR 10: For both these datasets we used a generator model that generates images from 64 Gaussian random variables. The number of neurons in subsequent fully-connected layers are 1200 and 5408 respectively. This is followed by two fractionally-strided or transposed convolution layers with filter sizes 3×3 and 5×5 respectively. Apart from the last layer that generates the 32×32 image, every layer has a ReLU activation. The last layer uses a tanh activation. Our discriminator networks including the discriminator part of the GANs have six convolutional layers with neurons $20, 50, 50, 100, 100$ and 250 respectively. Except the first layer, which

has a filter size of 5×5, every layer has filter sizes of 3×3. Every third layer maxpools by 2. These are followed by two fully-connected layers of 1024 nodes each. All activations are ReLU.

Results of the CIFAR 10 dataset are shown in figure 5.4 and that of SVHN are shown in figure 5.5. CIFAR 10 and SVHN contain three channel full-color images that are sophisticated. GANs, as originally proposed by Goodfellow et. al, fail to generate reasonably good looking samples for these datasets Goodfellow *et al.* (2014). Since we used the same models, the results shown here could be improved significantly with the invention (or adoption) of better generative models.

We can observe from figures 5.4 and 5.5, that they follow patterns similar to the MNIST results in figure 5.3. The CIFAR-10 results clearly demonstrate that only after about 20% of data is transmitted, do the performance come close to matching the phantom sampler approach. In the SVHN results shown in figure 5.5, we can observe the marked difference in performance with only few samples being transmitted. Because SVHN is a large dataset in number of samples, the GANs were able to generate sufficiently good images that lead to superior performance. This result shows us the advantage our strategy has when working with big datasets. Firstly, having a big dataset imposes additional penalties for requiring to transmit data and therefore should be avoided. Secondly, having more number of samples implies that a simple GAN could generate potentially good looking images, helping us maintain consistent performance throughout.

5.4.2 Cross-Domain Increments

It could be argued that performing incremental learning within the same dataset has some advantages in terms of the domain of the datasets being similar. The similarity in domains could imply that the datasets are general and therefore, the base

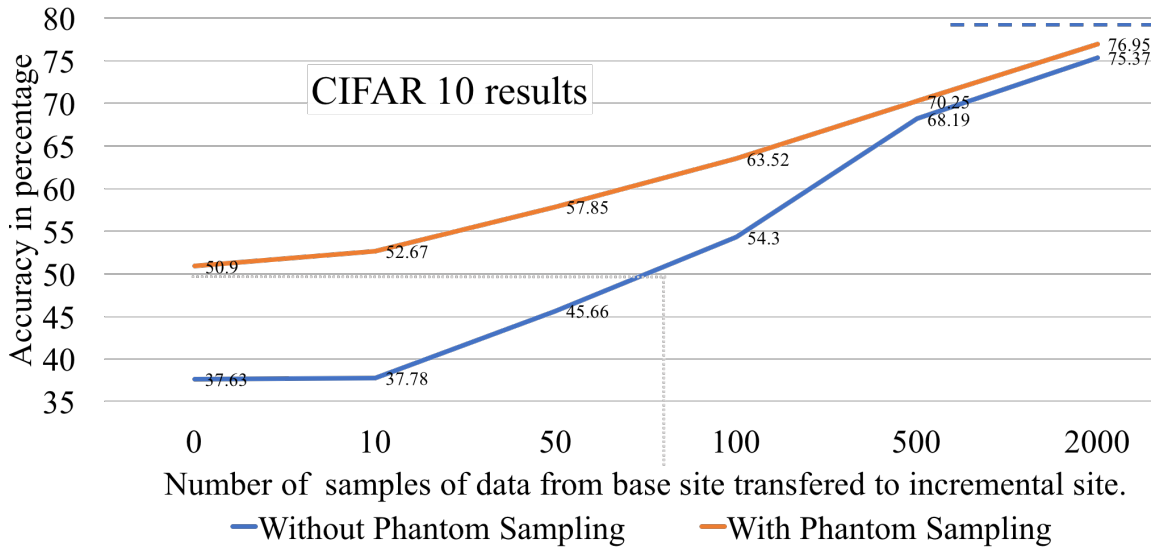


Figure 5.4: Results for the CIFAR10 dataset. The dotted line is the baseline accuracy learnt by a network with access to all data. This is a proxy for the expected best achievable results by this architecture.

network already has some features of the incremental dataset encoded in it Venkatesan *et al.* (2016a). In this section we demonstrate two special cross-domain cases. In the first case, the incremental data $\mathcal{D}_i$, while sampled from a new domain, has the same label space as $\mathcal{D}_b$. In the second case, $\mathcal{D}_i$ has new classes that are not seen in $\mathcal{D}_b$.

Case 1: In this experiment, our base dataset $\mathcal{D}_b$ is the MNIST-rotated dataset developed by Larochelle et al. Larochelle *et al.* (2007). This is used to learn G_b and N_b. This is a dataset that is the same as the MNIST dataset, but the samples are randomly rotated. The incremental data comes from the MNIST dataset . The incremental data and the base dataset has the same label space. The domain of incremental dataset $\mathcal{D}_i$ (MNIST) can be considered as a special subset of the domain of $\mathcal{D}_b$ (MNIST-rotated). Therefore, this setup is ripe for a scenario where the incremental site forgets the expanse of the domain of the base site. The network architecture remains the same as for the MNIST experiments. The results for this experiment are shown in figure 5.6. It can be clearly noted that there is about 20% difference in performance

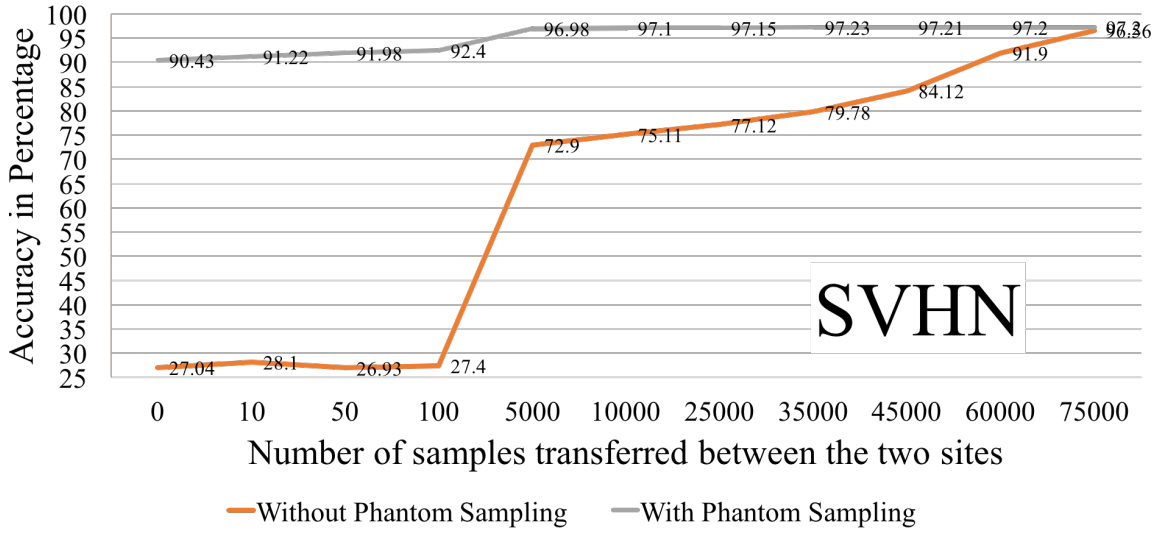


Figure 5.5: Results for the SVHN results. These show a much stronger trend because of the amount of data involved being larger and that GANs have produced better images than for CIFAR 10.

using our strategy.

Case 2: In this experiment, our base dataset $\mathcal{D}_b$ is the MNIST dataset and it is used to learn G_b and N_b. The incremental dataset $\mathcal{D}_i$ is SVHN. The classes of SVHN are labelled $10-19$ at $\mathcal{S}_i$ and the labels of MNIST are maintained as $0-9$. This is essentially incrementing on a new task from a disjoint domain. The results of this experiment are shown in figure 5.7. It can be clearly noted that there is about 20% increase in performance using our strategy.

5.5 Extension to Bounded-Continual Learning

So far we have defined and studied incremental learning. Incremental learning consists of a single increment. In this section, we extend this idea to bounded-continual learning. Continual learning is incremental learning with multiple increments. Bounded-continual learning is a special case of continual learning, where the number of increments is limited. Life-long learning for instance, is an example of unbounded-continual learning.

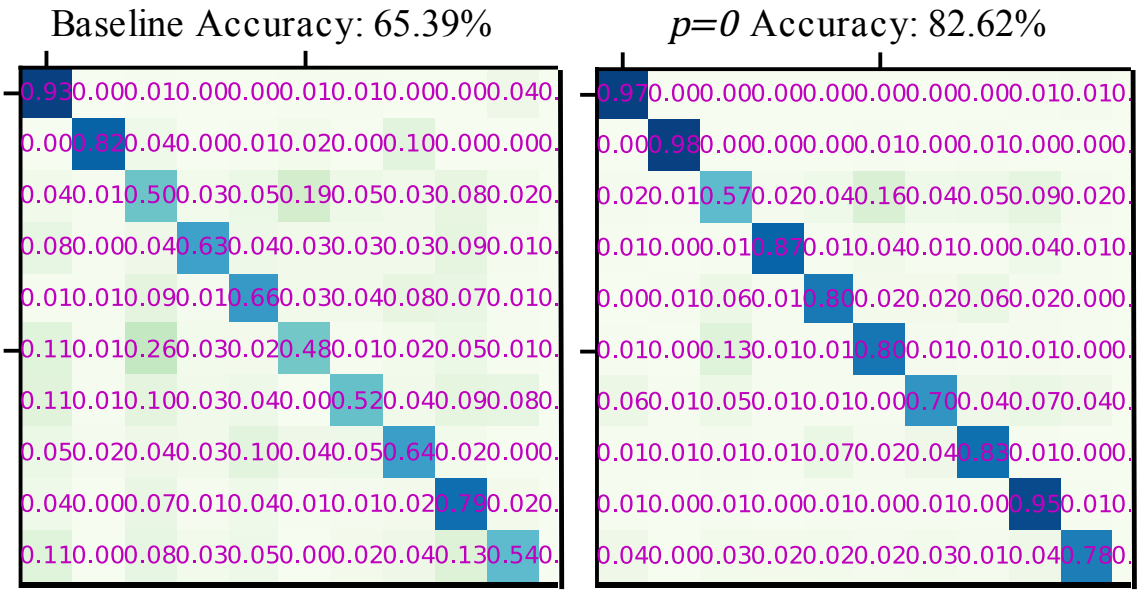


Figure 5.6: Results for MNIST-rotated trained at $\mathcal{S}_b$ and incremented with new data from the MNIST original dataset at $\mathcal{S}_i$. The class labels for both these datasets is $[0, \ldots 9]$. The confusion matrix on the left is for the baseline network and the one on the right is for our strategy with $p = 0$.

The proposed strategy can be trivially modified to work for multiple increments. Consider there are s sites. Consider also that we have one base network N_b^i, with i indicating its state after the increment i. We learn for every increment i, a new GAN G_i. We use the set of GANs $\{G_0, \ldots G_{i-1}\}$ to create i phantom samplers, one for each increment.

Continual learning can be implemented in the following manner. At the beginning, we construct a base network N_b^0. Once N_b^0 is trained with $\mathcal{D}_0$, we create a copy (P^0) of N_b^0 for phantom labelling. The samples generated by G_0 are fed through P^0, to get phantom samples for the increment $i = 0$. This phantom sampler will be used when learning the increment $i = 1$.

On receiving the data increment $\mathcal{D}_i$, we have i GANs $G_0, \ldots G_{i-1}$. We can create an updated copy of the phantom sampler P^{i-1}, by making a copy of N_b^{i-1}. We create a phantom sampler, where P^{i-1} samples from all the GANs uniformly and hallucinates

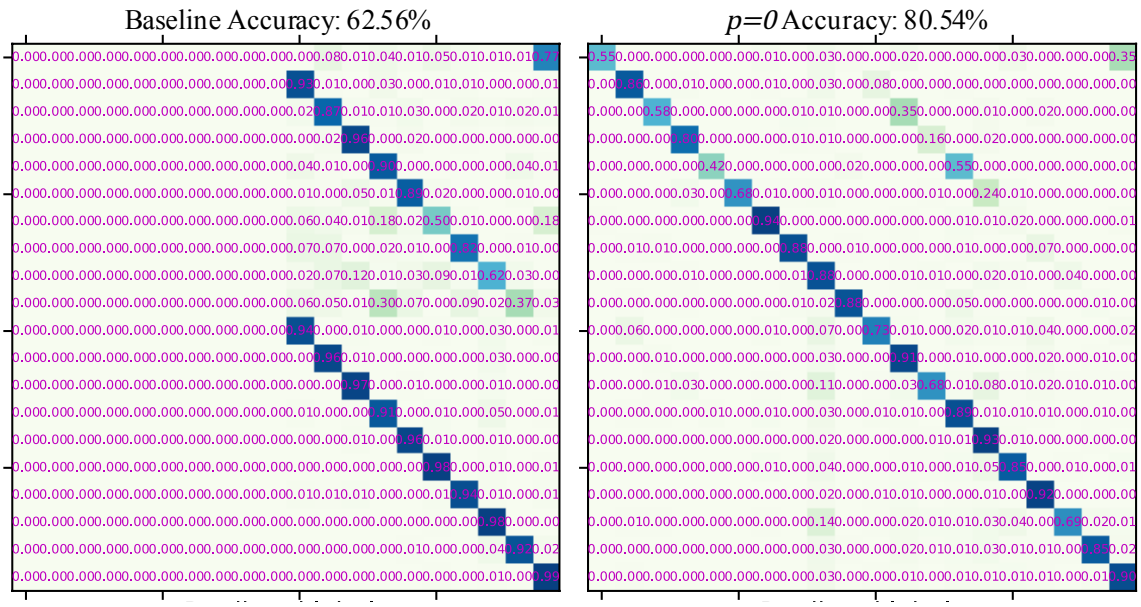


Figure 5.7: Results for MNIST trained at $\mathcal{S}_b$ and incremented with new data from the SVHN dataset at $\mathcal{S}_i$. The SVHN classes are considered as novel classes in this experiment, therefore we have twenty classes. The confusion matrix on the left is for the baseline network and the one on the right is for our strategy with $p = 0$.

the labels. We update N_b^{i-1} to N_b^{i+1}, by training it on $\mathcal{D}_i$ along with this new phantom sampler P^{i-1}.

This approach of bounded-continual learning is apt in cases where the data at each increment is large enough to warrant training a GAN. While, this approach works well for bounded-continual learning systems, it is not scalable to lifelong learning. This is because unbounded-continual learning could result in an infinite number of GANs. Seff et al, recently proposed an idea to update the same GAN for a large number of increments Seff *et al.* (2017). Such a GAN could generate data from the combined distributions of all increments it has seen. While this still works only on a bounded number of increments, this is a step towards unbounded-continual learning. If we employ this idea in our system, we could eliminate the need for having multiple GANs and extend our strategy trivially to life-long learning as well. This idea is still in its infancy and is not fully mature yet. Although we have drawn a road map, we

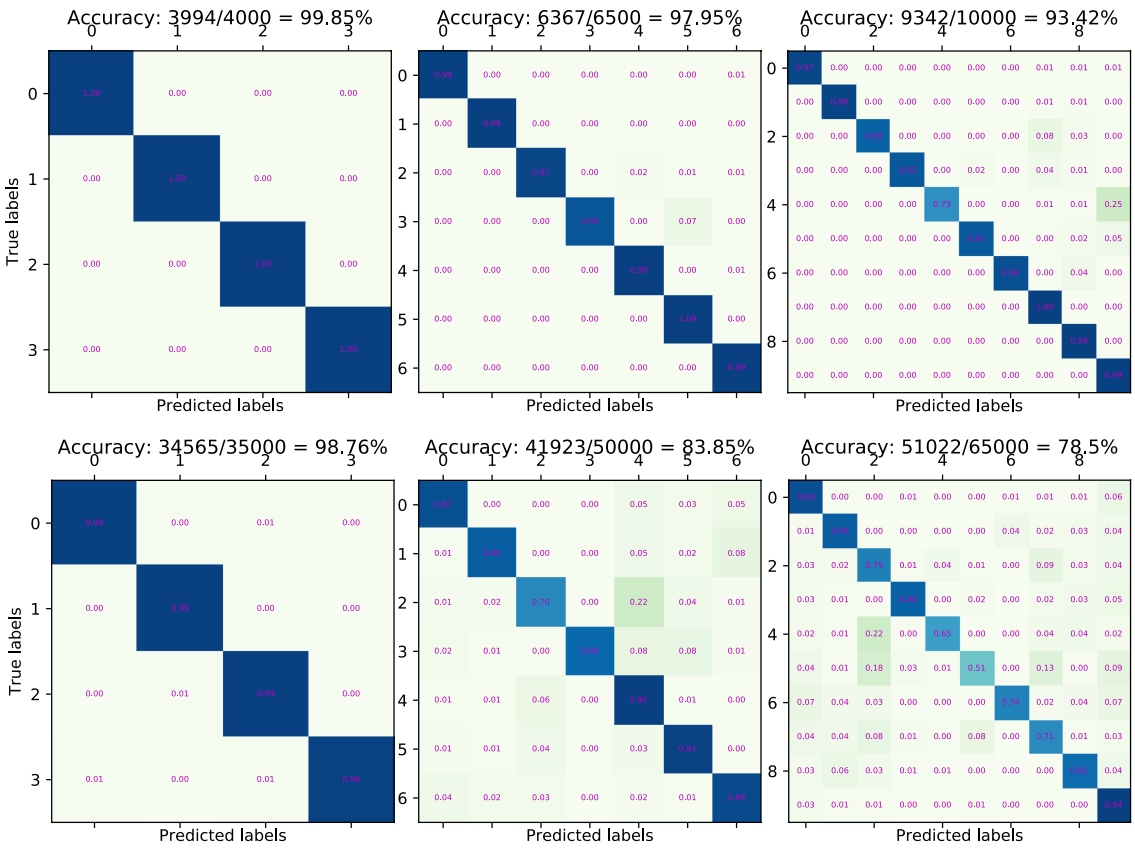


Figure 5.8: Results for the bounded-continual learning experiments. There are two steps of increment. Each increment has its own GAN. The top row is MNIST and the bottom row is SVHN. In each row, the image on the left is the confusion of the base net N_0 with classes $[0, 1, 2, 3]$. The center image is the confusion for the first increment with training data in classes $[4, 5, 6]$ and testing data in classes $[0, \ldots 6]$. The confusion on the right is the final increment with training data from classes $[7, 8, 9]$ and testing data from the classes $[0, \ldots 9]$.

await further development of this idea to incorporate it fully into our strategy.

5.5.1 Experiments and Results

We use GANs and classifier architectures which are the same as defined for MNIST and SVHN in the previous section, respectively. We demonstrate continual learning on both datasets by performing two increments. The base dataset contains the classes $[0, 1, 2, 3]$, the first increment contains classes $[4, 5, 6]$ and the last increment contains

[7, 8, 9]. Figure 5.8 shows the results for continual learning for both datasets. It can be easily noticed that we can achieve close to state-of-the-art accuracy even while performing continual learning. A note of prominence is that even at the end of the third increment, there is little confusion remaining from the first increment. This demonstrates strong support for our strategy even when extending to continual learning.

5.6 Conclusions

In this chapter, we redefined the problem of incremental learning, in its most rigorous form so that it can be a more realistic model for important real-world applications. Using a novel sampling technique involving generative models and the distillation technique, we implemented a strategy to hallucinate samples with appropriate targets using models that were previously trained and broadcast. Without having access to historic data, we demonstrated that we could still implement an uncompromising incremental learning system without relaxing any of the constraints of our definitions. We show strong and conclusive results on three benchmark datasets in support of our strategy.

Chapter 6

CONCLUSIONS

In this dissertation several novel approaches to representation learning and task learning were studied. The study was divided into two parts:

1. Non-parametric multiple-instance learning and
2. Representation learning

A novel non-parametric multiple-instance learning technique was devised using modified Parzen window and k-NN ideas. These ideas were shown to our perform other existing approaches. This solution wass applied to a diabetic retinopathy pathology detection problem effectively. Several theoretical properties of this approach were also studied.

In representation learning, generality of neural features were investigated first. This investigation yielded some surprising results among the relationships of features that were learnt.

The possibility of learning from a mentor network instead of from labels were then investigated. One way to train a mentee network without the presence of labels is from distillation. Distillation of dark knowledge was used to efficiently mentor a small network from a pre-trained large mentor network, both with and without labels.

The next study involved incremental learning. Phantom-sampling from a generative model helps us in avoiding catastrophic forgetting in incremental learning setup. In the context of network compression, batch-decorrelation and other correlation effects of neuron activities are studied. These studies help us understand representation learning with smaller and compressed networks.

Amores, J., "Multiple instance classification: Review, taxonomy and comparative study", Artificial Intelligence **201**, 81–105 (2013). 23

Andrews, S., I. Tsochantaridis and T. Hofmann, "Support vector machines for multiple-instance learning", Advances in neural information processing systems **15**, 561–568 (2002). 22, 26, 35, 36, 37, 38, 40

Antić, B. and B. Ommer, "Robust multiple-instance learning with superbags", in "Computer Vision–ACCV 2012", pp. 242–255 (Springer, 2013). 22, 27, 36, 37

Ba, J. and R. Caruana, "Do deep nets really need to be deep?", in "Advances in neural information processing systems", pp. 2654–2662 (2014). 74

Babenko, B., M. Yang and S. Belongie, "Robust object tracking with online multiple instance learning", IEEE PAMI **33**, 8, 1619–1632 (2011). 21

Balan, A. K., V. Rathod, K. P. Murphy and M. Welling, "Bayesian dark knowledge", in "Advances in Neural Information Processing Systems", pp. 3420–3428 (2015). 72, 75

Bastien, F., P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard and Y. Bengio, "Theano: new features and speed improvements", Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012). 84

Belkin, M. and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering", in "Advances in neural information processing systems", pp. 585–591 (2002). 5

Bengio, Y., P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", IEEE transactions on neural networks **5**, 2, 157–166 (1994). 20

Bergeron, C., G. Moore, J. Zaretzki, C. M. Breneman and K. P. Bennett, "Fast bundle algorithm for multiple-instance learning", Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**, 6, 1068–1079 (2012). 22, 27, 36, 37

Boiman, O., E. Shechtman and M. Irani, "In defense of nearest-neighbor based image classification", in "Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on", pp. 1–8 (IEEE, 2008). 27

Bucilu, C., R. Caruana and A. Niculescu-Mizil, "Model compression", in "Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 535–541 (ACM, 2006). 74

Candes, E. J., J. K. Romberg and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements", Communications on pure and applied mathematics **59**, 8, 1207–1223 (2006). 5

Chan, W., N. R. Ke and I. Lane, "Transferring knowledge from a rnn to a dnn", arXiv preprint arXiv:1504.01483 (2015). 72, 75

Chandakkar, P. S., R. Venkatesan and B. Li, "Retrieving clinically relevant diabetic retinopathy images using a multi-class multiple-instance framework", in "SPIE Medical Imaging", pp. 86700Q–86700Q (International Society for Optics and Photonics, 2013a). 4, 39

Chandakkar, P. S., R. Venkatesan and B. Li, "Mirank-knn: Multiple instance retrieval of clinically-relevant diabetic retinopathy images", SPIE Journal of Medical Imaging (2017). 4, 12

Chandakkar, P. S., R. Venkatesan, B. Li and H. Li, "Retrieving clinically relevant diabetic retinopathy images using a multi-class multiple-instance framework", in "SPIE Medical Imaging", pp. 86700Q–86700Q (International Society for Optics and Photonics, 2013b). 12

Chandakkar, P. S., R. Venkatesan, B. Li and H. K. Li, "A machine-learning approach to retrieving diabetic retinopathy images", in "Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine", pp. 588–589 (ACM, 2012). 4

Chen, Y., J. Bi and J. Z. Wang, "Miles: Multiple-instance learning via embedded instance selection", Pattern Analysis and Machine Intelligence, IEEE Transactions on **28**, 12, 1931–1947 (2006). 22, 26, 28, 36, 37, 38, 39, 40

Chen, Y. and J. Wang, "Image categorization by learning and reasoning with regions", The Journal of Machine Learning Research **5**, 913–939 (2004). 21, 22, 25, 26, 33, 36, 38

Dalal, N. and B. Triggs, "Histograms of oriented gradients for human detection", in "Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on", vol. 1, pp. 886–893 (IEEE, 2005a). 3

Dalal, N. and B. Triggs, "Histograms of oriented gradients for human detection", in "Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on", vol. 1, pp. 886–893 (IEEE, 2005b). 11

Dauphin, Y. N., H. de Vries, J. Chung and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization", arXiv preprint arXiv:1502.04390 (2015). 60, 84

de Campos, T. E., B. R. Babu and M. Varma, "Character recognition in natural images", in "Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal", (2009). 59, 67

Dietterich, T., R. Lathrop and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles", Artificial Intelligence **89**, 1, 31–71 (1997). 22, 24, 35

Escorcia, V., J. C. Niebles and B. Ghanem, "On the relationship between visual attributes and convolutional networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1256–1264 (2015). 49

et al., T., "Diabretdb0: Evaluation database and methodology for diabetic retinopathy algorithms.", (2005). 38

Everingham, M., L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge", International journal of computer vision **88**, 2, 303–338 (2010). 52, 71

Fei-Fei, L., R. Fergus and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories", Computer Vision and Image Understanding **106**, 1, 59–70 (2007). 59, 67

Felzenszwalb, P. F., R. B. Girshick, D. McAllester and D. Ramanan, "Object detection with discriminatively trained part-based models", Pattern Analysis and Machine Intelligence, IEEE Transactions on **32**, 9, 1627–1645 (2010). 21, 26

French, R. M., "Catastrophic interference in connectionist networks: Can it be predicted, can it be prevented?", in "Proceedings of the 6th International Conference on Neural Information Processing Systems", pp. 1176–1177 (Morgan Kaufmann Publishers Inc., 1993). 100

French, R. M., "Pseudo-recurrent connectionist networks: An approach to the'sensitivity-stability'dilemma", Connection Science **9**, 4, 353–380 (1997). 103

Fu, Z., A. Robles-Kelly and J. Zhou, "Milis: Multiple instance learning with instance selection", Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**, 5, 958–977 (2011). 26, 36, 38

Fukushima, K. and N. Wake, "Handwritten alphanumeric character recognition by the neocognitron", Neural Networks, IEEE Transactions on **2**, 3, 355–365 (1991). 49

Furlanello, T., J. Zhao, A. M. Saxe, L. Itti and B. S. Tjan, "Active long term memory networks", arXiv preprint arXiv:1606.02355 (2016). 102

Girshick, R., J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in "Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on", pp. 580–587 (IEEE, 2014). 52, 71

Goldbaum, M., N. Katz, S. Chaudhuri and M. Nelson, "Image understanding for automated retinal diagnosis", in "Proceedings of the Annual Symposium on Computer Application in Medical Care", p. 756 (American Medical Informatics Association, 1989). 38

Gonzalez, R. C. and R. E. Woods, "Digital image processing prentice hall", Upper Saddle River, NJ (2002). 8

Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning* (MIT Press, 2016), `http://www.deeplearningbook.org`. 13

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets", in "Advances in neural information processing systems", pp. 2672–2680 (2014). 96, 105, 108

Goodfellow, I. J., M. Mirza, D. Xiao, A. Courville and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks", arXiv preprint arXiv:1312.6211 (2013a). 100

Goodfellow, I. J., D. Warde-Farley, M. Mirza, A. C. Courville and Y. Bengio, "Maxout networks.", ICML (3) **28**, 1319–1327 (2013b). 105

Green, S., S. I. Wang, D. M. Cer and C. D. Manning, "Fast and adaptive online training of feature-rich translation models.", in "ACL (1)", pp. 311–321 (2013). 84

Griffin, G., A. Holub and P. Perona, "Caltech-256 object category dataset", (2007). 81

Guillaumin, M., J. Verbeek and C. Schmid, "Multiple instance metric learning from automatically labeled bags of faces", Computer Vision–ECCV pp. 634–647 (2010). 27

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 770–778 (2016). 20

Hinton, G., O. Vinyals and J. Dean, "Dark knowledge", Presented as the keynote in BayLearn (2014). 72, 74

Hinton, G., O. Vinyals and J. Dean, "Distilling the knowledge in a neural network", arXiv preprint arXiv:1503.02531 (2015). 7, 54, 94, 98

Huang, J., S. Kumar, M. Mitra, W. Zhu and R. Zabih, "Image indexing using color correlograms", in "Computer Vision and Pattern Recognition, IEEE Computer Society Conference on", pp. 762–768 (IEEE, 1997). 38

Ioffe, S. and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", arXiv preprint arXiv:1502.03167 (2015). 56, 60, 72, 83, 105

Jain, A. K., *Fundamentals of digital image processing* (Prentice-Hall, Inc., 1989). 8

Jantke, P., "Types of incremental learning", in "AAAI Symposium on Training Issues in Incremental Learning", pp. 23–25 (1993). 90

Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", in "Proceedings of the ACM International Conference on Multimedia", pp. 675–678 (ACM, 2014). 52, 71

Jung, H., J. Ju, M. Jung and J. Kim, "Less-forgetting learning in deep neural networks", arXiv preprint arXiv:1607.00122 (2016). 103

Kauppi, T., V. Kalesnykiene, J. Kamarainen, L. Lensu, I. Sorri, A. Raninen, R. Voutilainen, H. Uusitalo, H. Kälviäinen and J. Pietilä, "Diaretdb1 diabetic retinopathy database and evaluation protocol", Proc. Medical Image Understanding and Analysis (MIUA) pp. 61–65 (2007). 38

Kirkpatrick, J., R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran and R. Hadsell, "Overcoming catastrophic forgetting in neural networks", Proceedings of the National Academy of Sciences URL `http://www.pnas.org/content/early/2017/03/13/1611835114.abstract` (2017). 103

Krizhevsky, A. and G. Hinton, "Learning multiple layers of features from tiny images", (2009). 52, 59, 67, 81, 103

Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in "Advances in neural information processing systems", pp. 1097–1105 (2012). 19, 49

Kulkarni, N. and B. Li, "Discriminative affine sparse codes for image classification", in "Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on", pp. 1609–1616 (IEEE, 2011). 5

Kuzborskij, I., F. Orabona and B. Caputo, "From n to n+1: Multiclass transfer incremental learning", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 3358–3365 (2013). 101

Larochelle, H., D. Erhan, A. Courville, J. Bergstra and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation", in "Proceedings of the 24th international conference on Machine learning", pp. 473–480 (ACM, 2007). 58, 67, 109

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", Neural computation **1**, 4, 541–551 (1989). 49

LeCun, Y., B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard and L. D. Jackel, "Handwritten digit recognition with a back-propagation network", in "Advances in neural information processing systems", pp. 396–404 (1990). 19

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE **86**, 11, 2278–2324 (1998a). 19, 58, 67, 81

LeCun, Y., C. Cortes and C. Burges, "Mnist handwritten digit database", (1998b). 103

Li, Y., D. M. Tax, R. P. Duin and M. Loog, "Multiple-instance learning as a classifier combining problem", Pattern Recognition **46**, 3, 865–874 (2013). 26, 36, 37

Li, Z. and D. Hoiem, "Learning without forgetting", in "Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 614–629 (Springer, 2016). 102

Maron, O. and T. Lozano-Pérez, "A framework for multiple-instance learning", NIPS pp. 570–576 (1998). 22, 24, 36, 40, 45

Maron, O. and A. Ratan, "Multiple-instance learning for natural scene classification", in "IEEE ICML", vol. 15, pp. 341–349 (1998). 21

McCloskey, M. and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem", Psychology of learning and motivation **24**, 109–165 (1989). 100

McCormick, B. and M. Goldbaum, "Stare= structured analysis of the retina: Image processing of tv fundus image", in "del USA-Japan Workshop on Image Processing, Jet Propulsion Laboratory, Pasadena, CA", (1975). 38

Mensink, T., J. Verbeek, F. Perronnin and G. Csurka, "Metric learning for large scale image classification: Generalizing to new classes at near-zero cost", in "Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 488–501 (Springer, 2012). 101

Mensink, T., J. Verbeek, F. Perronnin and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost", IEEE Trans. on Pattern Analysis and Machine Intelligence **35**, 11, 2624–2637 (2013). 101

Muhlbaier, M. D., A. Topalis and R. Polikar, "Learn++. nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes", IEEE Trans. on Neural Networks **20**, 1, 152–168 (2009). 101

Nagesh, P. and B. Li, "A compressive sensing approach for expression-invariant face recognition", in "Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on", pp. 1518–1525 (IEEE, 2009). 5

Nair, V. and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in "Proceedings of the 27th International Conference on Machine Learning (ICML-10)", pp. 807–814 (2010). 60, 83, 105

Nesterov, Y., "A method of solving a convex programming problem with convergence rate o (1/k2)", in "Soviet Mathematics Doklady", vol. 27,2, pp. 372–376 (1983). 84

Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning", in "NIPS workshop on deep learning and unsupervised feature learning", No. 2, p. 5 (Granada, Spain, 2011). 58, 67, 103

Nguyen, A., J. Yosinski and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 427–436 (2015). 49

Polikar, R., L. Upda, S. S. Upda and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks", IEEE Trans. on Systems, Man, and Cybernetics, part C (Applications and Reviews) **31**, 4, 497–508 (2001). 101

Polyak, B. T., "Some methods of speeding up the convergence of iteration methods", USSR Computational Mathematics and Mathematical Physics **4**, 5, 1–17 (1964). 60, 61, 84

Quellec, G., M. Lamard, M. Abràmoff, E. Decencière, B. Lay, A. Erginay, B. Cochener and G. Cazuguel, "A multiple-instance learning framework for diabetic retinopathy screening", Medical Image Analysis (2012a). 38

Quellec, G., M. Lamard, B. Cochener, C. Roux, G. Cazuguel, E. Decenciere, B. Lay and P. Massin, "A general framework for detecting diabetic retinopathy lesions in eye fundus images", in "Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on", pp. 1–6 (IEEE, 2012b). 38

Rahmani, R., S. Goldman, H. Zhang, S. Cholleti and J. Fritts, "Localized content-based image retrieval.", IEEE transactions on pattern analysis and machine intelligence **30**, 11, 1902 (2008). 22, 25, 26

Rebuffi, S.-A., A. Kolesnikov and C. H. Lampert, "iCaRL: Incremental classifier and representation learning", in "accepted to the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2017). 102

Ristin, M., M. Guillaumin, J. Gall and L. Van Gool, "Incremental learning of NCM forests for large-scale image classification", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 3654–3661 (2014). 101

Robins, A., "Catastrophic forgetting, rehearsal and pseudorehearsal", Connection Science **7**, 2, 123–146 (1995). 100, 103, 106

Romero, A., N. Ballas, S. E. Kahou, A. Chassang, C. Gatta and Y. Bengio, "Fitnets: Hints for thin deep nets", arXiv preprint arXiv:1412.6550 (2014). 72, 75, 76, 100

Roweis, S. T. and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding", science **290**, 5500, 2323–2326 (2000). 5

Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation", Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985). 13, 15

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge", International Journal of Computer Vision (IJCV) pp. 1–42 (2015). 52, 53, 71

Seff, A., A. Beatson, D. Suo and H. Liu, "Continual learning in generative adversarial nets", arXiv preprint arXiv:1705.08395 (2017). 112

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556 (2014a). 19

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", CoRR **abs/1409.1556**, URL `http://arxiv.org/abs/1409.1556` (2014b). 51, 71, 82

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", The Journal of Machine Learning Research **15**, 1, 1929–1958 (2014). 60, 83, 105

Srivastava, R. K., K. Greff and J. Schmidhuber, "Highway networks", arXiv preprint arXiv:1505.00387 (2015). 20

Srivastava, R. K., J. Masci, S. Kazerounian, F. Gomez and J. Schmidhuber, "Compete to compute", in "Advances in neural information processing systems", pp. 2310–2318 (2013). 100

Szegedy, C., S. Ioffe, V. Vanhoucke and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.", in "AAAI", pp. 4278–4284 (2017). 20

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions", arXiv preprint arXiv:1409.4842 (2014). 49

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1–9 (2015). 20

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2818–2826 (2016). 20

Tenenbaum, J. B., V. De Silva and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction", science **290**, 5500, 2319–2323 (2000). 5

Thorpe, S., D. Fize and C. Marlot, "Speed of processing in the human visual system", nature **381**, 6582, 520 (1996). 1

Venkatesan, R., P. Chandakkar and B. Li, "Simpler non-parametric methods provide as good or better results to multiple-instance learning", in "The IEEE International Conference on Computer Vision (ICCV)", (2015). 4, 48

Venkatesan, R., P. Chandakkar, B. Li and H. K. Li, "Classification of diabetic retinopathy images using multi-class multiple-instance learning based on color correlogram features", in "Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE", pp. 1462–1465 (IEEE, 2012a). 4, 10, 39

Venkatesan, R., P. Chandakkar, B. Li and H. K. Li, "Classification of diabetic retinopathy images using multi-class multiple-instance learning based on color correlogram features", in "Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE", pp. 1462–1465 (IEEE, 2012b). 12

Venkatesan, R., V. Gattupalli and B. Li, "Neural dataset generality", arXiv preprint arXiv:1605.04369 (2016a). 6, 66, 109

Venkatesan, R., V. Gattupalli and B. Li, "Neural dataset generality", CoRR **abs/1605.04369**, URL `http://arxiv.org/abs/1605.04369` (2016b). 66

Venkatesan, R. and B. Li, "Diving deeper into mentee networks", arXiv preprint arXiv:1604.08220 (2016). 7, 100

Venkatesan, R. and B. Li, *Convolutional Neural Networks in Visual Computing: A Concise Guide*, Data-Enabled Engineering (Taylor & Francis Group, 2017), URL `https://books.google.com/books?id=hFUknQAACAAJ`. 3

Venkatesan, R., H. Venkateshwara, S. Panchanathan and B. Li, "A strategy for an uncompromising incremental learner", arXiv preprint arXiv:1705.00744 (2017). 7

Wang, D., C. Liu, Z. Tang, Z. Zhang and M. Zhao, "Recurrent neural network training with dark knowledge transfer", arXiv preprint arXiv:1505.04630 (2015). 72, 75

Wang, H., H. Huang, F. Kamangar, F. Nie and C. Ding, "Maximum margin multi-instance learning", in "NIPS", (NIPS, 2011a). 21, 27

Wang, H., F. Nie and H. Huang, "Learning instance specific distance for multi-instance classification.", in "AAAI", (2011b). 22, 27, 36, 37

Wang, H., F. Nie and H. Huang, "Robust and discriminative distance for multi-instance learning", in "IEEE CVPR", pp. 2919–2924 (IEEE, 2012a). 21, 27

Wang, J. and J. Zucker, "Solving the multiple-instance problem: A lazy learning approach", in "Proceedings of the Seventeenth International Conference on Machine Learning", pp. 1119–1126 (Morgan Kaufmann Publishers Inc., 2000). 22, 26, 30, 36, 37, 40

Wang, Q., L. Si and D. Zhang, "A discriminative data-dependent mixture-model approach for multiple instance learning in image classification,", in "In Proceedings of the 12th European Conference on Computer Vision (ECCV-12),", (2012b). 27

Wang, Z., S. Gao and L.-T. Chia, "Learning class-to-image distance via large margin and l1-norm regularization", in "Computer Vision ECCV 2012", pp. 230–244 (2012c). 22, 27, 36, 37

Watamaniuk, S. N. and A. Duchon, "The human visual system averages speed information", Vision research **32**, 5, 931–941 (1992). 1

Welling, M., "Herding dynamical weights to learn", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 1121–1128 (2009). 102

Wu, D., J. Bi and K. Boyer, "A min-max framework of cascaded classifier with multiple instance learning for computer aided diagnosis", in "IEEE CVPR", pp. 1359–1366 (IEEE, 2009). 28

Xiao, T., J. Zhang, K. Yang, Y. Peng and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification", in "Proceedings of the ACM Intl. Conf. on Multimedia (ACM-MM)", pp. 177–186 (2014). 101

Xu, Y., J. Zhu, E. Chang and Z. Tu, "Multiple clustered instance learning for histopathology cancer image segmentation, classification and clustering", CVPR (IEEE, 2012). 28

Yosinski, J., J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?", in "Advances in Neural Information Processing Systems", pp. 3320–3328 (2014). 53, 57

Zhang, D., Y. Liu, L. Si, J. Zhang and R. Lawrence, "Multiple instance learning on structred data", in "Twenty-Fifth Annual Conference on Neural Information Processing Systems (NIPS)", (2011). 27

Zhang, Q. and S. Goldman, "Em-dd: An improved multiple-instance learning technique", Advances in neural information processing systems **14**, 1073–1080 (2001). 22, 25, 26, 36, 40

Zhang, Q., S. Goldman, W. Yu and J. Fritts, "Content-based image retrieval using multiple-instance learning", in "Machine Learning-International Worskshop-Then Conference-", pp. 682–689 (2002). 21

Zhang, Q. and B. Li, "Discriminative k-svd for dictionary learning in face recognition", in "Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on", pp. 2691–2698 (IEEE, 2010). 5

APPENDIX A

PERMISSION STATEMENTS FROM CO-AUTHORS

Permission for including co-authored material in this dissertation was obtained from co-authors, Prof. Baoxin Li, Parag Chandakkar and Vijetha Gatupalli.