INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY DELHI

# Data Constrained Deep Learning

by

Rohit Keshari

Under the supervision of

Prof. Richa Singh

Prof. Mayank Vatsa

Indraprastha Institute of Information Technology Delhi
March, 2024


# Data Constrained Deep Learning

by

Rohit Keshari

Submitted
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

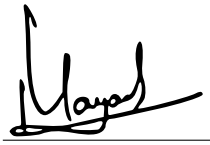
to the
Indraprastha Institute of Information Technology Delhi
March, 2024

# Certificate

This is to certify that the thesis titled "**Data Constrained Deep Learning**" being submitted by **Rohit Keshari** to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Prof. Richa Singh
March, 2024



Prof. Mayank Vatsa
March, 2024

Indraprastha Institute of Information Technology Delhi
New Delhi 110020

# Acknowledgment

I would like to express my heartfelt gratitude to the many individuals who have contributed to my journey towards obtaining a Ph.D. This endeavor would not have been possible without the unwavering support, guidance, and encouragement of numerous people, whose collective efforts have shaped both my academic and personal growth.

First and foremost, I extend my deepest appreciation to my esteemed advisors, **Prof. Richa Singh** and **Prof. Mayank Vatsa**. Their guidance, expertise, and constant support have been instrumental in shaping my research. Their visionary insights and unwavering commitment to excellence have inspired me throughout this journey. I am profoundly grateful for their mentorship, which has not only enriched my academic knowledge but also nurtured my research skills. I would also like to thank **Prof. Afzel Noore** for hosting me during my internship at West Virginia University. His patience, willingness, and constant feedback helped me to grow as a researcher.

I would like to acknowledge the invaluable contributions of my collaborators: **Anush**, **Tejas**, **Soumyadeep**, **Akshay**, **Saheb**, **Pavani**, **Yasmeena**, **Mahapara**, and **Ishan**. Your expertise, insights, and dedication have enhanced the quality of our research manifold. Working with such a talented and dedicated team has been an enriching experience, and I am grateful for the collaborative spirit that you have all brought to our projects.

I would like to extend my heartfelt thanks to my beloved wife, **Sharmistha**. Her unwavering support, understanding, and sacrifices have been the bedrock of my academic pursuits. Her belief in me has been a constant source of motivation, and her presence has made this journey all the more meaningful. To my **parents**, whose love, encouragement, and sacrifices have been the cornerstone of my life, I owe an immense debt of gratitude. Their unwavering support and belief in my abilities have been a driving force behind my academic achievements.

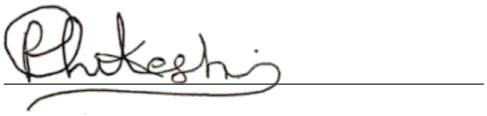
I am thankful to all the faculty members and staff at **IIIT-Delhi, India** for creating a conducive academic environment. The intellectual discussions, resources, and opportunities provided by the university have been pivotal in shaping my research journey. I extend my gratitude to my friends and colleagues, who have been a constant source of support and camaraderie. Your friendship and encouragement have made the challenges of graduate life more manageable. I also want to express my appreciation for the financial support I received through scholarships and grants, which

allowed me to pursue my research without undue financial stress.

To all those whose names I may have inadvertently omitted but have nonetheless played a role in my academic journey, please accept my sincere thanks. Your contributions, however small or significant, have been deeply appreciated. Lastly, I would like to dedicate this achievement to my family, whose unwavering belief in me has been my greatest strength. This Ph.D. is not just my accomplishment but a collective achievement of all those who have stood by me.

In conclusion, this journey has been a rollercoaster of challenges and triumphs, and I am grateful to each and every one of you who has been a part of it. Your support and encouragement have been invaluable, and I am excited to carry the lessons and experiences from this journey forward into the next chapter of my academic and professional life.

Thank you from the bottom of my heart.



Rohit Keshari

March, 2024

PhD14004

# Data Constrained Deep Learning

by

Rohit Keshari

## Abstract

Deep Neural Networks (DNNs) have achieved remarkable success across various machine learning and computer vision tasks, especially when abundant training samples are available. In Convolutional Neural Network (CNN) research, it has been established that a model's generalization capability improves with the combination of complex architectures, strong regularization, domain-specific loss functions, and extensive databases. However, training DNNs in environments with limited data remains a significant challenge, calling for attention from the research community. Many applications lack the requisite volume of data needed to train models effectively. Data constraint in this context is influenced by factors such as 1) a scarcity of domain experts, 2) long-tail distribution in large datasets, 3) insufficient domain-specific data, and 4) the challenge of mimicking human cognition and learning. The issues above are common challenges encountered while designing deep models, underscoring the importance of addressing *Data Constrained Learning* (DCL). This thesis investigates the formulation of deep learning strategies explicitly tailored for scenarios with DCL. The objective is to ensure that the training of numerous parameters does not adversely affect the model's ability to learn meaningful patterns, as this could elevate the risk of overfitting and result in suboptimal generalization performance.

To address the DCL challenge, we introduce a novel strength parameter in deep learning named SSF-CNN, which concentrates on learning both the "structure" and "strength" of filters. The filter structure is initialized using a dictionary-based filter learning algorithm, while the strength is learned under data-constrained settings. This architecture demonstrates adaptability, delivering robust performance even when used with small databases and consistently attaining high accuracy. We validate the effectiveness of our algorithm on databases such as MNIST, CIFAR10, and NORB, with varying training sample sizes. The results indicate that SSF-CNN substantially reduces the required training parameters while maintaining high test accuracy. Our approach achieves state-of-the-art results for real-world data-constrained problems such as newborn face recognition and the Omniglot dataset. Notably, on the IIITD Newborn Face Database, our method enhances rank-1 identification accuracy by at least 10

In our second contribution, we propose Guided Dropout, a novel regularization technique tailored for the DCL problem, enhancing the traditional Dropout method often used in deep neural networks to mitigate overfitting. Standard Dropout randomly drops nodes from a Neural Network (NN) during training. In contrast, the proposed Guided Dropout strategically selects nodes to drop, leading to better generalization than its traditional counterpart. We also establish that conventional Dropout is a specific instance of the proposed Guided Dropout. Through extensive experimentation on multiple datasets, including MNIST, CIFAR10, CIFAR100, SVHN, and Tiny ImageNet, we demonstrate the superior performance of Guided Dropout.

Our third contribution addresses challenges in zero-shot and generalized zero-shot learning (ZSL and GZSL), where, for a class, few or no samples are present in the training set, and only class attributes are known. The performance of many supervised DNN algorithms deteriorates in ZSL settings, highlighting the necessity for model generalization while learning the mapping from class to attribute. To address this, we modify the input and feature space across the deep learning pipeline. Furthermore, to ensure robust performance on both seen and unseen classes, we introduce an Over-Complete Distribution (OCD) generated using a Conditional Variational Autoencoder (CVAE). On this generated OCD, the proposed Online Batch Triplet Loss (OBTL) and Center Loss (CL) work to enhance class separability and reduce intra-class variance, improving performance in ZSL/GZSL scenarios across various benchmark databases.

In our fourth contribution, we focus on model space, particularly in CNN models, to address training challenges in data-constrained environments, which typically require millions of parameters. Reducing the number of parameters may compromise model performance. To address this, we introduce Guided DropBlock and Filter Augmentation for resource-constrained deep learning scenarios. Guided DropBlock is inspired by guided Dropout and the DropBlock regularization methods. Unlike its predecessor, which randomly omits a contiguous image segment, the proposed approach is more selective, focusing the omission on the background and specific blocks that carry critical semantic information about the objects in question. On the other hand, the filter augmentation technique we propose involves performing a series of operations on the Convolutional Neural Network (CNN) filters during the training phase. Our findings indicate that integrating filter augmentation while fine-tuning the CNN model can substantially enhance performance in data-limited situations. This approach results in a smoother decision boundary and behavior resembling an ensemble model. Imposing these additional constraints on loss optimization helps mitigate the challenges posed by data scarcity, ensuring robust feature extraction from the input signal, even when some learnable parameters within the CNN layers are frozen. We have validated these enhancements on seven publicly accessible benchmark datasets and two real-world use cases, namely, identifying newborns and monitoring post-cataract surgery conditions, providing empirical support for our claims.

# Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

# List of Figures

# List of Tables

# Acronyms

**ADDA** Adversarial Discriminative Domain Adaptation. 12

**CL** Center Loss. 20, 59, 61, 65, 66, 70

**CMPD** Cataract Mobile Periocular Database. 90, 91, 99

**CNN** Convolutional Neural Network. 8, 14, 15, 17, 18, 21–30, 32–36, 38–40, 42, 45, 53, 54, 56, 77, 79–81, 85, 87, 93

**CVAE** Conditional Variational Autoencoder. 22, 60, 61, 64–70, 72

**DADA** Deep Adversarial Aata Augmentation. 12

**DCL** Data Constrained Learning. 6, 8–11, 15, 18, 19, 21

**DFT** Data Fine-Tuning. 14

**DML** Deep Metric Learning. 18–20

**DNN** Deep Neural Network. 21, 41, 57

**DTN** Domain Transfer Network. 12

**FA** Filter Augmentation. 79, 83, 85, 90

**GAN** Generative Adversarial Network. 12, 13

**GZSL** Generalized Zero-Shot Learning. 13, 20, 21, 58–60, 68–70, 72, 73

**ILSVRC** ImageNet Large Scale Visual Recognition Challenge. 5

**LFW** Labeled Faces in the Wild. 38

**LSTM** Long Short-Term Memory Networks. 81

**ML** Machine Learning. 6, 8, 10, 13

**MMD** Maximum Mean Discrepancy. 11, 13, 16, 17

**NN** Neural Network. 2, 42–45, 47, 48, 50, 51, 53, 55

**OBTL** Online Batch Triplet Loss. 20, 59, 61, 65, 66, 70, 72, 73

**OCD** Over-Complete Distribution. 58, 59, 61–67, 69, 70, 72, 73

**PixelDA** Pixel level Domain Adaptation. 12

**RNN** Recurrent Neural Networks. 18

**SAE** Stacked Auto-Encoder. 15

**SDAE** Stacked Denoising Auto-Encoder. 15

**SELU** Scaled Exponential Linear Unit. 41, 49

**SGD** Stochastic Gradient Descent. 29, 31, 32

**SOTA** state-of-the-art. 59, 96, 98, 99

**SSF** Structure and Strength Filtered. 24, 26, 28, 33–36, 39

**SVM** Support Vector Machine. 2, 16

**TJM** Transfer Joint Matching. 11

**VQA** Visual Question Answering. 13

**YTF** YouTube Faces. 38

**ZSL** Zero-Shot Learning. 13, 20, 21, 57, 59, 60, 62, 64, 66, 68–70, 72, 73

THIS PAGE INTENTIONALLY LEFT BLANK

# Publications

## A* Conferences/Journals

1. **R. Keshari**, M. Vatsa, R. Singh, and A. Noore, **"Learning Structure and Strength of CNN Filters for Small Sample Size Training"**, Conference on Computer Vision and Pattern Recognition (**CVPR**), 2018, pp. 9349-9358

2. **Keshari, R.**, Singh, R., Vatsa, M., **"Guided Dropout"**., In Association for the Advancement of Artificial Intelligence (**AAAI**), 2019, Vol. 33, pp. 4065-4072

3. Nigam, I*., **Keshari, R.***, Vatsa, M., Singh, R., and Bowyer, K. ''**Phacoemulsification Cataract Surgery Affects the Discriminative Capacity of Iris Pattern Recognition**''., *Scientific Reports, Nature* 9, 11139 (2019) doi:10.1038/s41598-019-47222-4.

4. A. Agarwal, **R. Keshari**, M. Wadhwa, M. Vijh, C. Parmar, R. Singh, and M. Vatsa, **"Iris sensor identification in multi-camera environment"**, In Information Fusion, 2019, volume 45, Pages 333-345

5. **R. Keshari**, R. Singh, and M. Vatsa, **"OCD: Over-Complete Distribution Generation For Zero-Shot Learning"**, Conference on Computer Vision and Pattern Recognition (**CVPR**), 2020

6. Tripathi, P., Y. Akhterb, M. Khurshidb, Lakra, A., **Keshari, R.**, Vatsa, M., Singh, R., ''**MTCD: Cataract Detection via Near Infrared Eye Images**'', Computer vision and image understanding, 2021

7. **R. Keshari**, M. Vatsa, and R. Singh, **"Guided DropBlock and Filter Augmentation in CNNs for Data Constrained Learning"**, IEEE TMLR, 2023 (Under Submission)

## Other Peer Reviewed Conferences

1. **R. Keshari**, S. Ghosh, A. Agarwal, R. Singh, and M. Vatsa, **"Mobile Periocular Matching With PRE-POST Cataract Surgery"**, In IEEE International Conference on Image Processing (ICIP), 2016, Vol. 23, pp. 3116-3120

2. M. Vatsa, **R. Keshari**, S. Ghosh, S. Chhabra, R. Singh, **"Finding Solutions to Small Sample Size Problems in the Deep Learning World"**, IEEE Sixth International Conference on Multimedia Big Data (BigMM), 2020

3. P. Tripathi, **R. Keshari**, S. Ghosh, M. Vatsa, R. Singh, **"AUTO-G: Gesture Recognition in the Crowd for Autonomous Vehicl"**, IEEE International Conference on Image Processing (ICIP), 2019, Vol. 26, pp. 3482-3486

4. A. Lakra, P. Tripathi, **R. Keshari**, M. Vatsa, R. Singh, **"SegDenseNet: Iris Segmentation for Pre and Post Cataract Surgery"**, International Conference on Pattern Recognition (ICPR), 2018, Vol. 24, pp. 3150-3155

5. S. Ghosh, **R. Keshari**, R. Singh, M. Vatsa, **"Face Identification from Low Resolution Near-Infrared Images"**, IEEE International Conference on Image Processing (ICIP), 2016, Vol. 23, pp. 938-942

6. S. Ghosh, T. I. Dhamecha, **R. Keshari**, R. Singh, and M. Vatsa, **"Feature and Keypoint Selection for Visible to Near-infrared Face Matching"**, In Proceedings of IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS), 2015, pp. 1-7

7. A. Sankaran, A. Agarwal, **R. Keshari**, S. Ghosh, A. Sharma, M. Vatsa, and R. Singh, **"Latent Fingerprint from Multiple Surfaces: Database and Quality Analysis"**, In Seventh IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS), 2015, pp. 1-6.

## Book Chapter

1. Tripathi, P., **Keshari, R.**, Vatsa, M., Singh, R., ''**Evolution of Newborn Face Recognition**'', Deep Learning-Based Face Analytics. Springer, Cham, 2021. 167-187

* These authors contributed equally.

# Chapter 1

# Introduction

*"What we want is a machine that can learn from experience."*

— Alan Turing, London, 1947

Alan Turing put forward the idea of creating a machine capable of learning from experiences, much like human beings do. Over the course of machine learning's evolution, various classifiers have been introduced. A fundamental prerequisite for establishing a machine learning system is the availability of ample high-quality training data. This necessity has become increasingly crucial as the field has advanced, with today's prevalent deep learning models demanding extensive training samples [91, 103, 167].

As depicted in Figure 1-1 (bottom), there has been an exponential increase in the number of learnable parameters from the time classifiers like perceptron, RDF [96], SVM [41], LeNet [139], and GoogLeNet [240] were published. Correspondingly, the requisite volume of observations in datasets needed to train these classifiers has also escalated, as illustrated in Figure 1-1 (top). In contemporary settings, most state-of-the-art classifiers are designed with multiple hidden layers (making them deep) and many nodes in each hidden layer, forming Deep Neural Networks (DNNs) inspired by the human brain's structure. The rise and success of deep learning methodologies are chiefly attributed to two critical factors: the accessibility of hardware resources and the availability of extensive training samples. Deep learning models have excelled in issues where large training databases are available, demonstrating unparalleled performance. Nevertheless, training DNNs under constraints, such as limited resources or small data sample sizes, may impede

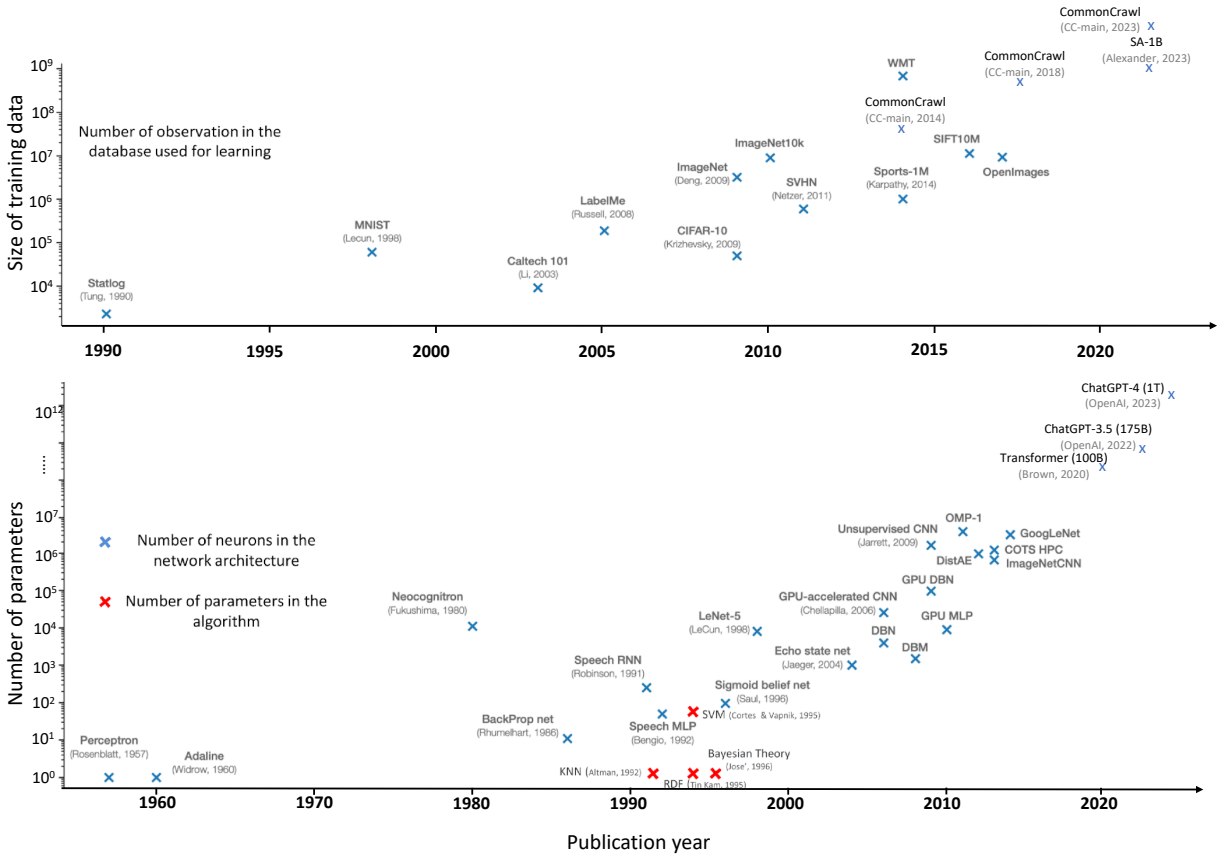



Figure 1-1: Illustrates the growth of several trainable parameters in classifiers from the past few decades. (**Bottom**) It can be observed that the perceptron with fewer parameters was proposed in early 1960. The performance of the Neural Network (NN) build on the perceptrons was limited. The complexity of NN increases while increasing the depth and width of NN without significantly improving accuracy. On the other hand, Support Vector Machine (SVM) became popular around 1990-2000 because having a clear mathematical understanding of the model and complexity depends on handling the data. (**Top**) It can also be observed that after the year 2000, the requirement for large-size databases increased exponentially [24].

the training process, given the necessity of resources like GPUs, and result in suboptimal generalization [65, 204]. Conversely, humans are capable of recognizing objects with just a single sample. Hence, enhancing the generalization capability of DNNs when faced with limited data is of paramount importance. Challenges of this nature fall under the category of Data-Constrained Learning (DCL) [65, 81, 97, 112, 115, 204], where acquiring large training datasets is impractical.

Figure 1-2 showcases the evolution from shallow to deep networks, highlighting the interplay between the size of the available training data and the resulting performance of the algorithms. It is critical to note that algorithms yielding superior results with extensive data may not maintain

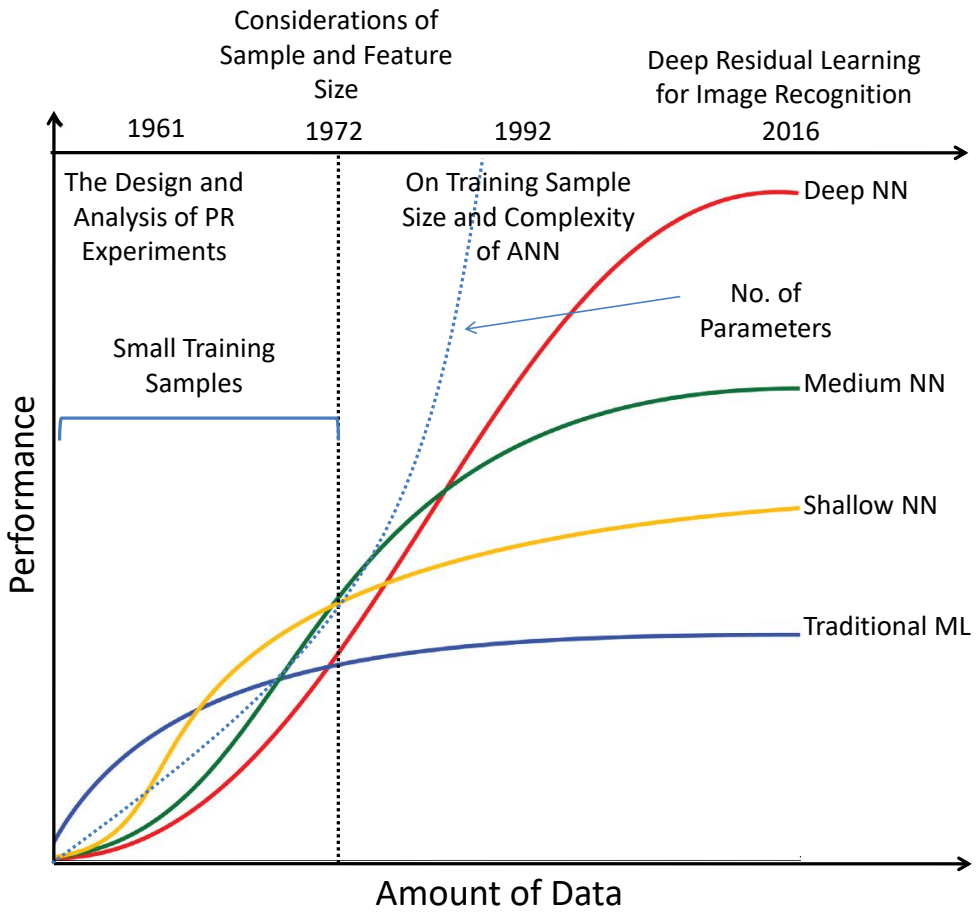



Figure 1-2: Illustrates the development of a shallow network to a deep network on the availability of a large amount of training data. It also shows the exponential growth in the number of learnable parameters while utilizing the deeper neural network. The base graph is taken from [45], and additional information is included to showcase the problem of Small Sample Size.

their efficacy when applied to smaller datasets. Foley [65] highlighted that the error rate derived from a training set is a skewed predictor of the classifier's true error rate, and he proposed a method to estimate this bias based on the sample size per class. Furthermore, he acknowledged the longstanding challenge of determining the optimal feature size for a sample. The findings indicate that when the ratio of samples per class to dimensions ($\frac{N}{L}$) falls below three, the training set error rate becomes a notably unreliable estimator of both the Bayes error rate and the test-set error rate. Moreover, he outlined that the error rate variance on the training set can be expressed through a function with an upper bound of $\frac{1}{8N}$.

Raudys and Jain [204] have explained the effect of the data-constrained problem in statistical pattern recognition. The authors have presented the significance of sample size in training and


Figure 1-3: Illustrates a few examples of rare diseases and caption generation from scenes. (**Top**). Aceruloplasminemia, Amyloidosis, Papillitis [57], Iris melanoma. (**Bottom**) The deep neural network has generated the captions. From the images, it can be observed that intuition and concept are missing in the generated captions. *Image credit:* The picture has been taken from [131].

testing sets to evaluate machine learning systems. The training and testing sets should be large enough to design and evaluate a complex system. They have also discussed problems in the feature selection and error estimation process while having a limited data size.

Conversely, amassing a vast quantity of samples and providing them with annotations is not consistently practical. For instance, images portraying rare ocular conditions are displayed in the upper section of Figure 1-3. Acquiring a plethora of examples for such infrequent diseases presents a considerable challenge, given the scarcity of documented cases [245]. In an additional scenario depicted in the lower section of Figure 1-3, captions produced by a deep learning model are showcased. However, these generated texts fail to accurately capture the scenes' details, a discrepancy attributable to the insufficient training samples representing the situations in the scenes. The issue of limited data availability extends beyond specific applications, with numerous factors contributing to data constraints, which are detailed as follows:

1. **Due to high-cost human annotations:** Acquiring unstructured and unlabeled data is relatively straightforward, as it can be readily sourced from the internet. However, this data

often necessitates extensive preprocessing and cleaning, necessitating human intervention and creating a bottleneck for utilizing such unclean data. In practical applications, obtaining a substantial volume of high-quality labeled data becomes challenging due to the scarcity of domain experts. For instance, in the typical end-to-end semantic segmentation task, even training with an extensive dataset is susceptible to failure, particularly when the annotations provided are imprecise or inadequate [98].

2. **Long-tail distribution in big data:** The long-tail distribution problem occurs when some classes have many occurrences and many are not, making the distribution skewed in one direction [11]. The portion of the distribution where a class with significant occurrences is known as "head". While the portion of the distribution where a class with less occurrence is known as "tail". The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset [211] contains millions of samples. However, Ouyang *et al.* [189] and Rahman *et al.* [200] have shown that the ImageNet dataset has long-tail distribution. They have analyzed that only 11 classes out of 1000 classes of the database cover nearly 50% of the entire database. Consequently, learning a classifier on such databases might have biasing problems for the "head" classes. So, the model's performance generalizes poorly on "tail" classes. A simple solution for imbalanced data is to re-balance the data: 1) by sampling and 2) by reducing the samples of head classes [89]. Still, creating multiple samples of the tail distribution leads to overfitting due to the repetition of samples. On the other hand, reducing samples of the head distribution may compromise the critical information of the classes [269].

3. **Domain-specific insufficient data:** In some applications, collecting large samples is challenging. As shown in Figure 1-3 (**top**), having multiple samples of rare diseases is challenging due to the limited availability amount of unique case studies. Figure 1-3 (**bottom**) illustrates the caption generation from the deep learning model. The generated text does not match the scenes in the images due to limited training samples of situations pressed. Hence, designing a model that can be trained on actual data behavior and search through a predefined space of potential hypotheses is essential. Utilizing these learning methods could

improve the generalization of the models on the unseen testing samples.

4. **Mimic human's thinking and learning:** The field of Machine Learning (ML) started with the motivation of making machines to learn as close as human's thinking [212]. Due to strong generalization capability, humans can understand objects even when one sample is available for the respective object. On the other hand, machines require multiple instances from multiple views of an object in the training data to produce samples for generalization. All possible views and samples are not likely to be collected. Therefore, training a deep network on a smaller number of samples is a significant problem requiring the research community's attention.

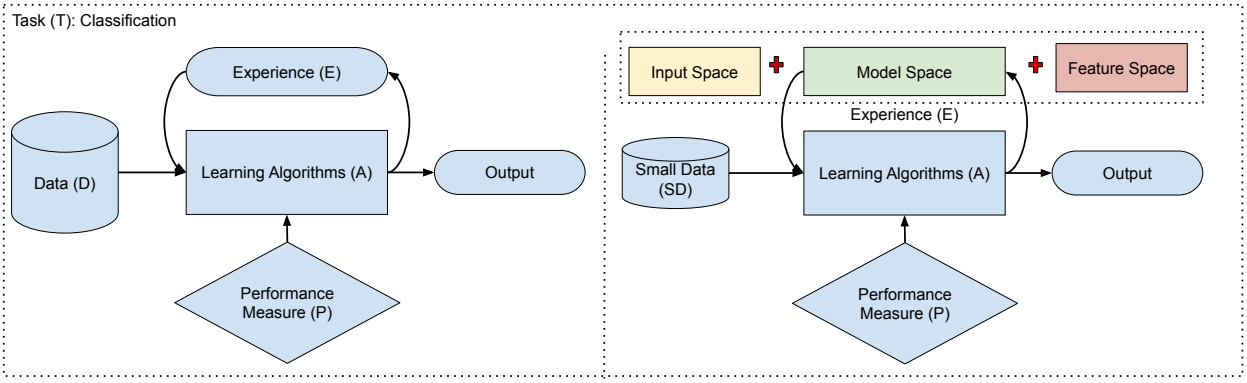



Figure 1-4: Graphical illustration of learning of task (T). (**Left:**) conventional machine learning pipeline, where the performance (P) of the learning algorithm (A) improves with experience (E) on the given task (T). (**Right:**) Illustrate the Data Constrained Learning (DCL) learning pipeline, where experience has been extended by knowledge and concept systems. In this case, data augmentation and matching rules can also be utilized to improve the performance.

## 1.1 Definition of Learning Problem (LP)

The definition of learning problem by Thomas M. Mitchell [176] is: "A computer program is said to learn from experience E with respect to some class of task T and performance P, if its performance at tasks in T, as measured by P, improves with experience E."

Figure 1-4 (**Left**) is the conventional machine learning pipeline which represents the learning problem containing sub-modules such as Data (D), Experience (E), Learning Algorithm (A), and Performance Measure (P). Now, for the given D, learning optimizes the algorithm's parameter (A) to improve the performance as the experience of the model. The E has been gained iteratively on multiple samples from the training set.

### 1.1.1 Constrained Learning Problem (CLP)

If any sub-modules in the learning pipeline are limited because of constraints because of hardware (compute and memory), software, graph based learning, special data type like series, limited data, etc., then the complete learning process would be depicted as constrained-based learning. Among all the CLPs, data constraints in deep learning are a more crucial and complex problem related to the human's thinking and perception of objects.

### 1.1.2 Data Constrained Learning (DCL)

Learning algorithms (A) can utilize the concept/experience learning along with data augmentation and matching rule as a part of experience E with respect to some class of task (T) and performance (P) if its performance at tasks (T), as measured by (P), improves with experience (E).

Figure 1-4 (**Right**) is the pictorial representation of a learning problem that utilizes the whole space (input, parameter, feature) to find the appropriate optimization for the specific task (T) when a small amount of data is available. Along with learning, E can be transferred or gained from iteratively learning on augmented samples with updated matching rules. DCL is also known as Learning with Less Labeling (LwLL), a program that is initiated by Defence Advanced Research Projects Agency (DARPA)[1]

### 1.1.3 DCL with Deep Learning

*"Deep Learning is getting really good on Big Data/millions of images. But Small Data is important too. Hope more researchers work on Small Data–ML needs more innovations there."*

[1] https://www.darpa.mil/program/learning-With-less-labeling

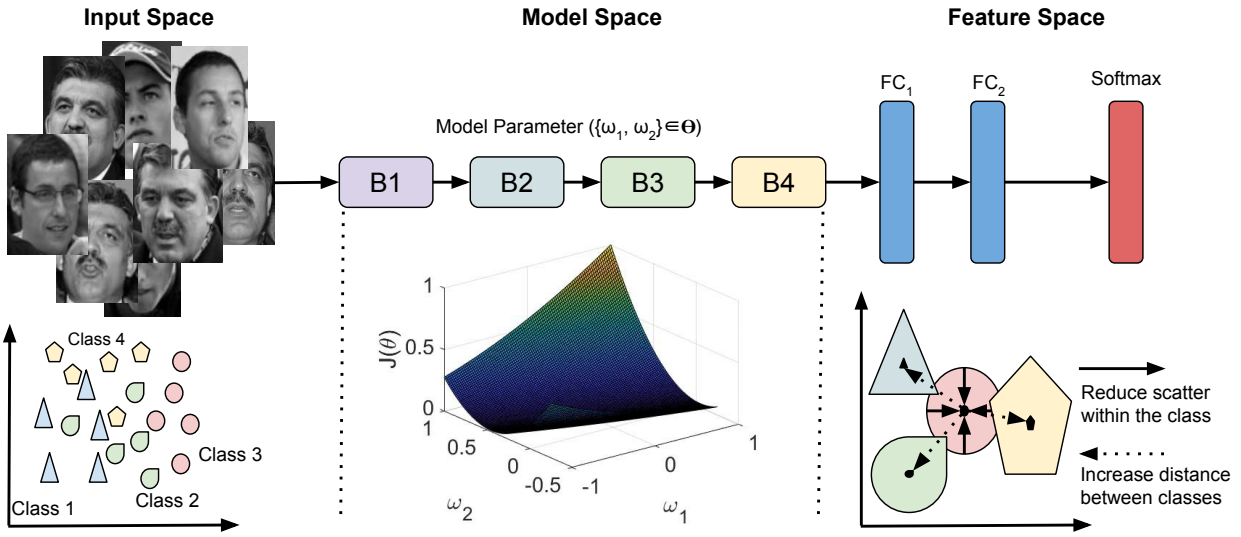



Figure 1-5: Illustration of input space, model space, and feature space for the deep learning model. In the case of a Convolutional Neural Network (CNN) model, B1 to B4 are basic blocks that contain operations such as $convolution$, $batchnorm$, and $ReLu$. $FC_1$ and $FC_2$ are fully connected layers. The last layer is a $softmax$ layer used for classification.

— Andrew Ng

The scientific problem is classified as DCL problem when the available training data is insufficient to learn the high dimensional feature and perform machine learning tasks. In such cases, scarcity of data leads to overfitting and inaccurate classification outputs. It is a significant research problem in the deep learning world and is rapidly gaining attention. Several solutions for DCL have been proposed by the researchers, and these solutions are often based on different questions, for instance, what are the database characteristics, and what is the deep learning pipeline being used for ML task.

Let us analyze the DCL problem using the general formulation of deep learning classifiers. Mathematically, a deep learning model can be represented as:

$$\mathbf{F} = \phi(\mathbf{W}\mathbf{X} + b) \tag{1.1}$$

where, $\phi$ is the model with weights $\mathbf{W}$ and bias $b$. This model inputs $\mathbf{X}$ and outputs the feature $\mathbf{F}$. We first propose that DCL algorithms can be categorized into *input, model*, and *feature space* depending on whether they are operating at $\mathbf{X}$, $< \mathbf{W}, \mathbf{b} >$ and $\mathbf{F}$, respectively (Figure 1-5). Input space refers to the set of algorithms that increases the database by generating more samples

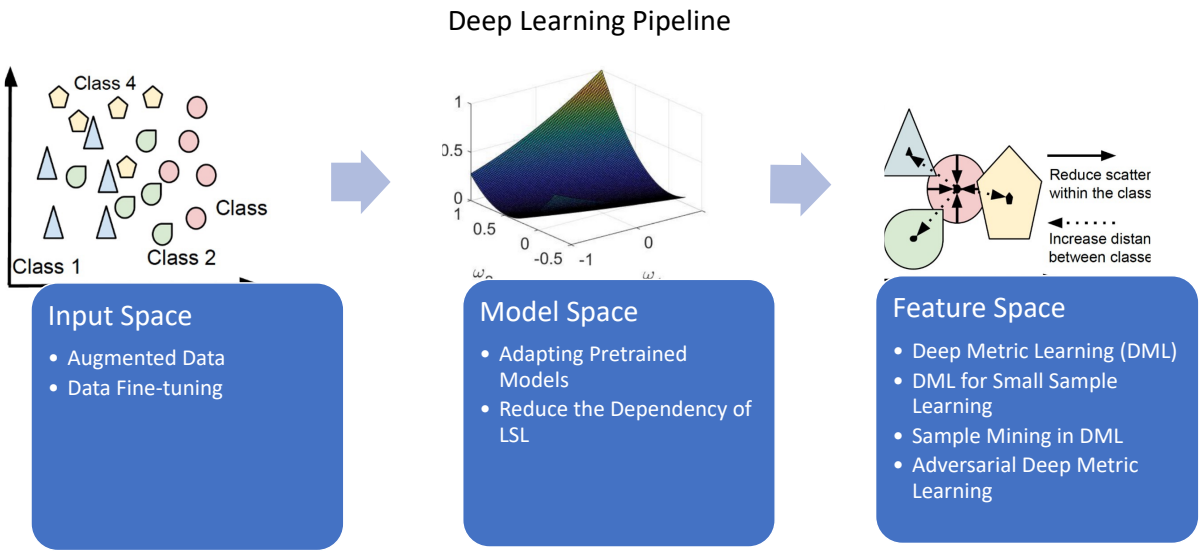



Figure 1-6: Illustration of taxonomy for DCL problem.

or perturbing the samples to optimize the feature space [18, 35, 105, 239]. In the feature space, algorithms operate on representations by reducing the intra-class distance and maximizing the inter-class distance to improve the classification performance [33, 84, 221]. The algorithms operating in the model space approximate the target function to map inputs to the outputs [91, 103]. Several regularization algorithms have also been proposed to learn better and generalize the model or target function [116, 233, 262].

In the era of deep learning, large training samples are the key to successfully training deep architectures, which have millions of learning parameters [91, 103, 127, 230, 240]. The deep neural network's learning parameters can be represented as $\theta$, $x$ is input, and $y$ is ground truth for $x$. Mathematically, learning of the network can be described as:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \tag{1.2}$$

where, $m$ is the number of training samples, and $h_\theta$ is the hypothesis $h_\theta(x) \in [0, 1]$. However, if training samples are less, the deep neural network might show overfitting on small training set samples. Therefore, Avoiding overfitting for DCL problems is challenging. Broadly, the taxonomy for the DCL problem can be divided into three learning spaces: 1) Input, 2) Model, and 3) Feature, as shown in Figure 1-6. Each of the spaces is explained in detail in the following sections 1.2, 1.3,

and 1.4.

It is worth mentioning that researchers have proposed solutions for data constrained problem in frequency domain [155, 270, 296]. CNNpack [270] is a technique to compress the CNN model in the frequency domain which focuses not only on smaller weights but on all the weights and their underlying connections. In order to achieve great compression without noticeably sacrificing accuracy, the authors propose that a large number of low-energy frequency coefficients in both sections can be eliminated. By convolutionally merging the discrete cosine transform (DCT) bases' convolution responses, they reduce the computational load of convolution operations in CNNs. Authors have proposed a frequency-domain dynamic pruning strategy to capitalize on the spatial correlations. The removal of unnecessary parameters has led to both network compression and acceleration because most CNN filters feature spatial redundancy. However, we are targeting to approach the DCL problem in the spatial domain.

## 1.2 Optimizing Input Space for DCL Problems

In machine (deep) learning, it is essential to understand the role of input space while learning the model space and feature space. Generally, the input space grows exponentially with the increase in dimension, which makes the ML problem intractable [51]. For instance, for a $100$ dimensional binary input data, there are $2^{100}$ possible inputs. A training set with trillion examples covers only $10^{-18}$ part of the input space, which is a small fraction. Machine learning models for DCL problems can be optimized if we can address data variations in the input space to an extent. These data variations broadly correspond to domain adaptation and zero-shot learning applications. In this section, we present an overview of data augmentation or alteration approaches that address the challenges related to supervised and unsupervised domain adaptation as well as zero-shot learning. Further, we discuss the concept of data fine-tuning to enhance the performance of pre-trained models.

### 1.2.1 Data Augmentation

For data-constrained learning problems, the role of input space on domain adaptation via data augmentation has been well explored in the literature. It is directly applied to data to increase the total

number of training samples to reduce the effect of DCL. This method aims to introduce multiple variability/views of the data, which might help train deep networks. The data augmentation can be utilized by doing 1) Transformation, 2) generative model, 3) pseudo-label method, 4) cross-domain synthesis, and 5) data transportation.

1. **Transformation–** Multiple transformations such as affine transformation [263], polar harmonic transform [288], pose and lighting [128] rotation [188], radial transform [216] have been utilized to augment the data. Moreover, for audio data, dynamic range compression, pitch shifting, and time stretching have been widely used augmentation methods [215]. Similarly, multiple transformation operations have been utilized to generate realistic data points as augmented data based on the understanding of domain knowledge [203].

   The idea is to form an augmented dataset in the target domain to compensate for small samples by transforming and augmenting the data from the source domain with certain constraints. [235] addressed the problem of covariate shift when the input training samples and testing samples follow different distributions. In conventional methods, the importance factor quantizes the covariate shift by accurately estimating the ratio between training and testing input density. However, it becomes hard to estimate the data density in higher-dimensional cases. Therefore, the authors proposed cross-validation-based techniques to compute the importance factor directly. To measure the discrepancy between source and target domains, Maximum Mean Discrepancy (MMD) based methods [10], [157], [191], [255] are widely used by researchers. However, these methods ignore the class weight bias across domains. To address this issue, [287] proposed a weighted MMD model which exploits the class prior probabilities of source and target domains. Saenko *et al.* introduced a new line of research based on the transformation between the source and target domains [214]. The aim is to learn a mapping between the points of the source domain and target domain in a supervised manner. Long *et al.* [158] have shown that feature matching and instance re-weighting play a key role in visual domain adaptation. Hence, they propose the Transfer Joint Matching (TJM) approach, which reduces the difference across domains by matching features and instance re-weighting in a dimensionality reduction manner. The proposed approach outputs a representation invariant to distribution differences and irrelevant instances.

2. **Generative model–** In this method, generative models have been used to create multiple samples, which can be further utilized for training the deep network [34, 36, 48]. Similarly, SimGAN has been proposed to generate more realistic data using unlabeled data [226]. Authors have improved gaze and hand pose estimation performance using these generated data [105, 220, 300].

   Bousmalis *et al.* [18] have proposed a novel technique that learns the transformation in pixel space between source and target domains in an unsupervised manner. They present a Generative Adversarial Network (GAN) based architecture which learns to map the images from the source domain to the target domain such that the images are drawn from the target domain. Their method, termed Pixel level Domain Adaptation (PixelDA), does not require one-to-one correspondence between source and target domain samples. Taigman proposes a Domain Transfer Network (DTN) [241], which maps a sample from the source domain to an analogous sample in the target domain such that the output of a function that takes images from either domain remains the same. This technique is performed in an unsupervised manner. Tzeng *et al.* [254] show that the generative adversarial networks are not good in discriminative tasks and are limited to smaller domain shifts. To address this problem, they proposed a generalized framework for adversarial adaptation that uses discriminative modeling, untied weight sharing, and a GAN loss and termed it as Adversarial Discriminative Domain Adaptation (ADDA). Murez *et al.* [177] have used the unpaired image-to-image translation framework and proposed a method to constrain the extracted features from the encoder network such that they can reconstruct the images in both source and target domains. Recently, Zhang *et al.* [302] proposed Deep Adversarial Aata Augmentation (DADA) technique to address the problem of extremely low data regimes. DADA enforces natural and augmented samples to contribute to finding the decision boundaries.

3. **Pseudo-label method–** This method is helpful for the semi-supervised problem, where less labeled training data is available. In this case, pseudo-labels are assigned to unlabeled data generated by curriculum/self-paced learning [12, 53, 109, 150], dual learning [88], and data programming [202]. In curriculum/self-paced learning, deep networks are first trained on "easy" samples, then after relatively "hard" samples have been given as input to the net-

work. Furthermore, dual learning learns the bi-directional mapping function, such as English to French and French to English. In this method, primal and dual tasks are defined based on the labels present in the data. In the first task, unlabeled English data can be translated into pseudo-French. In dual-task, pseudo-French can be translated into pseudo-English. The reward has been obtained by reinforcement by applying the ground truth of an English word and obtaining a pseudo-English label. This technique is quite useful for Visual Question Answering (VQA) [145]. In data programming, the labeled training set can be generated pragmatically [202]. By utilizing this idea, Wu *et al.* [276] have proposed to provide weak supervision of domain expertise for extracting information from richly formatted data. Learning a mapping function has also been utilized to solve zero/few-shot problems [8]. The mapping learns generalized representation from the training set to correctly classify new concepts (novel classes) from the testing set. Classifying the novel classes at the inference is termed as Zero-Shot Learning (ZSL) or Generalized Zero-Shot Learning (GZSL).

4. **Cross-domain synthesis–** Srivastava and Salakhutdinov [234] have proposed the cross-domain synthesis while utilizing the other domain data to generate synthesized for the target domain, which has less amount of training data. The cross-domain synthesis method is widely used in medical imaging analysis such as cross-MRI image synthesis [29, 111], MR to CT image synthesis [23, 248], and so on.

5. **Data transportation–** In the domain adaptation, the target problem is solved by importing the solution generated by the source domain, which has a sufficiently large amount of data [192]. In initial works, instance re-weighting based on the likelihood and MMD estimation has been used [17, 293]. Recently, Liang *et al.* [146] have utilized the domain irrelevant information to decrease the intra-class distance in both domains. While learning the transformation from the source to the target domain, GAN methods are also utilized for data transportation from the source to the target domain [18, 260].

### 1.2.2 Data Fine-Tuning (DFT)

Pre-trained models are widely used for ML tasks ranging from face recognition to object classification and segmentation. However, these pre-trained networks generally do not perform well if the

target database differs from the source (pre-training) database. For example, a CNN pre-trained on VGGFace2 database [22] may not yield good results when tested with ImageNet database [47]. Two methods are widely used for fine-tuning pre-trained CNN models: (i) freezing the convolutional layers and training the dense layers added after the convolutional layers and (ii) re-training few convolutional layers along with the dense layers by updating weights while leaving the other convolutional layers frozen. However, the number of trainable parameters in these methods is significant, especially for deeper models such as ResNet-150 [91] and DenseNet-201 [103]. Chhabra *et al.* [35] recently have proposed Data Fine-Tuning (DFT) which leverages the input space to enhance the performance of the pre-trained CNN models. In this technique, input data (target database) is "adjusted" corresponding to the pre-trained model's unseen decision boundary. To illustrate this concept, let $\phi$ be the pre-trained model with weight $\mathbf{W}$ and bias $b$. DFT can be represented as,

$$\phi(\mathbf{WX} + b) \xrightarrow{\text{DFT}} \phi(\mathbf{WZ} + b) \tag{1.3}$$

where, $\mathbf{Z}$ represents the updated dataset. a uniform perturbation is learned corresponding to each dataset to adjust the input data. Comparing it to model fine-tuning, where parameters $\mathbf{W}$ and $b$ get updated, whereas in DFT, input data $\mathbf{X}$ is updated. It is important to note that the number of trainable parameters in DFT is the same as the size of the input image, which is significantly less than the number of trainable parameters in deep learning models.

## 1.3 Optimizing Model Space for DCL Problems

Large training samples play an important role in successfully training deep architectures, which have millions of learning parameters [91, 103, 240]. The parameters of the deep neural network can be represented as $\theta$, $x$ is input, and $y$ is ground truth for $x$. Mathematically, learning of a network can be defined as:

$$J(\theta) = \underbrace{\frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})}_{\text{Model Specific}} + \underbrace{R(\theta)}_{\text{Domain Specific}} \tag{1.4}$$

where, $m$ is the number of training samples, $h_\theta$ is the hypothesis $h_\theta(x) \in [0, 1]$, and $R$ is the regularizer constrained on $\theta$ while learning.

### 1.3.1 Adapting Pre-trained Models

In the literature, adaptation of the model has been achieved by fine-tuning, distillation, and model adaptation of the pre-trained model. Generally, large datasets can be utilized for pre-training the models. In *fine-tuning*, models pre-trained for a similar task can be re-trained on data-constrained settings (target dataset) [94, 127, 290]. Hinton and Salakhutdinov [94] suggested simultaneously training the Stacked Auto-Encoder (SAE) end to end the whole model. Therefore, the overfitting and vanishing gradient problem can be minimized. Inspired by SAE, [15] have utilized Stacked Denoising Auto-Encoder (SDAE) for newborn face recognition, which has a small number of training samples. Stack-wise learning can also be utilized in the dictionary learning paradigm. Tariyal *et al.* [242] have shown that popular deep learning models can be designed with the help of a dictionary and, hence, can be used in DCL. In CNN [290], the concept of fine-tuning the last few layers works because of learning of common feature extractor at the initial layer of CNN model, irrespective of databases. The initial layer of the CNN model learns Gaussian-type filters to extract the edge and blob-based information. After the initial layers, subsequent layers learn a complex feature extractor, which can provide the abstract view of the object [37, 205].

1. **Fine-Tuning–** In this method, a knowledge base in the form of a pre-trained model trained on a similar task is re-trained on small sample data (target dataset) [94, 127, 290]. Fine-tuning a few layers of the CNN model works because of standard feature extractors (filters of the convolutional layers) from the objects. The CNN model's initial layer learns Gaussian-type filters to extract the edge and blob-based information. After the initial layer, subsequent layers learn a complex feature extractor, which can provide the abstract view of the object [37, 205].

   In a recent couple of years, conventional fine-tuning has been changed to improve the performance of the network, i.e., progressive network [213], block-module network [244], developmental network [268], and structure and strength learning [118]. These networks are

trained on multiple tasks to have a more generalized network or divide all the learning parameters into modules and train the network modularly.

2. **Distillation–** This is similar to the knowledge transfer method [92]. Distillation has a teacher-student network, where the student network is trained while receiving feedback from the teacher network. The feedback has been introduced by reducing the cross-entropy between the teacher network's softened output and the student network's output. Romero *et al.* [209] have proposed utilizing the softened output and the intermediate representation of the teacher network. Similarly, Yim *et al.* [289] has attempted to manage the flow between network layers by computing the Gram matrix of both consecutive layers. Radosavovic *et al.* [198] have proposed Omni-supervision for the distillation of the network. This method uses unlabeled data to generate new training data while predicting multiple transform data from a pre-trained model. Luo *et al.* [162] have proposed graph-based distillation of the network for partially observed modalities. Shmelkov *et al.* [224] has proposed reducing the source data's catastrophic forgetting while training on the target data.

3. **Model adaptation–** In this type of domain adaptation, unlike the data transportation, the model's decision boundary is adapted to the target dataset, which has fewer observations. To do the model adaptation in SVM, domain transfer [54], cross-domain [110], residual transfer [159], and adaptive multiple kernel learning [55] have been proposed.

   In the case of deep models, fine-tuning might only work when the source and target datasets are related. Therefore, Tzeng *et al.* [253] have utilized two losses (softmax cross-entropy loss and domain confusion loss) to train the network on the target dataset. Moreover, in place of confusion loss, Long *et al.* [156] have used MMD loss on fully connected layers' of AlexNet [127]. Sener *et al.* [222] have proposed to optimize the target label inference along with in-depth feature representation in the deep network such that domain transfer on small sample data can be done. In *model adaptation*, unlike the data transportation, the decision boundary of the model is adapted on the target dataset, which has less number of observations. To do the model adaptation in SVM, domain transfer [54], cross-domain [110], residual transfer [159], and adaptive multiple kernel learning [55] have been proposed. Tzeng *et al.* [253] have utilized two losses (softmax cross-entropy loss and domain confusion loss) to

train the network on the target dataset. Moreover, in place of confusion loss, Long *et al.* [156] have used MMD loss on fully connected layers' of AlexNet [127]. Sener *et al.* [222] have proposed to optimize the target label inference along with features in the deep network such that domain transfer on small sample data can be done.

Recently, researchers have proposed several techniques which outperform the performance of conventional model fine-tuning. This includes progressive network [213], block-module network [244], utilizing intermediate information of the CNN blocks [132], class-based penalty at each convolutional layer [227], and collaborative learning [77].

### 1.3.2 Reduce the Dependency of Large Sample Learning

In general, the learning mechanism of machine learning models follows Equation 1.4, which can be seen as a combination of 1) model-specific learning and 2) domain-specific regularization. The multi-attention framework proposed by Huynh and Elhamifar [106] can be considered model-specific learning. In this work, they have introduced a shared multi-attention framework for multi-label zero-shot learning. They have demonstrated that having an attention mechanism for recognizing multiple seen/unseen labels is a complex task. Hence, instead of generating attention for novel classes, they have let the novel classes select from a set of shared attention. Generally, regularizers are considered domain-specific knowledge used to improve the network's training. In this school of thought, Tenenbaum *et al.* [243] have suggested that strong prior makes a difference in making inferences beyond the data availability. This prior can be side information [256], domain knowledge [192], and common sense [46]. In the literature, dropout [233], drop-connect [262], batchnorm [107], and class-based sparsity [16, 80, 217, 218] have been proposed to reduce the dependency on Large Scale Learning by reducing overfitting.

## 1.4 Optimizing Feature Space for DCL Problems

In the last decade, advancements in deep learning algorithms have enabled the utilization of several real-world applications ranging from face and gesture recognition to autonomous vehicles and drones. Face recognition from surveillance cameras is essential to ensure public safety and avoid

instances of terrorist attacks and intrusion. One of the primary reasons for the advancement of such applications is the advent of novel and effective loss functions focused primarily on the feature space. These loss functions update the model's parameters to produce feature-rich representations in the embedding space of the model. The most prolific of these are loss functions formulated using Deep Metric Learning (DML), which allow us to train discriminative classifiers even on databases with insufficient training samples. This section highlights DML as an effective technique for training discriminative models for $S^3$ problems.

### 1.4.1 Deep Metric Learning (DML)

Conventional deep learning models like CNNs and Recurrent Neural Networks (RNN) are trained with data samples and their corresponding labels to correctly predict the class/label of an input sample during testing. However, deep metric learning algorithms train a model to distinguish between a pair of data samples and whether they belong to the same class/category or not. During training, a vanilla deep metric learning loss function would update the weights of the model so that it produces embeddings/features (unlike class labels in conventional deep learning models) of data samples that belong to the same class close to each other and that of different classes away from each other in the output embedding space of the model. To train such a model, we need large quantities of data samples during training. Since a discriminative model is being trained, it may be tested/evaluated on classes that the model does not encounter during training. This flexibility makes deep metric learning models popular for building real-world recognition systems.

The most seminal work in deep metric learning was by Hadsel *et al.* [84] where the contrastive loss was proposed. It optimized a Siamese network for matching a pair of images using the same optimization goal as illustrated above. Thereafter, several research works [164, 237, 307] have utilized this optimization technique using a CNN network as the backbone model, before a new loss function, known as the triplet loss was proposed by [221]. This was extended in 2017 by a recent loss function known as the quadruplet loss [33]. An N-pair loss metric [232] is also proposed, which uses an N-tuple as a training data sample. Further, different variants of these loss functions are proposed for DCL scenarios.

### 1.4.2 DML for Small Sample Learning

To address one-shot and few-shot learning scenarios, several deep metric learning algorithms have been proposed for small-sample learning. Vinyals *et al.* [259] have proposed an algorithm that is used to train the model in episodic cycles. In each training cycle or episode, a few training examples are selected to learn embeddings for predicting the class of these samples. The purpose of this episodic training strategy is to mimic the real environment where only a few samples would be available. Several other one-shot and few-shot based DML approaches [5, 231, 238] have also been proposed to address small sample training scenarios. Recently, a density-aware deep metric learning algorithm has been proposed [75] where results are shown on a surveillance face dataset (SCface [82]), which has a minimal number of training samples. This algorithm has a mechanism of avoiding outliers and noisy training data, which can hinder the learning process, especially in DCL scenarios.

### 1.4.3 Sample Mining in DML

Train deep networks using loss functions such as the triplet loss requires preparing triplets (or 3-tuples) using the data available for training. Given $N$ training classes and $K$ samples for each class, the total number of triplets that can be prepared for training is upper bounded by $N(N-1)K^2(K-1)$. Therefore, the number of training samples (each triplet is treated as a training sample) increases from $O(N.K)$ (available for conventional deep learning algorithms) to $O(N^2.K^3)$, which is a considerable sample space. This space would be even more significant for quadruplets or N-pair loss functions. This increased sample space is beneficial for learning a model with a DML algorithm on a database with a small number of data samples. It also makes DML algorithms a natural choice where the amount of training data is insufficient to learn a conventional classifier.

This enormous input sample space on large databases may hinder efficient learning for several reasons. One of the reasons is that, after several epochs of training, the model would have learned to solve most of the data samples, each of which is a triplet/quadruplet. Thus, fewer samples would be helpful for the model to continue learning and make significant weight updates. During this phase, it is required to provide only valuable triplets/quadruplets; in other words, mine those triplets/quadruplets that are hard (which are still not correctly classified by the model) to continue

learning the model. This technique, known as hard mining, has been extensively explored in the last few years for DML methods [87, 187, 225, 236, 292].

### 1.4.4 Adversarial Deep Metric Learning

Recently, a new way of applying DML algorithms to small databases has been through deep adversarial metric learning. This is proposed by Duan *et al.* [56], where synthetic data samples are generated. As illustrated above, hard-mining approaches mine hard triplets/quadruplets from the existing pool of training data. However, at some point in time, the pool of training data would be exhausted, especially for small databases. This technique generates new synthetic samples from existing data samples using an adversarial loss function generator. Unique triplets/quadruplets can be generated from these synthetic examples, most of which are hard samples. Keshari *et al.* [117] have utilized metric learning in ZSL/GZSL setting as well. On the generated over-complete distribution, they proposed a framework that utilizes Online Batch Triplet Loss (OBTL) and Center Loss (CL) to enforce the separability between classes and reduce the class scatter. Another technique with a similar objective is Energy Confused Adversarial Metric Learning [56], where synthetic samples are generated using an energy confusion regularization term to confuse the DML model. This energy confusion term is trained together with the conventional metric objective in an adversarial manner. Recently, an adaptive margin-based approach [271] has also been proposed for the same.

## 1.5 Research Contributions

As previously mentioned, to successfully train deep architectures with millions of learning parameters, large training datasets are essential [91, 103, 240]. The learning parameters in a deep neural network encapsulate model and domain-specific information, as illustrated by Equation 1.4. Here, $R$ is a regularizer applied to the parameters $\theta$ during the learning process. In scenarios where training samples are sparse, there is a tendency for the deep neural network to overfit the limited examples available in the training set. The issue of Data-Constrained Learning (DCL) is a prominent challenge in deep learning, and its significance is growing with the stagnating performance improvement in deep models. Our research introduces four innovative solutions to mitigate the

limitations associated with learning from a data-constrained environment.

Figure 1-5 succinctly captures our contributions through a visual representation of the typical deep learning model pipeline. A deep learning pipeline can be conceptualized as a synergy of three distinct spaces: 1) input, 2) model, and 3) feature space. In addressing the Data-Constrained Learning (DCL) challenge, our approach uniquely leverages the model space, a less frequented avenue, as depicted in Figure 1-6. Additionally, we have employed both the input and feature spaces to ascertain a generalized mapping function that facilitates the transformation from visual to semantic feature representations.

- *Learning Structure and Strength of CNN Filters for Small Sample Size Training:* Traditionally, pre-trained deep models have been utilized for DCL problems while finetuning only the deep model's last layer on a small set. This can be categorized into model space learning, where experience comes from the pre-trained models. In this research, the structure of the CNN filters is taken from pre-trained models or dictionaries. After initializing the learned filters in the CNN framework, the filters' parameters are frozen, and only the strength of a filter has been trained. The experiments support our hypothesis that learning only the strength of filters can reduce the total learning parameters. So, the model's performance on small sample data can improve.

- *Guided Dropout:* In this research, another variant of dropout called as *"Guided Dropout"* has been proposed. It can be categorized into model space learning in the subcategory of reducing the dependency of DCL. Guided dropout uses strength-based guidance of dropping nodes in the training phase to utilize the neural network's full potential. This new dropout method ensures more penalty than a conventional dropout. Higher strength represents the higher importance of the node in Deep Neural Network (DNN). Guiding dropping criteria leads the network to learn generalization even if samples are fewer.

- *Over-complete distribution generation:* The above three methods utilize model space learning. In this research, ZSL and GZSL problems have been tackled while generating an over-complete distribution. This method can be categorized in feature space and input space learning, where new concepts in the form of novel classes that occurred in the testing set

must be classified correctly. Our proposed method utilizes the latent space of the Conditional Variational Autoencoder (CVAE) and generates more hard samples. The generated distribution then trains the regressor while optimizing inter and intraclass distance. Hence, predicting the distribution of unseen classes and improving the separability and compactness of the distribution have shown improvement in the model's generalization. Center loss and batch triplet loss have been used to improve compactness and separability, respectively.

- *Filter Augmentation:* In this research, augmentation methods of CNN filters have been proposed. Given that the presented approach employs the model space, it falls under the category of model space learning. The total quantity of learning parameters remains consistent with traditional finetuning methodologies. A filter augmentation module is activated across all convolutional layers of the CNN model, incorporating operations like rotation, flipping (vertical, horizontal, and channel flipping), and guided DropBlock. Additionally, we have illustrated how certain existing regularization techniques directly or indirectly connect to the filter augmentation method.

# Chapter 2

# Learning Structure and Strength of CNN Filters for Small Sample Size Training

*"The higher your structure is to be, the deeper must be its foundation."*

— Saint Augustine, 430AD

*"Success is achieved by developing our strengths."*

— Marilyn vos Savant, 1986

## 2.1 Introduction

CNN is a multilayer representation learning architecture that has received immense success in multiple applications such as object classification, image segmentation, and natural language processing. From LeNet [137] to AlexNet [127], GoogleNet [240], VGG-Net [230], ResNet [91], and now DenseNet [103], given large training data, CNNs have shown state-of-the-art performance for several applications. However, large training data is also a limiting requirement for applications with small sample sizes, and many of these architectures easily overfit on small training samples. For example, as shown in Figure 2-1, a face recognition model trained on large training data of adult faces (e.g., CelebA or LFW databases) may not provide good performance when tested for newborn face recognition [14, 15]. In newborn face recognition, the available training data may be small, and therefore, even after fine-tuning, standard deep learning-based face

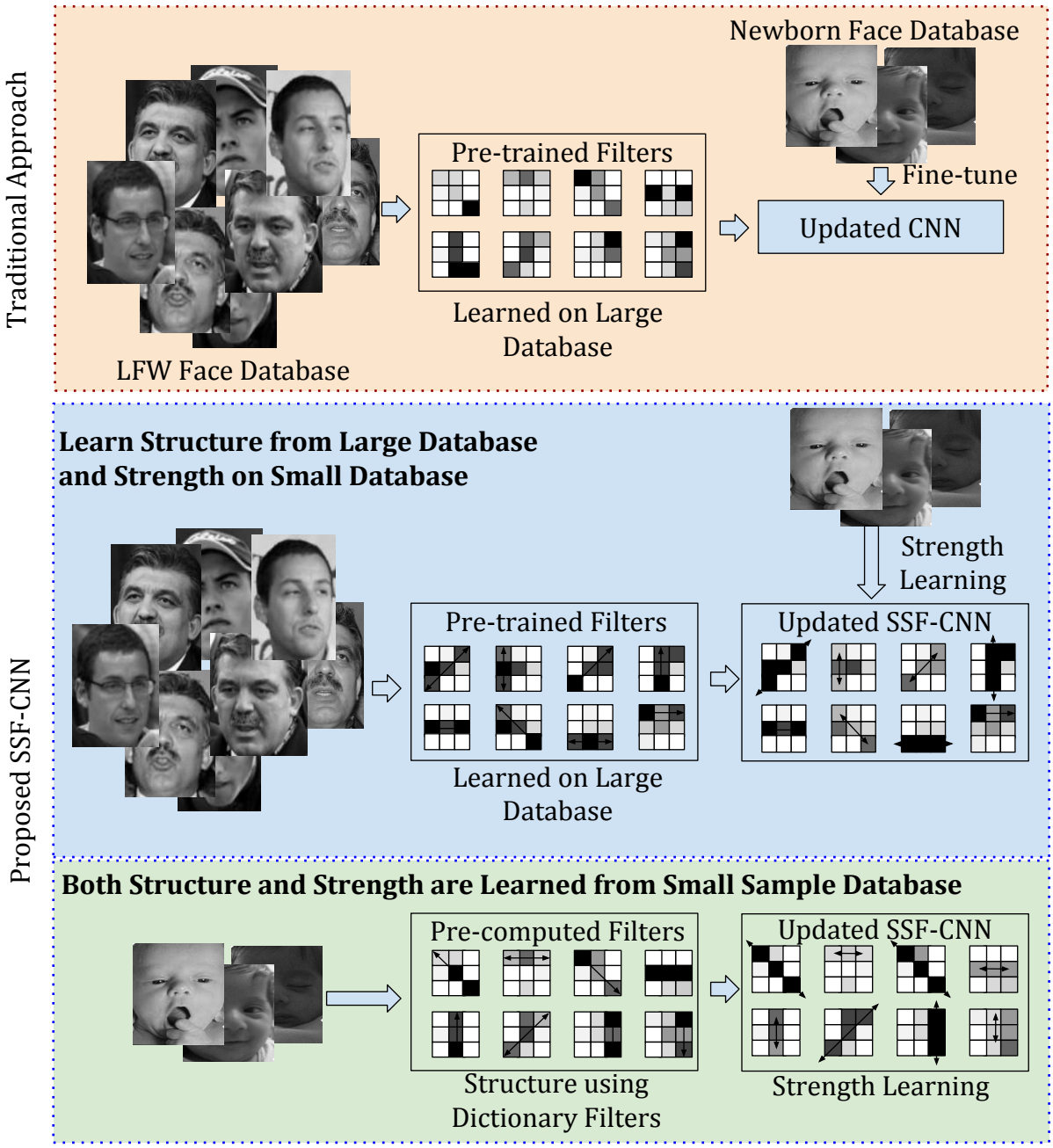



Figure 2-1: Face recognition models trained on adult face images may not provide good performance for newborn face recognition. Structure and Strength Filtered (SSF)-CNN proposes to learn structure and strength of the filters for improving the classification performance for small sample databases.

recognition models may not yield high performance. Recently, new methods called dataset distillation [26, 182, 183, 305] utilized to condense the training set into a few samples to reduce the training time without compromising testing set performance.

To address the challenge of small sample size, researchers have proposed algorithms focusing on CNN initialization tricks and modifications to CNN architecture. Erhan *et al.* [59] have investigated the importance of unsupervised pre-training of deep architecture and empirically shown that pre-trained weights of the network generalize better than randomly initialized weights. Similarly, Mishkin and Matas [174] have proposed Layer-Sequential Unit-Variance (LSUV) initialization that utilizes the orthonormal matrices to initialize the weights of each convolutional layer and normalize the weight to the unit variance. Along the same lines, pre-defined handcrafted filters are also proposed to handle the small sample size problem. For example, Andén and Mallat [7] propose a Scattering network (ScatNet), which is a CNN like architecture where pre-defined Morlet filter bank is utilized to extract features. However, these handcrafted filters may not represent the true distribution of the data and hence extract not-so-meaningful features. To overcome this limitation, Oyallon *et. al.* [190] have proposed a hybrid network, where they have utilized the ScatNet feature followed by CNN architecture. Similarly, Chan *et. al.* [27] propose PCANet architecture that utilizes Principal Component Analysis (PCA) to learn the filter banks. They also present an extension, termed LDANet, in which the selection of the cascade filters is trained from Linear Discriminant Analysis (LDA). Gan *et al.* [69] propose a PCA-based Convolutional Network (PCN) which influences both CNN [108] and PCANet [27]. Dan *et al.* [275] utilize the concept of kernel PCA to further improve the PCANet architecture. Zeng *et al.* [295] propose a multilinear discriminant analysis network (MLDANet) which is a variant of PCANet and LDANet. Feng *et al.* [63] propose a Discriminative Locality Alignment Network (DLANet), which is based on manifold learning. These architectures learn filters in a stack-wise manner, and once the network (filters) is trained, generally, it is not allowed to fine-tune the filters on other databases.

In other research directions for small sample size training, Mao *et al.* [169] propose a neural network learning method based on posterior probability (PPNN) to improve the accuracy. Ngiam *et al.* [181] propose tied weights in a filter using a tiling parameter, which handles the total number of learning parameters. In another work, Indian Buffet Process (IBP) priors are utilized to propose semi-supervised ibpCNN, which shows better generalizability [60]. Xiong *et al.* [285] present Structured Decorrelation Constrained (SDC) for hidden layers. The authors have also proposed a novel approach termed Regularized Convolutional Layers (Reg-Conv) that can help SDC to regularize the complex convolutional layers. Similarly, Cogswell *et al.* [38] propose DeConv loss

for CNN architecture that helps in training small databases.

As mentioned previously, one of the major problems with adapting pre-trained CNN models for small sample size problems is a large number of parameters; therefore, insufficient training samples may cause overfitting. If we reduce these parameters to a significantly small number, the problem can be addressed better. This paper focuses on two novel ways to develop CNN based feature representation algorithm for small sample size problems: (i) associating "strength" parameter to control the effect of each pre-trained filter, and (ii) utilizing a generalizable approach that pre-learns the "structure" of the filters using small training samples. The proposed architecture is motivated by ScatNet, but in place of pre-defined filters, we utilize a dictionary learning model to *pre-learn* the filters. Further, unlike CNN approaches where we update the weights in every iteration, we introduce the strength of the filter and update only the strength parameter, not the filters. Introducing "*strength*" of filters significantly reduces the number of parameters to learn (detailed calculations shown later) and, therefore, avoids overfitting with limited training. Experiments are performed on object classification databases, MNIST [138], CIFAR-10 [126], NORB [140], Omniglot [130], and a challenging small sample size database of newborn faces [15]. Comparison with existing algorithms shows that the proposed approach achieves state-of-the-art performance for small sample size problems and significantly reduces the number of parameters to learn/fine-tune.

## 2.2 Proposed SSF-CNN

Learning the entire network from scratch while training with small databases is challenging. Existing approaches with pre-defined or handcrafted filters [7], and pre-trained filters [27, 69, 295], may not allow fine-tuning the filters and therefore, the learned model may not represent the true data distribution for minor sample size problems. To mitigate these challenges, we propose a novel approach, termed as SSF-CNN, which has two components: (i) structure of the filter and (ii) strength of the filter. Our hypothesis is that the structure of the CNN filters can be learned from domain-specific larger databases or other representation learning paradigms that require less training data, such as dictionary learning [242, 249]. It is well known that matrix factorization or dictionary learning allows us to learn the *dictionary* that helps encode the representative fea-

**Algorithm 1** Hierarchical Dictionary Filter Learning

**Notation:** $N$ is a number of training samples, $n$ number of extracted patches, $y$ is a patch from $Y$
**Input:** $X_N$
**Output:** $D$
    **for** *each layer* $l := 1$ *to* $numLayer$ **do**
    **end**
$[x^n]^N \leftarrow extractPatch(X_N)$
Y $\leftarrow$ reshape($[x^n]^N$)
$\min_{D \in \mathbb{R}^{m \times k}} \frac{1}{n} \sum_{i=1}^{n} \min_{\alpha^i} (\frac{1}{2}||y^i - D^l \alpha^i||_2^2 + \lambda ||\alpha^i||_1)$
W $\leftarrow$ reshape($D^l$)
    **for** $j := 1$ *to* $N$ **do**
    **end**
$fmap_j = X_j * W$

$X_N \leftarrow ReLu(fmap)$

tures. If we represent CNN filters using a dictionary, it can provide the "structure"; however, it may not be well optimized for the classification task. Therefore, the next part of the framework is computing the "strength" of every filter to adapt the weights of these filters according to the data characteristics. Strength can be interpreted as the attuning parameter to update or adapt the filters based on the small-size training data. For illustration, columns (a) to (d) in Figure 2-2 represent the samples from trained dictionary filters for the MNIST database, and columns (e) to (h) represent the updated filters where changes are due to the strength parameter.

In the proposed approach, the hierarchical dictionary filters are first learned to initialize the CNN, followed by the strength parameter to train the CNN model. We introduce strength parameter '$\mathbf{t}$' for the CNN filters '$\mathbf{W}$', which allows the network to assign *weight* for each filter based on its structural importance. In CNN model, strength and structural parameters $\mathbf{t}$ and $\mathbf{W}$ can be learned in two ways: 1) pre-train $\mathbf{W}$, use it in CNN by freezing the values of $\mathbf{W}$ followed by learning the strength $\mathbf{t}$, and 2) pre-train $\mathbf{W}$ which is used to initialize the CNN model followed by learning $\mathbf{t}$ and $\mathbf{W}$ iteratively. While the second approach, which simultaneously learns structure and strength, may be desirable, the first approach requires very few parameters to be trained in CNN model. We next describe the approach to learn $\mathbf{W}$ hierarchically, filters of CNN model, using dictionary learning followed by learning the strength parameter $\mathbf{t}$.

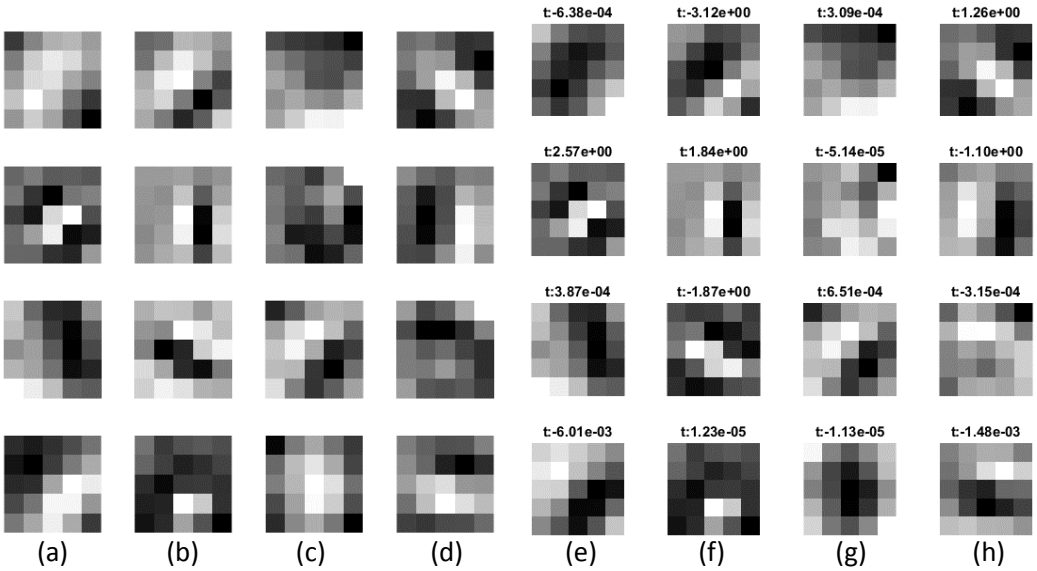



Figure 2-2: Filters (a) to (d) are dictionary-trained filters. Filters (e) to (h) illustrate the change due to the proposed strength parameter in CNN architecture. These filters are trained on the MNIST database.

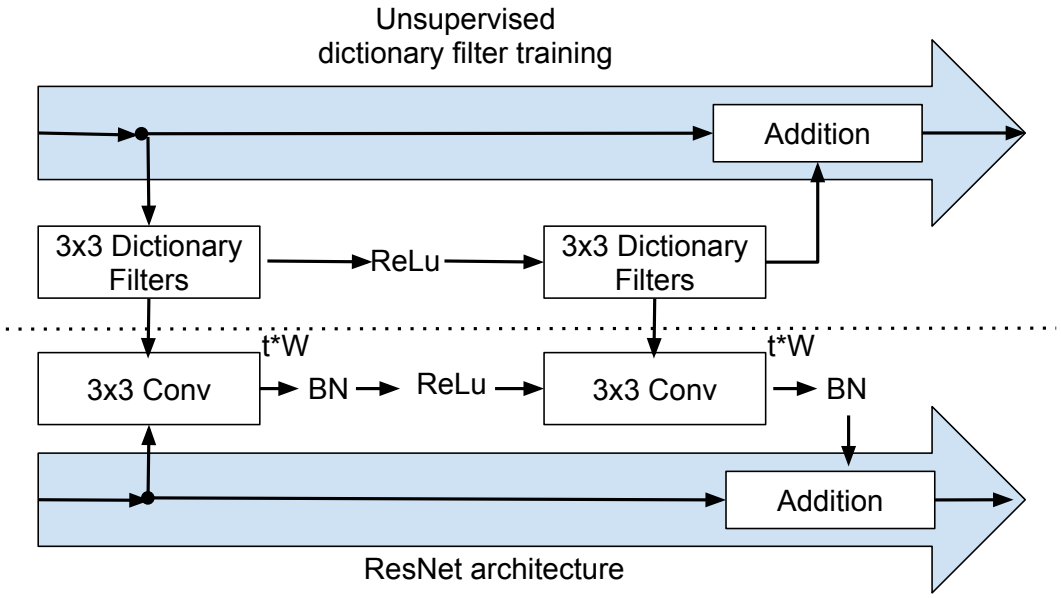



Figure 2-3: The proposed SSF-CNN architecture for initializing the ResNet architecture with the filters learned from the dictionary.

### 2.2.1 Learning Structure of Filters

In this research, we propose using a dictionary-learning algorithm to learn the filters' structure. The algorithm can be divided into two steps: 1) learn hierarchical dictionary filters and utilize trained dictionary filters to initialize the CNN, and (2) train CNN with dictionary-initialized filters.

**Hierarchical Dictionary Filter Learning**: Dictionary learning focuses on learning a sparse representation of the input data in the form of a linear combination of basic elements or atoms [58, 142, 166, 242, 249]. For a given input $\mathbf{Y}$, a dictionary $\mathbf{D}$ is learned along with the coefficients $\alpha$:

$$\min_{\mathbf{D},\alpha} \|\mathbf{Y} - \mathbf{D}\alpha\|_F^2, \; such\ that \; \|\alpha\|_0 \leq \tau \tag{2.1}$$

where, the $\ell_0$-norm imposes a sparsity constraint on the learned coefficients and $\tau$ corresponds to the maximum number of non-zero elements. Often, the $\ell_0$-norm is relaxed, and the updated dictionary learning formulation can be written as:

$$\min_{\mathbf{D},\alpha} \|\mathbf{Y} - \mathbf{D}\alpha\|_F^2 + \lambda||\alpha||_1 \tag{2.2}$$

where, $\lambda$ is a regularization parameter which controls the sparsity promoting $\ell_1$-norm. In this research, we utilize dictionary learning to pre-train the filters of CNN in a hierarchical manner. As shown in Algorithm 1, a hierarchical dictionary learning technique is utilized to initialize the CNN model (ResNet [91]). The trained dictionary atoms are used to convolve over the input image. After convolution, feature maps are normalized according to the activation function (e.g., ReLu) used in CNN models. Figure 2-3 presents the structure of a block of the SSF-ResNet architecture. The extracted feature map is an input for the next level of the hierarchical dictionary. In this manner, the number of dictionary layers is the same as the number of convolutional layers in CNN models. In Algorithm 1 $extractPatch$ function is used to tessellate the input image into small patches. The trained dictionary is organized in a two-dimensional array where each filter is arranged in one column. These learned filters are reshaped and convolved over the input image to produce the feature maps for the next dictionary level.

**Training CNN with Dictionary Initialized Filters**: Typically, CNN has multiple convolutional layers, each layer has multiple filters, and these filters are trained using Stochastic Gradient Descent

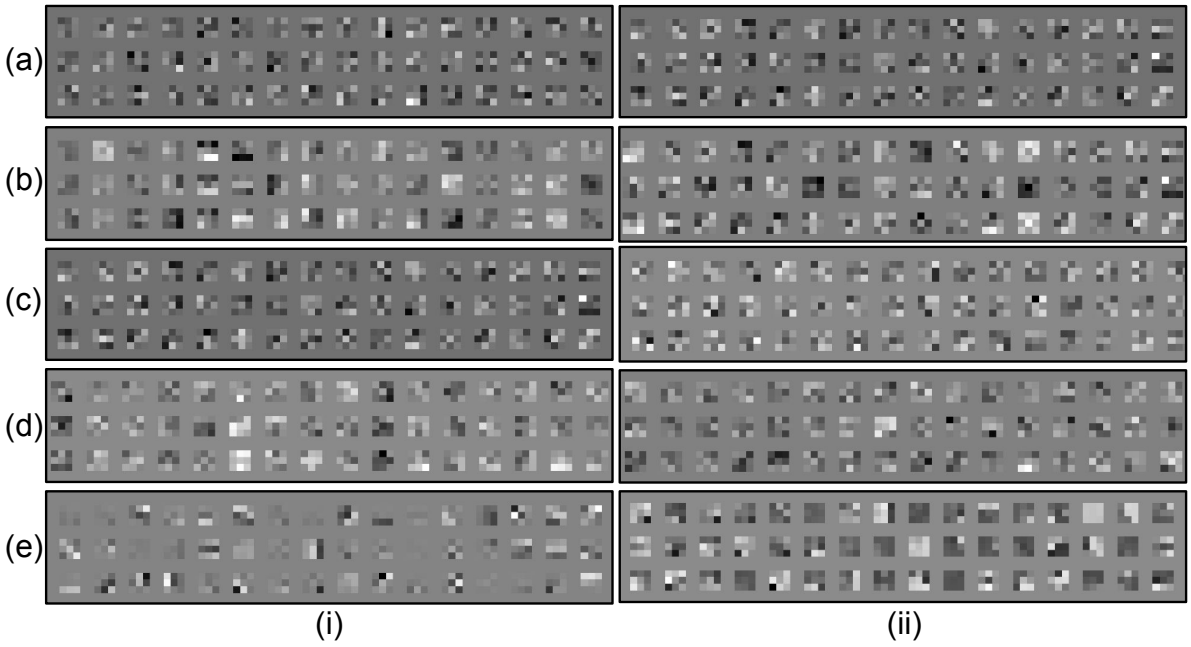



Figure 2-4: Filter visualization of the (i) $1^{st}$ layer and (ii) $2^{nd}$ layer of the ResNet architecture on the CIFAR10 dataset. (a) Xavier [76] initialized filters at zero epoch, (b) Xavier [76] initialized filters are trained on 1000 training samples, (c) MSRA [90] initialized filters at zero epoch, (d) MSRA [90] initialized filters are trained on 1000 training samples, and (e) Dictionary initialized filters at zero epoch. For better visualization, only 16 filters are used from the $2^{nd}$ layer.

(SGD) [141]. For input $\mathbf{X}$ and convolutional filter $\mathbf{W}$, the convolutional function of the CNN can be defined as $f(\mathbf{X}, \mathbf{W}, b) = \mathbf{X} * \mathbf{W} + b$, where $*$ is the convolutional operation and $b$ is the bias. A CNN architecture is designed by stacking multiple convolutional and pooling layers. These deep CNN architectures are trained in two passes: 1) forward pass and 2) backward pass. The network propagates the input signal to the last classification layer in the forward pass. In the backward pass, the error $\delta_j^l$ for each layer $l$ on node $j$ is computed with respect to the cost, and the weights of the CNN filters are updated accordingly.

Let $\mathbf{a}^l$ be the output feature map at $l^{th}$ layer of the CNN with a cost function $C$. The weights are updated as per the gradient direction, i.e., $\Delta \mathbf{W}^l = \frac{\partial C}{\partial \mathbf{W}^l}$. Using chain rule, $\Delta \mathbf{W}^l = \mathbf{a}^{l-1}\delta^l$. In traditional CNN learning, the weights are initialized in different ways such as Xavier [76], or MSRA [90] approach and even randomly. In this research, we propose initializing the CNN filters using dictionary-learned filters as discussed above. Figure 2-4 shows filters learned from the dictionary learning technique show more "structure" than traditional approaches, particularly with small training data. While dictionary initialization helps find improved features, traditionally,

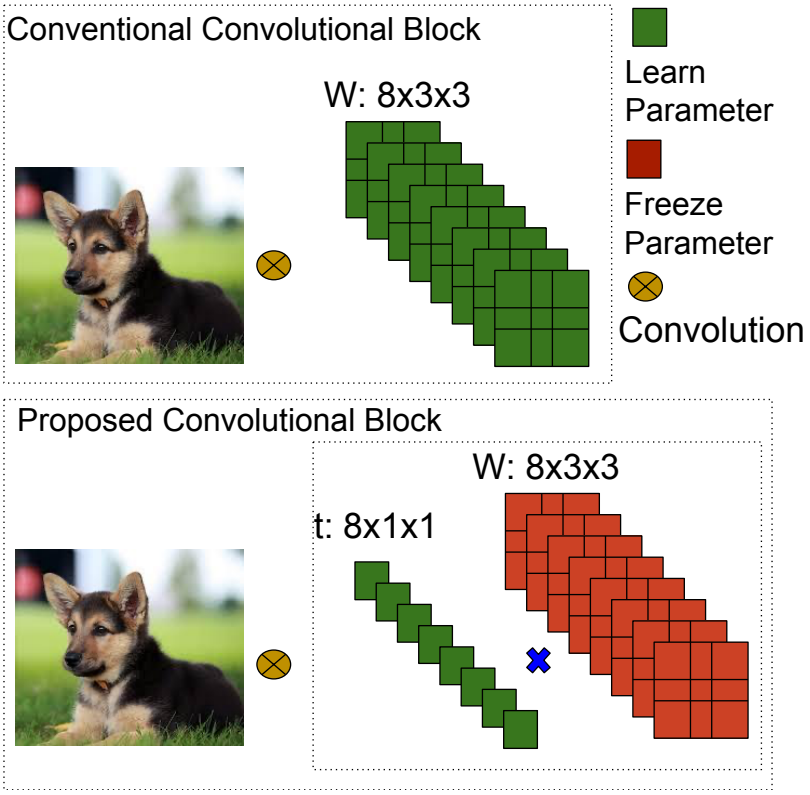



Figure 2-5: Illustrating the concept of learning the strength of a filter, which significantly reduces the number of training parameters.

updating the filters requires large parameter space, which is unsuitable for small training data. In the next subsection, we present the proposed approach of incorporating the strength of the filters and not updating the filters using SGD, which significantly reduces the number of learning parameters.

### 2.2.2 Learning Filter Strength

The proposed concept of learning strength of the filter is illustrated in Figure 2-5. Here, we freeze the values of filters obtained from the dictionary learning technique and update only the strength of the filter. As shown in Figure 2-5, this significantly reduces the number of learning parameters. For $l^{th}$ layer, the strength parameter '$\mathbf{t}^l$' is learned using the stochastic gradient descent method; i.e., a scalar value is learned rather than learning the complete filter. The proposed process can be

written as,

$$f(\mathbf{X}, \mathbf{W}, b, \mathbf{t}) = \mathbf{X} * (\mathbf{t} \odot \mathbf{W}) + b \tag{2.3}$$

where, $(\mathbf{t} \odot \mathbf{W})$ represents element-wise multiplication. The pre-trained filters learned from dictionary learning or pre-trained models are selected, and the only variable to be learned is $\mathbf{t}$, which can be learned using SGD. Since $|\mathbf{W}| >> |\mathbf{t}|$, even small training data can be used to train the network. In literature, various regularization techniques have been utilized for better convergence. Existing regularization techniques such as dropconnect and $\ell_1$ regularization can also be used while learning $\mathbf{t}$.

## 2.3 Experimental Results

The effectiveness of the proposed algorithm is evaluated on multiple databases with state-of-the-art CNN architectures, including ResNet [91] and DenseNet [103]. The details of the experiments and results are described below.

### 2.3.1 Database and Experimental Protocol

Since the proposed architecture is for small-size training data, the experiments are performed with varying training sizes on three databases: MNIST [138], CIFAR10 [126], and NORB [140]. More specifically, as shown in Table 2.1, the experiments are performed with 14 data sizes, $100, 200, \cdots, 1000, 2000, \cdots, 5000$. The proposed algorithm is also tested with the complete/standard training set. Further, experiments are performed on an interesting and small sample size problem of newborn face recognition [15]. The newborn database has images from 96 babies, and as per the predefined protocol [15], training data consists of images from 10 newborns, and the remaining images, corresponding to 86 newborns, are used for testing (with 1, 2, 3, and 4 images per subject in the gallery). Finally, experiments are also performed on the Omniglot database [130], which comprises $1623$ handwritten characters pertaining to $50$ different alphabets. The background database has 30 alphabets, and the evaluation set has 20 alphabets. All the experiments are performed with five-fold cross-validation, and average accuracies are reported in the following subsections.

Table 2.1: Experimental protocols for MNIST, CIFAR-10 and NORB databases.

| Databases | Small Training Data | Standard Training | Standard Testing |
|---|---|---|---|
| **MNIST** | $100:100:1k; 1k:1k:5k$ | 50k | 10k |
| **CIFAR-10** | $100:100:1k; 1k:1k:5k$ | 40k | 10k |
| **NORB** | $100:100:1k; 1k:1k:5k$ | 20k | 24.3k |

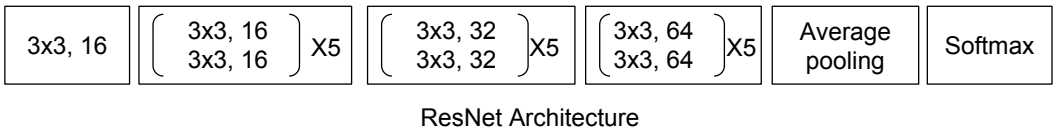



Figure 2-6: Illustrating the ResNet architecture used in the experiments.

### 2.3.2 Implementation Details

To demonstrate the results of the proposed SSF-CNN, a popular ResNet [91] architecture is used. Figure 2-6 illustrates the ResNet architecture, which has one input layer, 31 convolutional layers, one global pooling layer, and one softmax layer. The strength parameter is regularized with both ElasticNet [308] ($\lambda_1|\mathbf{t}|_1 + \lambda_2|\mathbf{t}|_2$) and DropConnect [262]. It is experimentally observed that in the first 20 epochs, $\lambda_2$ is $0.0001$ and $\lambda_1$ is $0$. After 20 epochs, both the regularization constants are set to $0.0001$. $\ell_1$ regularization introduces sparsity in $\mathbf{t}$ parameters and helps to fade out the less contributing filters, thus improving the strength of filters with large contributions. Further, at every epoch, the dropconnect parameter is randomly initialized by $Bernoulli(pr)$ where $pr$ has $0.8$ and $0.2$ probability for generating $1s$ and $0s$ respectively.

The proposed model utilizes a dictionary and pre-trained model to initialize and train the CNN filters. Specifically, dictionary filters are learned using the K-SVD algorithm 1. These dictionaries are layered in a similar manner as CNN layers and are referred to as hierarchical dictionaries. The parameter values for K-SVD, such as the sparsity parameter, the total number of iterations, and batch size for the dictionary, have been initialized with $0.1$, $1000$, and $100$, respectively. The input signal for the dictionary is patches extracted from randomly selected $N$ number of balanced training samples. The value of $N$ varies from $100, 200, \cdots, 1000, 2000, \cdots, 5000$, as shown in Table 2.1.

### 2.3.3 Parameter Learning

In traditional ResNet architecture, the total number of parameters to be learned in convolutional layers for the CIFAR-10 dataset is $242,352$. On the other hand, in the proposed SSF-CNN, the total number of strength parameters to be learned for the same database is $26,928$. This shows that the proposed architecture reduces the total number of parameters to be learned by $1/9^{th}$ factor in each convolutional layer. Similarly, we observe the reduced number of parameters to train for other databases and architectures.

### 2.3.4 Results on Limited Training Data - MNIST, CIFAR-10, and NORB

The main focus of the proposed SSF-CNN is to learn the deep neural network models with a few training samples. Since the proposed initialization is performed using dictionary learning, we also compute the results of a shallow dictionary, which serves as the baseline for all the experiments. We have also compared the proposed algorithm with PCANet [27], Deep Hybrid Network [190], ScatNet [7], ResNet initialized with Xavier [76], and ResNet initialized with MSRA [90]. For the proposed SSF-CNN, two sets of results are computed based on the manner in which the parameters $\mathbf{W}$ and $\mathbf{t}$ are learned.

- **Experiment 1** - *Learn* $\mathbf{W}$: Initialized filters are fine-tuned while doing backpropagation.
- **Experiment 2** - *Learn* $\mathbf{t}$*, Freeze* $\mathbf{W}$: Only the strength parameter $\mathbf{t}$ is learned while the initialized filters are not updated.

**Filter Visualization:** We first analyze the filters learned from the proposed method and CNN. Figure 2-4 shows the first and second layer filters trained on CIFAR-10 database: (a) & (c) showcase filters with two existing initialization techniques in CNN architecture, (b) & (d) trained CNN filters on $1000$ training samples, and (e) trained dictionary filters on $1000$ training samples. In Figure 2-4, it can be observed that dictionary-trained filters have less noisy patterns compared to CNN trained filters on small data. In literature, Zeiler and Fergus [294] have also suggested that the filters with structural properties are good while those with noisy, correlated, and unstructured patterns are bad. This visualization illustrates that the proposed SSF-CNN utilizes good filters. We next support these assertions with experimental results.

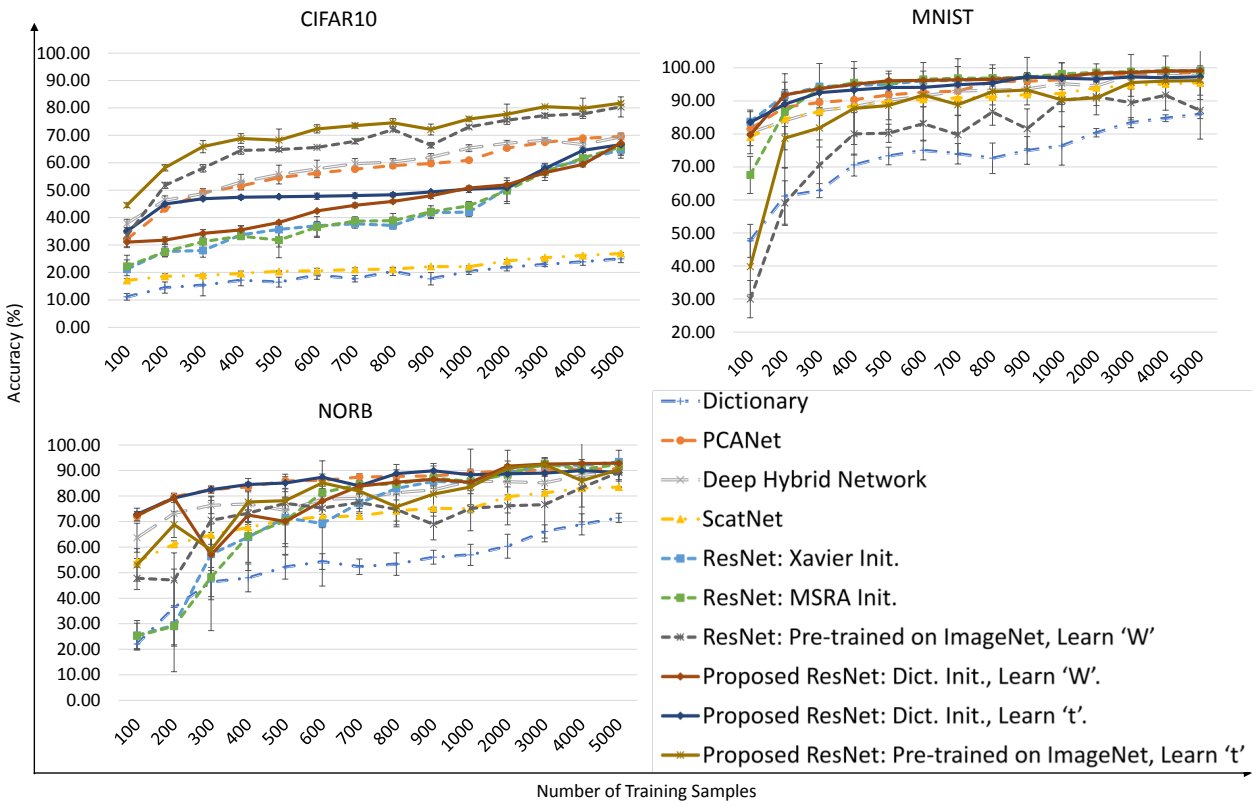



Figure 2-7: Classification accuracies (%) for CIFAR-10, MNIST, and NORB databases with varying the number of training samples.

**Performance with Shallow Dictionary:** To analyze the performance of the proposed method with varying training data sizes, 14 subsets of the training data of size $100, 200, \cdots, 1000, 2000, \cdots, 5000$, are created. These sets are used to train the dictionary and SSF-CNN on each of the three databases individually. To train a shallow dictionary for each database, 50 atoms are initialized and trained with varying numbers of training samples. The trained dictionary then computes sparse features for training and testing samples. These features are input to a 3-layer neural network with two hidden layers of size $\{40, 20\}$. The results of shallow dictionary learning on three object classification databases are reported in Figure 2-7. From these results, it can be inferred that shallow dictionary learning might not require large training data, and increasing data may not improve classification results. This figure also shows that shallow dictionary learning may not be able to yield high classification accuracy, and deep CNN architectures may further help.

**Performance with SSF-CNN and Comparison with Existing Algorithms:** We next evaluate the performance of the proposed SSF-CNN on three object classification databases by varying the

training data size. The results in Figure 2-7 show that, generally, Xavier and MSRA initialization yield lower performance than the proposed dictionary initialization for minimal training data. It can be consistently observed that the differences in results are more profound when the strength parameter $\mathbf{t}$ is learned with fixed $\mathbf{W}$. The results further show that the performance of the proposed SSF-CNN increases with an increase in training database size. It can be inferred that unlike a shallow dictionary, where the performance does not improve significantly with increased training database size, the parameters learned by the proposed SSF-CNN evolve with extensive data.

We also observe that the proposed algorithm, in general, yields higher performance compared to three existing algorithms, PCANet [27], Deep Hybrid Network [190], and ScatNet [7]. We next perform the experiments when the structure of filters is obtained from training on ImageNet data, and then the strength parameter is used to adapt to a small sample size problem (i.e., Proposed ResNet: pre-trained on ImageNet, Learn $\mathbf{t}$). Results in Figure 2-7 show that our hypothesis that the structure of filters can be learned from training on large databases and knowledge can be *adapted* with small training data using the strength $\mathbf{t}$ is valid.

After our effort on benchmarking results for small sample learning, active research is ongoing and showing promising results on the benchmarking databases [85, 283, 303] and comparing our proposed methods with current DP-SSL [301], on 4k CIFAR10 training samples our method achieved around $18.12 \pm 0.55\%$ error and DP-SSL able to reduce it further $4.23 \pm 0.20\%$. Similarly, on SVHN with 1k training samples, our method achieved $9.5 \pm 0.28\%$ and DP-SSL $1.99 \pm 0.18\%$, respectively.

**Results on Complete Training Data:** We have also evaluated the proposed dictionary learning-based initialization method on the standard training protocols of all three databases, i.e., using the complete training data. Similar to small training data size, the experiments are performed with multiple methods of initializations and two ways of learning $\mathbf{W}$ and $\mathbf{t}$, i.e., (i) *learn* $\mathbf{W}$ and (ii) *learn* $\mathbf{t}$, *freeze* $\mathbf{W}$. This experiment compares the proposed dictionary learning-based initialization for ResNet with Xavier and MSRA initialization. On the MNIST database, the proposed initialization yields an accuracy of $99.70\%$, which is comparable with $99.71\%$ achieved by standard initialization. The proposed approach yields at least $3.8\%$ higher classification accuracy on the NORB database than existing initialization approaches. It is also observed that even if the filters

Table 2.2: Rank-1 identification accuracies (%) on the newborn face database [15]. The results are reported for fine-tuned pre-trained models and with learning the strength of pre-trained filters. The last three models are trained on face databases, and the remaining models are trained on ImageNet [47] database.

| **Pre-trained Model** | **Number of Gallery Images** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Fine-tuning** | | | | **Proposed Strength Learning** | | | |
| | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| ResNet 50 | 35.77 ± 2.34 | 43.59± 0.92 | 49.90 ± 2.57 | 52.14 ± 3.31 | 37.80 ± 2.01 | 46.77± 1.79 | 52.61 ± 1.89 | 56.73 ± 1.79 |
| ResNet 101 | 35.86 ± 2.78 | 45.90± 2.54 | 51.17± 2.10 | 54.59 ± 3.41 | 36.62 ± 4.06 | 46.71 ± 3.73 | 52.79 ± 1.72 | 56.16 ± 3.07 |
| ResNet152 | 36.30 ± 3.19 | 46.74 ± 2.42 | 51.99 ± 2.24 | 55.47 ± 2.34 | 38.30 ± 3.57 | 47.92 ± 2.29 | 53.71 ± 2.62 | 59.57 ± 2.46 |
| VGG13 | 56.34 ± 2.46 | 68.49 ± 3.07 | 73.37 ± 2.53 | 76.47 ± 2.33 | 65.54 ± 3.20 | 78.14 ±1.97 | 84.05 ± 1.40 | 87.76 ±1.88 |
| VGG16 | 57.07 ± 2.85 | 67.84 ± 2.61 | 73.21± 3.10 | 76.21 ± 2.86 | 65.29 ± 1.99 | 79.18 ± 2.85 | 84.24 ± 2.82 | 87.50 ± 1.47 |
| VGG19 | 53.87 ± 4.49 | 66.95 ± 2.15 | 72.33± 1.25 | 75.75 ± 1.77 | 62.29 ±1.70 | 75.36 ± 2.03 | 80.90 ±0.77 | 84.20 ± 0.75 |
| DenseNet161 | 50.64 ± 3.27 | 63.65 ± 2.95 | 68.98± 1.79 | 72.86 ±1.82 | 58.39 ±5.59 | 72.14 ±1.82 | 77.36 ± 1.57 | 81.04 ± 1.40 |
| DenseNet169 | 54.15± 4.33 | 68.91± 2.99 | 73.31± 1.72 | 72.97 ± 2.05 | 58.25 ± 1.68 | 73.10 ±0.99 | 78.91 ± 1.02 | 83.31 ± 1.12 |
| DenseNet201 | 60.78 ±2.00 | 71.19 ± 0.84 | 71.48±2.17 | 73.64 ±1.39 | 61.45 ± 5.09 | 74.58 ±2.40 | 80.75 ± 3.86 | 85.02 ±3.98 |
| LightCNN-9 | 55.72 ±2.90 | 66.09 ±2.27 | 67.65 ±2.29 | 71.81 ±1.64 | 56.48 ± 4.60 | 69.82 ±4.49 | 76.91±3.69 | 81.87± 3.93 |
| LightCNN-29 | 53.10 ±3.75 | 65.28 ±2.47 | 71.91±1.99 | 75.85 ±2.02 | 62.67 ±2.59 | 76.19 ±1.15 | 82.55 ± 0.87 | 86.00 ± 1.03 |
| VGG-Face | 60.77 ±1.28 | 72.93 ±1.40 | 77.19 ±1.27 | 79.66 ±1.97 | **70.42 ±0.50** | **81.37 ±1.59** | **86.50 ±1.20** | **90.01±1.53** |

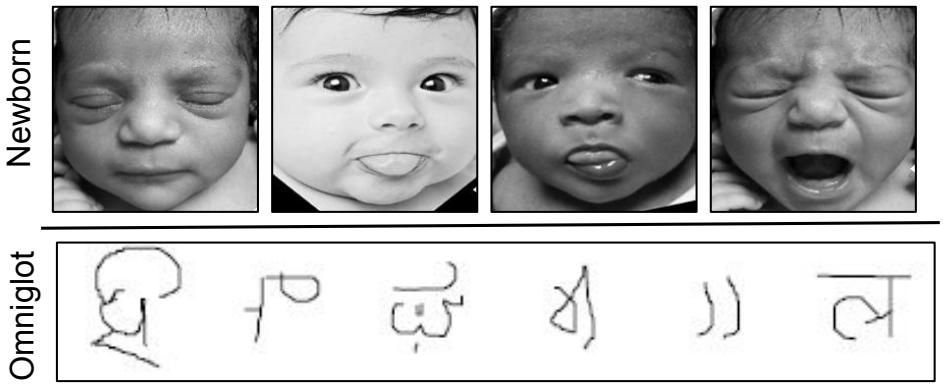



Figure 2-8: Samples images from the Omniglot and Newborn Faces databases.

have random values, learning strength produces considerably high accuracies. Once the filters are trained, optimizing the strength of those filters can further improve the performance.

### 2.3.5 Small Sample Size Case Studies

To showcase the effectiveness of the proposed *structure* and *strength* concept on small sample size databases, we present three case studies: (i) newborn face database [15], (ii) Omniglot database [130] and (iii) Cataract Mobile Pre-post Database (CMPD) [114]. Figure 2-8 and 5-5 shows sample images from the databases.

**Newborn Face Recognition**: Bharadwaj *et al.* [15] have shown that newborn face recognition is a challenging small sample size application. The publicly available IIITD Newborn database

[15] contains face images from 96 newborns. The pre-defined protocol limits us to using training samples from only ten newborns, and testing is performed with 86 newborns. We compute the performance of ResNet architecture where the proposed dictionary-based initialization helps in estimating the structure using images from 10 newborns, and then the strength parameter is used to attune the filters. The observed rank-1 accuracy, in this case, is 36.32%, which is at least 0.5% better than pre-trained ResNet architecture (which is traditionally fine-tuned with newborn training data). Also, the test accuracies are extremely low when we use training images of only ten newborns to train the filter of CNN models from scratch.

As discussed before, we can learn "structure" from large domain-specific data, and then the proposed "strength" can help us attune the filters for problem-specific data. Therefore, we perform experiments with pre-trained networks (pre-trained filters are obtained after learning from either ImageNet or Labeled Faces in the Labeled Faces in the Wild (LFW) [104] and YouTube Faces (YTF) [274] databases) and use strength parameter to attune it for newborn face recognition based on training data of 10 newborns. For this experiment, as shown in Table 2.2, we use variants of ResNet [91], VGG [230], VGGFace [194], LightCNN [278], and DenseNet [103] architectures, and the performance is compared with standard fine-tuning approaches using same images from 10 newborns. As shown in Table 2.2, we have observed that the learning strength of the filters improves the performance of CNN models compared to the conventional fine-tuning approach. With a single gallery image per subject, the best rank-1 accuracy of over 70% is obtained when the proposed strength parameter is used with pre-trained VGG-Face [194], which is at least 10% better than the conventional fine-tuning-based approach. This shows that in real-world applications, the concept of learning structure and strength helps achieve improved performance.

The performance of the proposed approach is also compared with deep hybrid network [190] and ScatNet [7]. For one gallery per subject, the rank-1 accuracies of these two algorithms are $25.18\% \pm 1.33\%$ and $31.04\% \pm 1.94\%$ respectively, which are at least 39% less than the best results reported in Table 2.2. Finally, we also compare the performance of the proposed algorithm with the Vinyals *et al.* [259], Hariharan *et al.* [86], and Bharadwaj *et al.* [15] on newborn face database. Using the same protocol, Figure 2-9 illustrates the comparison between the proposed method (best-reported result in Table 2.2) with existing methods. The proposed method improves the rank-1 accuracies by $11 - 19\%$ for varying numbers of sample(s) per subject. However, the

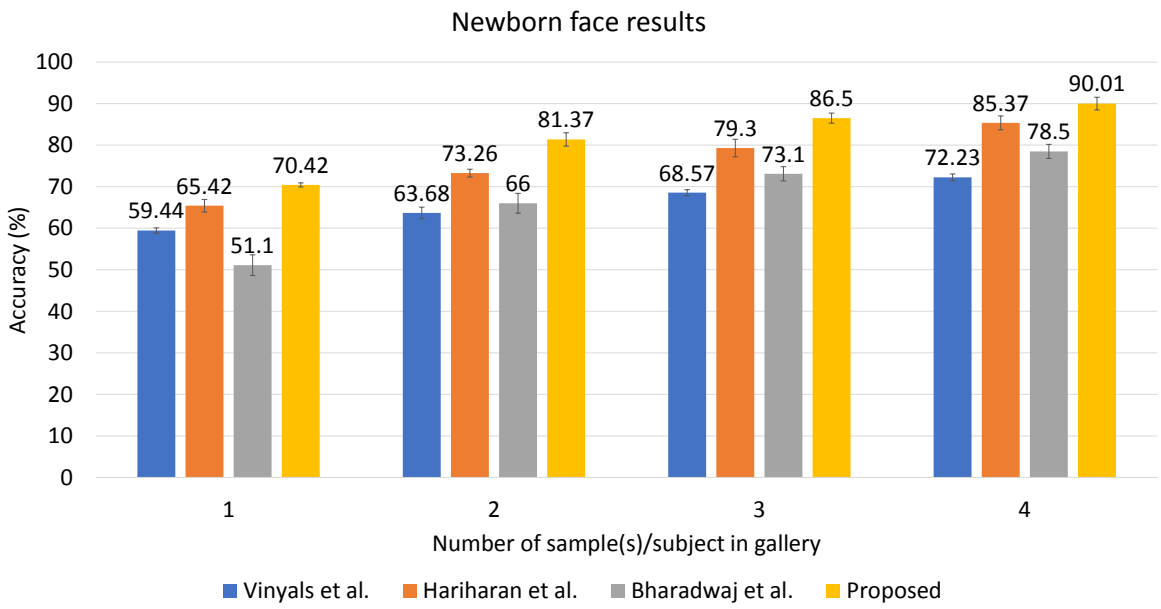



Figure 2-9: Summarizing the results on the newborn face database.

proposed algorithm consistently yields improved accuracy and is approximately 4.5%.

**Omniglot Database**: On the Omniglot database [130], SSF-CNN yields classification accuracies of $97.6\% \pm 0.84\%$ and $98.3\% \pm 1.03\%$ for 1-shot, 5-way and 5-shot, 5-way, respectively which are comparable to state of the art results. Table 2.3 summarizes the results of the proposed algorithm and compares them with existing algorithms. The results show that SSF-CNN is among the top-performing algorithms for both protocols.

**Cataract Mobile Pre-post Database (CMPD)**: On the CMPD database [114], our proposed SS-VGG-face achieved $52.4$, $60.1$ rank-1 identification accuracies on unregistered S1-S2 pre-post surgery improved from $16.26$ and $15.51$ computed on left and right eyes mobile images, respectively. Also, with registered eyes, our model achieved $68.7$ and $66.6$ rank-1 identification accuracy on registered S1-S2 pre-post surgery improved from $30.10$ and $22.41$ computed on left and right eyes mobile images, respectively.

## 2.4 Summary

An extensive training database is a crucial requirement for training convolutional neural networks. However, several applications and problem statements do not have the luxury of large training

Table 2.3: Classification results (%) on the Omniglot database [130].

| **Algorithm** | **1-shot, 5-way** | **5-shot, 5-way** |
|---|---|---|
| Santoro *et al.* [219] | 82.8 | 94.9 |
| Koch *et al.* [122] | 97.3 | 98.4 |
| Vinyals *et al.* [259] | 98.1 | 98.9 |
| **Proposed** | 97.6 | 98.3 |

databases. This research proposes Structure and Strength Filtered CNN as a framework for learning a CNN model with small training databases. We offer to initialize the filters of CNN using dictionary filters, which can be trained with small training samples. Since the dictionary atoms are learned for reconstruction, they may not be optimal for classification. Therefore, we suggest learning the filters' strength with the given training data. The effectiveness of the proposed model has been demonstrated on multiple object classification databases and a real-world newborn face recognition problem. Using different architectures and experiments, we demonstrate the efficacy of the proposed approach. Specifically, in the case of newborn face recognition, remarkable improvement in accuracy is achieved with the proposed approach. The proposed CNN has the flexibility to work for small and large databases. The current model incorporates unsupervised dictionary filters to initialize the CNN network. As a future work, other trained filters, such as supervised dictionary filters, can also be used. They can also be used to adapt the filters from one task to another while learning only the strength of the filters. The proposed algorithm can also be extended to other applications such as face recognition with variations in disguise [50], matching faces in videos [79], and sketch to photo matching [178].

# Chapter 3

# Guided Dropout

*"Better than a thousand days of diligent study is one day with a great teacher."*

— Japanese proverb

## 3.1 Introduction

Deep neural networks have gained a lot of success in multiple applications. However, due to optimizing millions of parameters, generalization of DNN is a challenging task. Multiple regularizations have been proposed in the literature such as $l_1 - norm$ [186], $l_2 - norm$ [186], max-norm [233], rectifiers [179], KL-divergence [93], drop-connect [262], and dropout [95], [233] to regulate the learning process of deep neural networks consisting of a large number of parameters. Among all the regularizers, dropout has been widely used to generalize DNNs.

Dropout [95], [233] improves the generalization of neural networks by preventing co-adaptation of feature detectors. The working of dropout is based on the generation of a mask by utilizing $Bernoulli$ and $Normal$ distributions. At every iteration, it generates a random mask with probability $(1 - \theta)$ for hidden units of the network. [266] have proposed a Gaussian dropout which is a fast approximation of conventional dropout. [120] have proposed variational dropout to reduce the variance of Stochastic Gradients for Variational Bayesian inference (SGVB). They have shown that variational dropout is a generalization of Gaussian dropout where the dropout rates are learned.

[121] have proposed alpha-dropout for Scaled Exponential Linear Unit (SELU) activation function. [9] have proposed "standout" for a deep belief neural network where, instead of initializ-

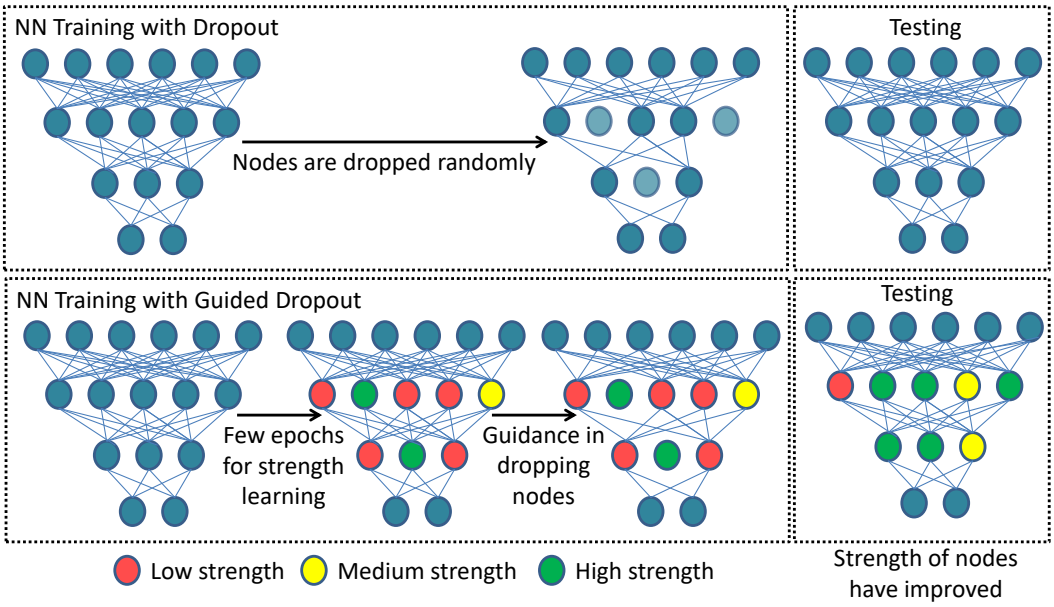



Figure 3-1: Illustrating the effect of conventional dropout and proposed guided dropout. In neural network training with dropout, nodes are dropped randomly from the hidden layers. However, in the proposed guided dropout, nodes are dropped based on their strength. (Best viewed in color).

ing dropout mask using $Bernoulli$ distribution with probability $p$, they have adapted the dropout probability for each layer of the neural network. In addition to the conventional learning methods of dropout, [67] have utilized the Gaussian process for the deep learning models, which allows estimating the uncertainty of the function, robustness to over-fitting, and hyper-parameter tuning. They have measured model uncertainty by measuring the first and second moments of their approximate predictive distribution. [68] have proposed "Concrete Dropout", a dropout variant where the concrete distribution has been utilized to generate the dropout mask. They have optimized the probability $p$ via a path-wise derivative estimator. Dropout is also explored between convolutional layer [21] of CNN model like DropBlock [73], CorrDropout [297], DropCluster [30], and FocusedDropout [152].

In the literature, methods related to dropout have been explored in two aspects: 1) sampling dropout mask from different distributions and maintaining the mean of the intermediate input while dropping nodes, and 2) adapting dropout probability. However, if prior information related to nodes of a NN is available, nodes can be dropped selectively so that the generalization of NN is improved. Therefore, in this research, we propose a "strength parameter" to measure the importance of nodes and features for dense NN and CNN, respectively, and use it for guiding dropout regularization.

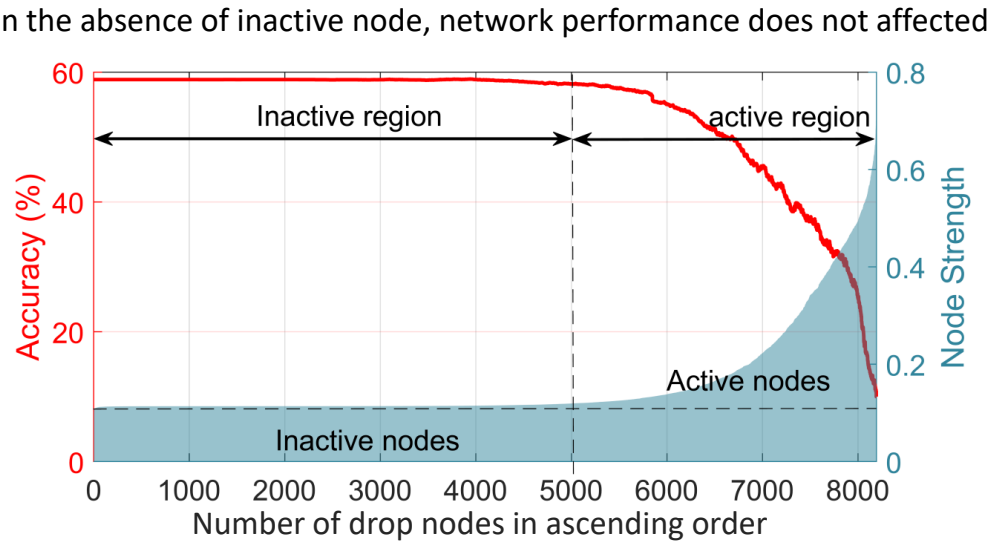



Figure 3-2: A Neural Network NN$[8192, 3]$ is trained with strength parameters on the CIFAR10 dataset. The bar graph represents the trained strength of the first layer of the NN. It can be observed that low-strength nodes are not contributing to the performance, and removing such nodes minimally affects the accuracy. Such nodes are termed inactive nodes in the inactive region. Similarly, high-strength nodes contribute to the performance, and eliminating such nodes affects the accuracy. Such nodes are termed as active nodes in the active region. (Best viewed in color).

Figure 3-1 illustrates the graphical abstract of the paper "guided Dropout". To understand the behavior of strength parameter $t$, a three hidden layer neural network with $8192$ nodes is trained using Equation 3.2 (details discussed in the next section). After training, the accuracy is evaluated by removing low-strength to high-strength nodes one by one. The effect on the accuracy can be observed in Figure 3-2. It shows that removing up to almost $5000$ low-strength nodes has minimal effect on the network accuracy. Therefore, such nodes are considered to be *inactive nodes*, lying in the *inactive region*. On removing nodes with high strength, the network accuracy reduces aggressively. Such nodes are considered to be *active nodes* in the *active region*.

Our hypothesis is that in the absence of high-strength nodes during training, low-strength nodes can improve their strength and contribute to the performance of NN. To achieve this, while training a NN, we drop the high-strength nodes in the active region and learn the network with low-strength nodes. This is termed as *Guided Dropout*. As shown in Figure 3-1, during training, the network generalizability is strengthened by "nurturing" inactive nodes. Once trained, more nodes contribute towards making predictions, thus improving accuracy. The key contribution of this paper is that a strength parameter is proposed for deep neural networks, which is associated with each node.

Using this parameter, a novel guided dropout regularization approach is proposed. To the best of our knowledge, this is the first attempt to remove randomness in the mask generation process of dropout. We have also presented that conventional dropout is a special case of guided dropout and is observed when the concept of active and inactive regions are not considered. Further, experimental and theoretical justifications are also presented to demonstrate that the proposed guided dropout performance is always equal to or better than the conventional dropout.

## 3.2 Proposed Guided Dropout

In dropout regularization, some of the nodes from the hidden layers are dropped at every epoch with probability $(1-\theta)$. Let $l$ be the $l^{th}$ layer of a network, where the value of $l$ ranges from $0$ to $L$, and $L$ is the number of hidden layers in the network. When the value of $l$ is zero, it represents the input layer, i.e., $a^0 = X$. Let the intermediate output of the network be $z^{(l)}$. Mathematically, it can be expressed as:

$$\begin{aligned} z_j^{(l+1)} &= w_{j\times i}^{(l+1)} a_i^l + b_j^{(l+1)} \\ a_j^{(l+1)} &= f(z_j^{(l+1)}) \end{aligned} \tag{3.1}$$

where, $i \in [1, ..., N_{in}]$, and $j \in [1, ..., N_{out}]$ are index variables for $N_{in}$ and $N_{out}$ at the $(l+1)^{th}$ layer, respectively. $f(.)$ is the $ReLU$ activation function. The conventional dropout drops nodes randomly using $Bernoulli$ distribution and is expressed as: $\widetilde{\mathbf{a}}^{(l)} = \mathbf{r}^{(l)} \odot \mathbf{a}^{(l)}$, where $\widetilde{a}$ is the masked output, $a^{(l)}$ is the intermediate output, $\odot$ is the element-wise multiplication, and $r^{(l)}$ is the dropout mask sampled from $Bernoulli$ distribution. While dropping nodes from NN, expected loss $\mathbb{E}(.)$ increases which enforces regularization penalty on NN to achieve better generalization [171].

### 3.2.1 Introducing strength parameter

As shown in Figure 3-3, network performance is almost similar in the initial few iterations of training with and without dropout. The effectiveness of dropout can be observed after a few iterations of training. Dropping some of the trained nodes may lead to more active nodes in the

network. Hence, the performance of the network can be improved. Utilizing this observation and the discussion presented with respect to Figure 3-2 (about active/inactive nodes), we hypothesize that a guided dropout can lead to better generalization of a network. The proposed guided dropout utilizes the strength of nodes for the generation of the dropout mask. In the proposed formulation, strength is learned by the network itself via Stochastic Gradient Descent (SGD) optimization. Mathematically, it is expressed as:

$$a_j^{(l+1)} = t_j^{(l+1)} \odot max\left(0, w_{j\times i}^{(l+1)} a_i^l + b_j^{(l+1)}\right) \tag{3.2}$$

where, $\mathbf{t}^{(l)}$ is sampled from $uniform$ distribution (assuming all nodes have equal contribution). It can also be used to measure the importance of the feature-map for CNN networks. Therefore, Equation 3.2 can be rewritten as:

$$\mathbf{a}^{(l+1)} = \mathbf{t}^{(l+1)} \odot max\left(0, \mathbf{a}^l * \mathbf{W}^{(l+1)} + b^{(l+1)}\right) \tag{3.3}$$

where, $*$ is a convolution operation, max(0, .) is a $ReLU$ operation[1], and $\mathbf{a}^l$ is a three-dimensional feature map (for ease of understanding, subscript has been removed).

**Strength parameter in matrix decomposition:** To understand the behavior of the proposed strength parameter **t** in a simpler model, let the projection of input $x \in \mathbb{R}^{d_2}$ on $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ represent label vector $y$ ($\in \mathbb{R}^{d_1}$). Matrix **W** can be linearly compressed using singular value decomposition (SVD), i.e., $W = Udiag(t)V^T$. Here, top few entries of $diag(t)$ can approximate the matrix **W** [49]. This concept can be utilized in the proposed guided dropout.

**Strength parameter in two hidden layers of an NN:** In an NN environment, let $V \in \mathbb{R}^{d_2 \times r}$ and $U \in \mathbb{R}^{d_1 \times r}$ be the weight matrices of the first and second hidden layers of NN, respectively. The hypothesis class can be represented as $h_{U,V}(x) = UV^T x$ [171]. In the case of *Guided Dropout*, the hidden node is parameterized as $h_{U,V,t}(x) = Udiag(t)V^T x$, which is similar to the SVD decomposition where parameter **t** is learned via back-propagation. Therefore, $t$ can be considered as a strength of a node, which is directly proportional to the contribution of the node in NN perfor-

[1]In the case of other activation functions, intermediate feature maps might have negative values. Therefore, $|t|$ (mod of '$t$') can be considered a strength parameter. In this case, '$t$' value approaching zero represents low strength and a node associated with low strength can be considered as an inactive node.

mance.

In this case, the weight update rule for parameters $U$, $V$ and $\mathbf{t}$ on the $(s+1)^{th}$ batch can be written as:

$$U_{s+1} \leftarrow U_s - \eta \left( \frac{1}{\theta} U_s diag(t_s \odot r_s) V_s^T x_s - y_s \right) x_s^T V_s diag(t_s \odot r_s)$$

$$V_{s+1} \leftarrow V_s - \eta x_s \left( \frac{1}{\theta} x_s^T V_s diag(t_s \odot r_s) U_s^T - y_s^T \right) U_s diag(t_s \odot r_s)$$

$$diag(t_{s+1}) \leftarrow diag(t_s) - \eta U_s^T \left( \frac{1}{\theta} U_s diag(t_s \odot r_s) V_s^T x_s - y_s \right) x_s^T V_s \quad (3.4)$$

where, $\{(x_s, y_s)\}_{s=0}^{S-1}$ is the input data, $(1-\theta)$ is the dropout rate, $\eta$ is the learning rate, and $r$ is the dropout mask. For the initial few iterations, $r$ is initialized with ones. However, after a few iterations of training, $r$ is generated using Equation 3.5.

In the proposed algorithm, active nodes are dropped in two ways:

1. Guided Dropout (top-$k$): Select (top-$k$) nodes (using strength parameter) to drop
2. Guided Dropout (DR): Drop Randomly from the active region.

**Proposed Guided Dropout (top-k):** While dropping (top-k) nodes based on the strength, the mask for the proposed guided dropout can be represented as:

$$\mathbf{r}^l = \mathbf{t}^l \leq th, \; where, \; th = \max_{\lfloor N \times (1-\theta) \rfloor} \mathbf{t}^l \quad (3.5)$$

$\max_{\lfloor N \times (1-\theta) \rfloor}$ is defined as the $k$ large elements of $\mathbf{t}$ where, $(1-\theta)$ is the percentage ratio of nodes needed to be dropped and $N$ is the total number of nodes. The generated mask $\mathbf{r}^l$ is then utilized in equation $\widetilde{\mathbf{y}}^{(l)} = \mathbf{r}^{(l)} \odot \mathbf{y}^{(l)}$ to drop the nodes. Since the number of dropped nodes are dependent on the total number of nodes $N$ and percentage ratio $(1-\theta)$, expected loss can be measured by $\mathbb{E}_{b,x}[||y - \frac{1}{\theta} U diag(r) V^T x||^2]$. If dropout mask $\mathbf{r}^l$ drops $top-k$ nodes, the expected

loss would be maximum with respect to conventional dropping nodes. Therefore, guided dropout ($top-k$) would impose a maximum penalty in NN loss.

**Proposed Guided Dropout (DR):** The second way of generating a guided dropout mask is to select the nodes from the active region, i.e., nodes are Dropped Randomly (DR) from the active region only. Since the number of inactive nodes is large and has a similar strength, to find the active or inactive region, the number of elements in all the bins[2] have been computed. The maximum number of elements among all the bins is considered as the count of inactive nodes $f_m$. Thus, $f_m$ and $(N - f_m)$ are the number of inactive and active nodes, respectively. Here, $(1 - \theta)$ is the probability for sampling dropout mask for active region nodes using $Bernoulli$ distribution. Probability with respect to the total number of nodes should be reduced to maintain the mean $\mu$ in the training phase. Therefore, new probability with respect to $N$ is modified as $(1 - \frac{f_m}{N}(1 - \theta))$. For the proposed guided dropout (DR), when the nodes are dropped randomly from the active region, in Equation 3.4, $\frac{1}{\theta}$ will be modified as $\frac{1}{\frac{f_m}{N}(1-\theta)}$. We have carefully mentioned $(1-\theta)$ as the percentage ratio for the proposed guided dropout ($top-k$). In the case of guided dropout ($top-k$), the generated mask might be fixed until any low-strength node can replace the ($top-k$) nodes. On the other hand, for the proposed guided dropout (DR), $(1 - \theta)$ is the dropout probability.

### 3.2.2 Why Guided Dropout Should Work?

Dropout improves the generalization of neural networks by preventing the co-adaptation of feature detectors. However, we assert that guidance is essential while dropping nodes from the hidden layers. Guidance can be provided based on the regions where nodes are dropped randomly or the top few nodes are dropped from the active region. To understand the generalization of the proposed guided dropout, we have utilized Lemma A.1 from [171]. In their proposed lemma: Let $x \in \mathbb{R}^{d_2}$ be distributed according to distribution $D$ with $\mathbb{E}_x[xx^T] = \mathbf{I}$. Then, for $\mathcal{L}(U, V) := \mathbb{E}_x[||y-UV^Tx||^2]$ and $f(U, V) := \mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r)V^Tx||^2]$, it holds that

$$f(U, V) = \mathcal{L}(U, V) + \lambda \sum_{i=1}^{n} ||u_i||^2||v_i||^2 \tag{3.6}$$

[2]In this case, 100 equally spaced bins are chosen.

Furthermore, $\mathcal{L}(U,V) = ||W - UV^T||_F^2$, where, $diag(r) \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{d_2\times n}$, $U \in \mathbb{R}^{d_1\times n}$, $W \in \mathbb{R}^{d_1\times d_2}$, $\lambda = \frac{1-\theta}{\theta}$. (The proof of this lemma is given in [171]).

*According to the lemma mentioned above, it can be observed that the guided dropout assists NN to have a better generalization*. Let $r$ be sampled from $Bernoulli$ distribution at every iteration to avoid overfitting in conventional dropout. In guided dropout, mask $r'$ is generated based on the strength value $t$. For Equation $\mathcal{L}(U,V) = ||W - Udiag(t)V^T||_F^2$, high strength nodes can be chosen to form mask $r'$. Therefore, loss $\mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r')V^Tx||^2] \geq \mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r)V^Tx||^2]$. In this case, the penalty would increase while dropping higher-strength nodes. The expected loss would be the same only if $r = r'$. If $r \neq r'$, the regularization imposed by the proposed guided dropout increases in the training process. Hence, optimizing the loss in the training process helps inactive nodes to improve their strength in the absence of higher-strength nodes.

From Equation 3.6, the path regularization term $\lambda \sum_{i=1}^{n} ||u_i||^2||v_i||^2$ regularizes the weights $u_i$ and $v_i$ of the inactive node such that the increase in loss due to dropping higher strength nodes can be minimized. Thus, the worst case of generalization provided by the proposed guided dropout should be equal to the generalization provided by the conventional dropout.

### 3.2.3 Implementation Details

Experiments are performed on a workstation with two 1080Ti GPUs under PyTorch [195] programming platform. The program is distributed on both GPUs. The number of epochs, learning rate, and batch size are kept as 200, $[10^{-2}, ..., 10^{-5}]$, and 64, respectively, for all the experiments. The learning rate starts from $10^{-2}$ and is reduced by $10$ at every 50 epochs. The best-performing results are obtained at $0.2$ dropout probability for conventional dropout. In the proposed guided dropout, $40$ epochs have been used to train the strength parameter. Once the strength parameter is trained, dropout probabilities for guided dropout (DR) are set to $0.2$, $0.15$, and $0.1$ for $60$, $50$, and $50$ epochs, respectively. However, after strength learning, the dropout ratio for guided dropout (top-k) is set to [$0.2$, $0.0$, $0.15$, $0.0$, $0.1$, $0.0$] for [$10$, $40$, $10$, $40$, $10$, $50$] epochs, respectively.

## 3.3 Experimental Results and Analysis

The proposed method has been evaluated using three experiments: i) guided dropout in the neural network, ii) guided dropout in the deep network (ResNet18 and Wide ResNet 28-10), and iii) case study with a small sample size problem. The databases used for evaluation are MNIST, SVHN, CIFAR10, CIFAR100, and Tiny ImageNet.

The proposed guided dropout is compared with state-of-art methods such as Concrete dropout[3][68], Adaptive dropout (Standout)[4] [9], Variational dropout[5] [120], and Gaussian dropout[5]. Alpha-dropout [121] has also been proposed in the literature. However, it is specifically designed for SELU activation function. Therefore, to have a fair comparison, the results of the alpha-dropout are not included in the Tables.

### 3.3.1 Database and Experimental Protocol

*Protocol for complete database:* Five benchmark databases including MNIST [138], CIFAR10 [126], CIFAR100 [126], SVHN [180], and Tiny ImageNet [247] have been used to evaluate the proposed method. MNIST is only used for benchmarking Neural Network with conventional dropout [233]. The MNIST dataset contains $70k$ grayscale images pertaining to 10 classes ($28 \times 28$ resolution). The CIFAR10 dataset contains $60k$ color images belonging to $10$ classes ($32 \times 32$ resolution). The experiments utilize $50k$ training samples and $10k$ as the test samples. CIFAR100 has a similar protocol with 100 classes. The protocol for CIFAR100 also has $50k$ and $10k$ training-testing split. The SVHN dataset contains $73,257$ training samples and $26,032$ testing samples. The Tiny ImageNet dataset is a subset of the ImageNet dataset with 200 classes. It has images with $64 \times 64$ resolution with $100k$ and $10k$ samples for training and validation sets, respectively. The test-set label is not publicly available. Therefore, the validation set is treated as a test set for all the experiments on Tiny ImageNet.

*Protocol for small sample size problem:* Recent literature has emphasized the importance of deep learning architecture working effectively with small sample size problems [118]. Therefore, the effectiveness of the proposed algorithm is tested for small sample size problems. The exper-

[3] https://tinyurl.com/yb5msqrk
[4] https://tinyurl.com/y8u4kzyq
[5] https://tinyurl.com/y8yf6vmo

Table 3.1: Test accuracy (%) on CIFAR10 and CIFAR100 [126] databases using four different architectures of a three layer Neural Network. (Top two accuracies are in bold).

| **Algorithm** | **CIFAR10** | | | | **CIFAR100** | | | |
|---|---|---|---|---|---|---|---|---|
| | 1024, 3 | 2048, 3 | 4096, 3 | 8192, 3 | 1024, 3 | 2048, 3 | 4096, 3 | 8192, 3 |
| Without Dropout | 58.59 | 59.48 | **59.72** | 59.27 | 28.86 | 30.01 | 30.73 | 32.02 |
| With Dropout | **58.77** | 59.61 | 59.62 | 59.86 | **31.52** | **31.63** | **31.37** | 31.63 |
| Concrete Dropout | 57.38 | 57.64 | 57.45 | 55.28 | 28.03 | 29.09 | 28.91 | 31.02 |
| Adaptive Dropout | 55.05 | 55.45 | 56.84 | 57.01 | 27.82 | 28.27 | 28.62 | 28.65 |
| Variational Dropout | 48.90 | 52.08 | 53.48 | 54.90 | 17.02 | 20.64 | 23.32 | 24.53 |
| Gaussian Dropout | 56.12 | 56.52 | 56.94 | 57.34 | 27.24 | 28.34 | 28.87 | 29.81 |
| Strength only | 58.30 | 58.92 | 59.21 | 59.49 | 29.66 | 30.20 | 30.84 | 31.12 |
| Proposed Guided Dropout (top-k) | 58.75 | **59.65** | 59.64 | **59.92** | 30.92 | 31.59 | 31.34 | **32.11** |
| Proposed Guided Dropout (DR) | **59.84** | **60.12** | **60.89** | **61.32** | **31.88** | **32.78** | **33.01** | **33.15** |

Table 3.2: Test accuracy (%) on SVHN [180] and Tiny ImageNet [247] databases using four different architectures of a three layer NN. (Top two accuracies are in bold).

| **Algorithm** | **SVHN** | | | | **TinyImageNet** | | | |
|---|---|---|---|---|---|---|---|---|
| | 1024, 3 | 2048, 3 | 4096, 3 | 8192, 3 | 1024, 3 | 2048, 3 | 4096, 3 | 8192, 3 |
| Without Dropout | **86.36** | **86.72** | **86.82** | 86.84 | 12.42 | 13.74 | 14.64 | 15.21 |
| With Dropout | 85.98 | 86.60 | 86.77 | 86.79 | **16.39** | 14.28 | 14.69 | 14.44 |
| Concrete Dropout | 83.57 | 84.34 | 84.97 | 85.53 | 11.98 | 12.50 | 12.65 | 14.85 |
| Adaptive Dropout | 77.67 | 79.68 | 80.89 | 81.96 | 12.41 | 12.98 | 13.75 | 14.17 |
| Variational Dropout | 74.28 | 77.91 | 80.22 | 81.52 | 7.95 | 10.08 | 12.91 | 14.69 |
| Gaussian Dropout | 72.46 | 78.07 | 80.42 | 80.74 | 13.88 | **15.67** | **15.76** | 15.94 |
| Strength only | 85.76 | 85.92 | 85.91 | 86.83 | 12.11 | 13.52 | 13.95 | 14.63 |
| Proposed Guided Dropout (top-k) | 86.12 | 86.57 | 86.78 | **86.85** | 15.47 | 15.45 | 15.55 | **16.01** |
| Proposed Guided Dropout (DR) | **87.64** | **87.92** | **87.95** | **87.99** | **17.59** | **18.84** | **18.41** | **17.74** |

iments are performed on the Tiny ImageNet database with three-fold cross-validation. From the entire training set, $200, 400..., 1k, 2k, .., 5k$ samples are randomly chosen to train the network, and evaluation is performed on the validation set.

### 3.3.2 Evaluation of Guided Dropout in Dense Neural Network (NN) Architecture

To showcase the generalization of the proposed method, training and testing loss at every epoch is shown in Figure 3-3. A three-layer NN with $8192$ nodes at each layer is trained without dropout, with dropout, and with the two proposed guided dropout algorithms $top-k$, and DR. It can be

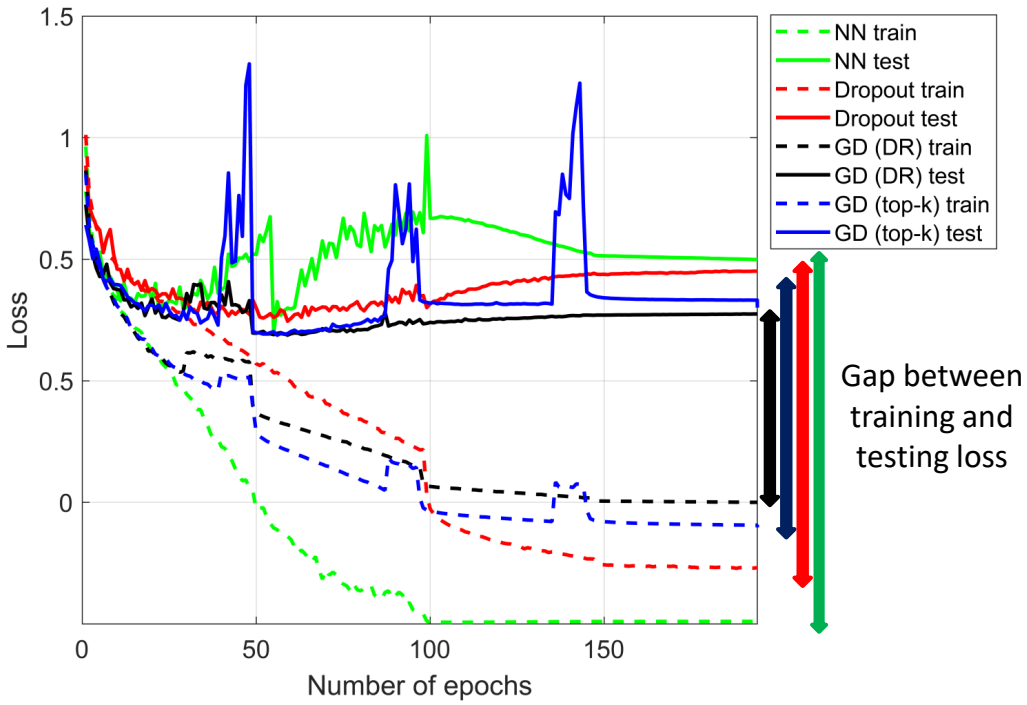



Figure 3-3: Illustrating training and testing losses at every epoch. On the CIFAR10 dataset, the proposed method is compared with the conventional dropout method. It can be observed that the gap between training and testing loss is minimal in the proposed guided dropout. (Best viewed in color).

inferred that the proposed guided dropout approaches help to reduce the gap between the training and testing losses.

The proposed guided dropout is evaluated on three layers NN with four different architectures as suggested in [233]. Tables 3.1 to 3.3 summarize test accuracies (%) on CIFAR10, CIFAR100, SVHN, Tiny ImageNet, and MNIST databases. It can be observed that the proposed guided dropout (DR) performs better than existing dropout methods. In large parameter settings such as three-layer NN with $8192$ nodes, the proposed guided dropout (top-k) algorithm also shows comparable performance. For NN$[1024, 3]$ architecture, conventional dropout is the second-best-performing algorithm on CIFAR10, CIFAR100, and Tiny ImageNet databases.

We have claimed that the strength parameter is an essential element in NN to measure the importance of nodes. Though the number of training parameters is increased, this overhead is less than $0.2\%$ of the total number of parameters of a NN[6].

[6]For a NN with three hidden layers of $8192$ nodes each, total number of learning parameters is only $8192 \times 8192 +$

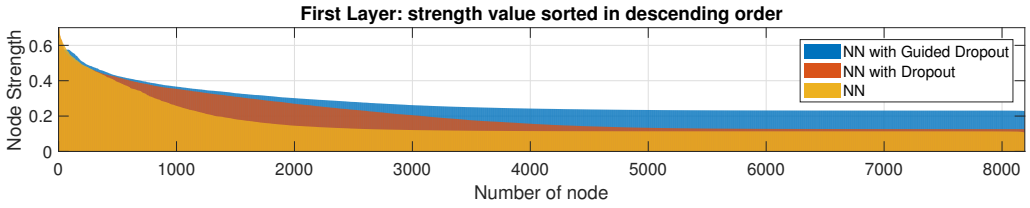



Figure 3-4: Illustrating the learned strength values of the first hidden layer nodes for the CIFAR10 database with NN$[8192, 3]$. The strength of nodes is improved by utilizing the proposed guided dropout compared to with/without conventional dropout. (Best viewed in color).

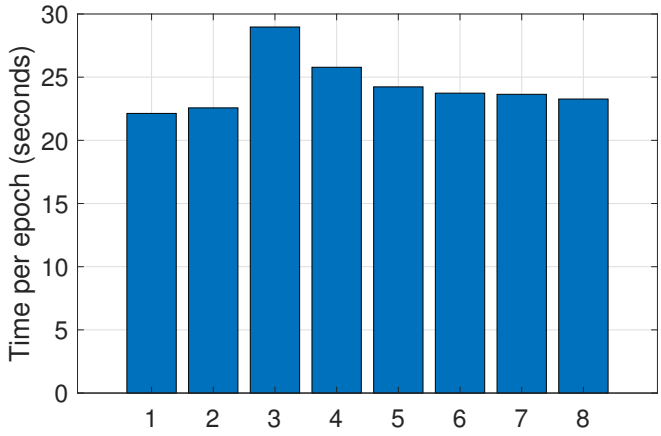



Figure 3-5: Illustration of time taken per epoch. X-axis represents without dropout (1), with dropout (2), concrete dropout (3), adaptive dropout (4), variational dropout (5), Gaussian dropout (6), proposed guided dropout (top-k) (7), and proposed guided dropout (DR) (8) algorithms, respectively.

Figure 3-4 represents the learned strength value of the first hidden layer of NN$[8196, 3]$. It can be observed that the conventional dropout improves the strength of hidden layer nodes. However, the strengths are further improved upon by utilizing the proposed guided dropout.

For understanding the computational requirements, a NN$[8192, 3]$ has been trained and time taken without dropout, with dropout, concrete dropout, adaptive dropout, variational dropout, Gaussian dropout, proposed guided dropout (top-k), and proposed guided dropout (DR) and the results are reported for one epoch. Figure 3-5 summarizes the time (in seconds) for these variations, showing that applying the proposed dropout approach does not increase the time requirement.

$8192 \times 8192 = 134,217,728$ and overhead of strength parameter is $24,576$.

Table 3.3: Test accuracy (%) on the MNIST [138] database using three layer NN. (Top two accuracies are in bold).

| Algorithm | Number of nodes, Layers | | | |
|---|---|---|---|---|
| | **1024, 3** | **2048, 3** | **4096, 3** | **8192, 3** |
| Without Dropout | 98.44 | 98.49 | 98.42 | 98.41 |
| With Dropout | 98.45 | **98.67** | 98.50 | 98.53 |
| Concrete Dropout | **98.66** | 98.60 | **98.62** | 98.59 |
| Adaptive Dropout | 98.31 | 98.33 | 98.34 | 98.40 |
| Variational Dropout | 98.47 | 98.55 | 98.58 | 98.52 |
| Gaussian Dropout | 98.35 | 98.43 | 98.47 | 98.44 |
| Strength only | 98.42 | 98.51 | 98.40 | 98.46 |
| Proposed Guided Dropout (top-k) | 98.52 | 98.59 | 98.61 | **98.68** |
| Proposed Guided Dropout (DR) | **98.93** | **98.82** | **98.86** | **98.89** |

Table 3.4: Test accuracy (%) on CIFAR10, CIFAR100 [126] (in Table written as C10, C100), SVHN [180], and Tiny ImageNet [247] databases using CNN architectures of ResNet18 and Wide-ResNet 28-10. (Top two accuracies are in bold).

| Algorithm | ResNet18 | | | | Wide-ResNet 28-10 | | | |
|---|---|---|---|---|---|---|---|---|
| | **C10** | **C100** | **SVHN** | **Tiny ImageNet** | **C10** | **C100** | **SVHN** | **Tiny ImageNet** |
| Without Dropout | 93.78 | **77.01** | 96.42 | 61.96 | 96.21 | 81.02 | 96.35 | 63.57 |
| With Dropout | **94.09** | 75.44 | **96.66** | **64.13** | **96.27** | **82.49** | **96.75** | **64.38** |
| Concrete Dropout | 91.33 | 74.74 | 92.63 | 62.95 | 92.63 | 75.94 | 92.79 | – |
| Adaptive Dropout | 90.45 | 73.26 | 92.33 | 61.14 | 79.04 | 52.12 | 90.40 | 62.15 |
| Variational Dropout | 94.01 | 76.23 | 96.12 | 62.75 | 96.16 | 80.78 | 96.68 | 64.36 |
| Gaussian Dropout | 92.34 | 75.11 | 95.84 | 60.33 | 95.34 | 79.76 | 96.02 | 63.64 |
| Strength only | 93.75 | 76.23 | 96.34 | 62.06 | 95.93 | 80.79 | 96.31 | 64.13 |
| Proposed Guided Dropout (top-k) | 94.02 | 76.98 | 96.62 | 64.11 | 96.22 | 82.31 | 96.42 | 64.32 |
| Proposed Guided Dropout (DR) | **94.12** | **77.52** | **97.18** | **64.33** | **96.89** | **82.84** | **97.23** | **66.02** |

### 3.3.3 Evaluation of Guided Dropout in Convolutional Neural Network (CNN) Frameworks

The proposed guided dropout is also evaluated on CNN architectures of ResNet18 and Wide-ResNet 28-10. The proposed guided dropout performance is compared with existing state-of-the-art dropout methods on the same protocol. Table 3.4 summarizes test accuracies of four benchmarking databases. It can be observed that on CIFAR10 (C10), dropout provides the second-best performance after the proposed algorithm. On CIFAR100 (C100), without dropout, it provides the second-best performance, and the proposed guided dropout (DR) provides the best performance.

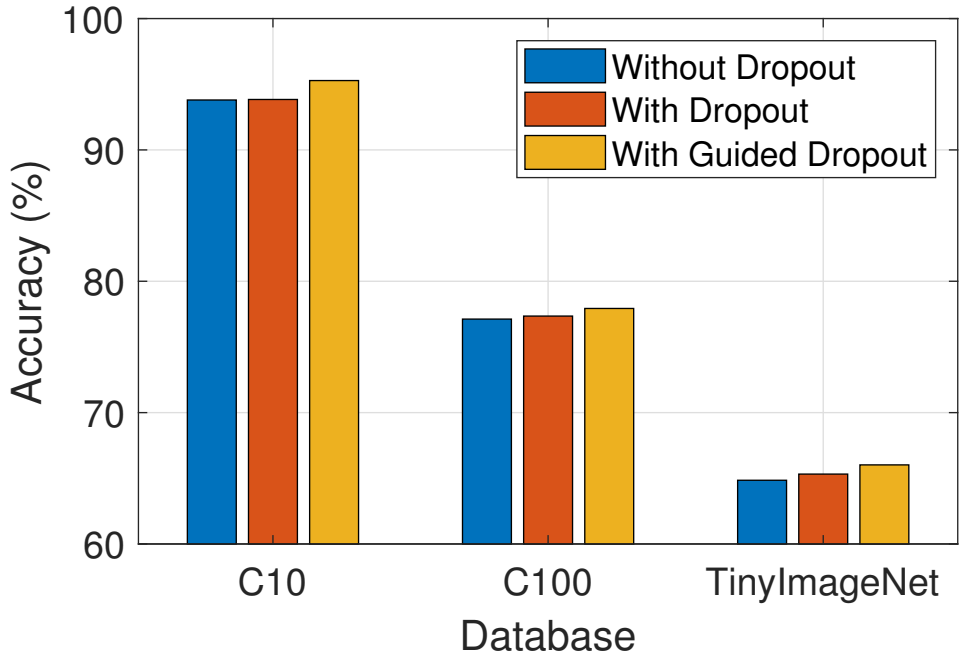



Figure 3-6: Classification accuracy on C10, C100, and Tiny ImageNet databases. The performance is measured with ResNet152 CNN architecture without dropout, with traditional dropout, and with the proposed guided dropout. (Best viewed in color).

In the case of Wide-ResNet 28-10, which has larger parameter space than ResNet18, conventional dropout consistently performs second best after the proposed guided dropout (DR). It improves the Wide-ResNet 28-10 network performance by $0.62\%$, $0.35\%$, $0.48\%$, and $1.64\%$ on C10, C100, SVHN, and Tiny ImageNet databases, respectively. We have also computed the results using ResNet152 CNN architecture on C10, C100, and Tiny ImageNet databases. As shown in Figure 3-6, even with a deeper CNN architecture, the proposed guided dropout performs better than the conventional dropout method.

### 3.3.4 Small Sample Size Problem

**Bench marking dataset:** Avoiding overfitting for small sample size problems is a challenging task. A deep neural network, which has many parameters, can easily overfit on small data. For measuring the generalization performance of models, [19] suggested measuring the model's generalization error by reducing the training dataset's size. Therefore, we have performed three-fold validation along with varying the training data size.

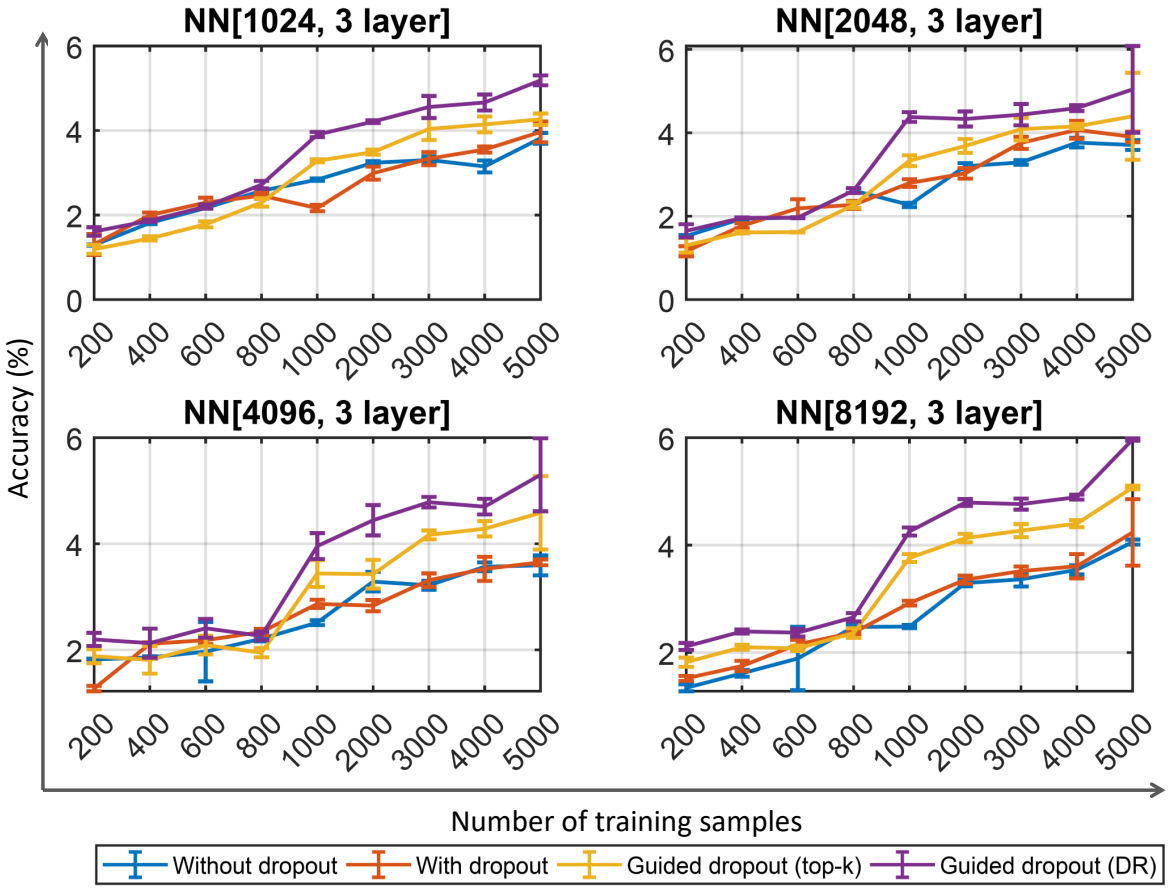



Figure 3-7: Results of small sample size experiments: Accuracies on varying training samples of the Tiny ImageNet dataset. The performance has been measured on four different dense NN architectures. (Best viewed in color).

The experiments are performed with ResNet-18 and four dense neural network architectures. As shown in Figures 3-7 and 3-8, with varying training samples of the Tiny ImageNet database, the proposed guided dropout yields higher accuracies than the conventional dropout.

**Real-World Small Sample dataset:** We have evaluated our proposed guided dropout on Newborn [15] and CMPD [114] datasets. We have achieved rank-1 accuracies $21.5 \pm 0.72$, $32.4 \pm 0.59$, $39.7 \pm 0.84$, and $44.2 \pm 36$ while having gallery 1, 2, 3, and 4, respectively. On CMPD dataset, rank-1 accuracies when eyes are not registered for left $15.1$ and for right $14.5$. After registration accuracies are $27.4$ and $25.8$ for left and right eye respectively.

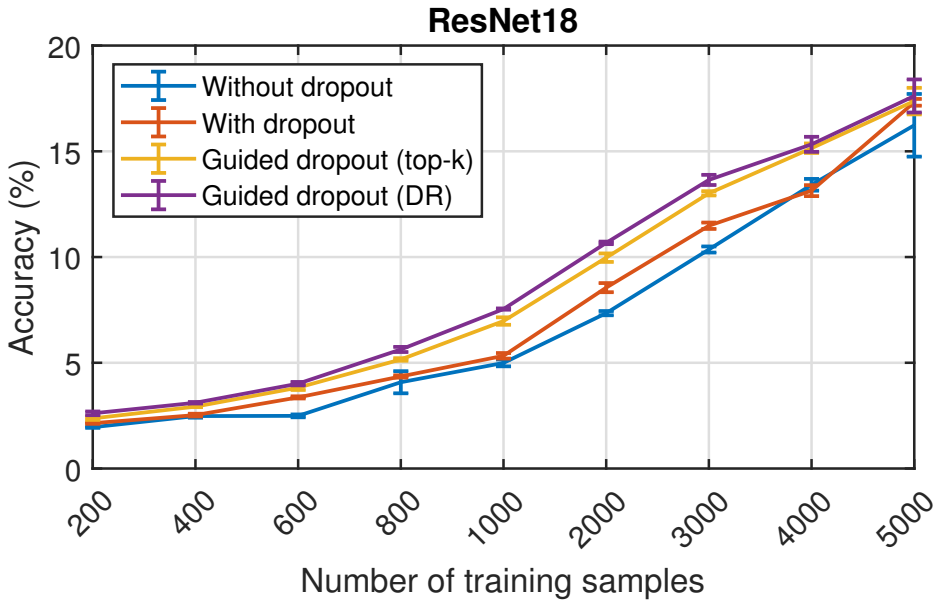



Figure 3-8: Classification accuracies obtained with varying training samples for the Tiny ImageNet dataset. The performance is measured with ResNet18 CNN architecture. (Best viewed in color).

## 3.4 Summary

Dropout is a widely used regularizer to improve the generalization of neural networks. In the dropout-based training, a mask is sampled from $Bernoulli$ distribution with $(1-\theta)$ probability, which is used to drop nodes at every iteration randomly. In this research, we propose a guidance-based dropout, termed guided dropout, which drops active nodes with high strength in each iteration to force non-active or low-strength nodes to learn discriminative features. Low-strength nodes start contributing to the learning process during training to minimize the loss, and eventually, their strength is improved. The proposed guided dropout has been evaluated using dense neural network architectures and convolutional neural networks. All the experiments utilize benchmark databases, and the results showcase the effectiveness of the proposed guided dropout.

We have also extended the idea of guided dropout in the convolutional network, Guided DropBlock, along with filter augmentation, which can reduce the dependency on large datasets explained in Chapter 5

# Chapter 4

# Generalized Zero-Shot Learning Via Over-Complete Distribution

*"Life calls not for perfection, but for completeness."*

— Carl Jung

## 4.1 Introduction

Deep Neural Network (DNN) models have exhibited superlative performance in a variety of real-world applications when the models are trained on large datasets. However, with a small sample size training set, deep learning models tend to overfit, thus leading to poor generalization. Based on the availability of labeled/unlabeled data, multiple learning settings such as transfer learning [25], life-long learning [246], self-taught learning [201], and one-shot learning [173] have been proposed for better generalization. In certain applications, it is possible that samples from all the classes are not present in the training data to train the model. This setting is characterized as zero-data learning or (Zero-Shot Learning) ZSL [135]. Since some of the classes are unseen in ZSL setting, the performance of deep models is generally not very high.

In ZSL problems, data set $X$ is split into two sets with zero intersection in classes, known as seen/training and unseen/testing sets. On the given seen/unseen sets, two types of protocols have been used to evaluate ZSL algorithms: 1) conventional ZSL: assign a label for test samples that

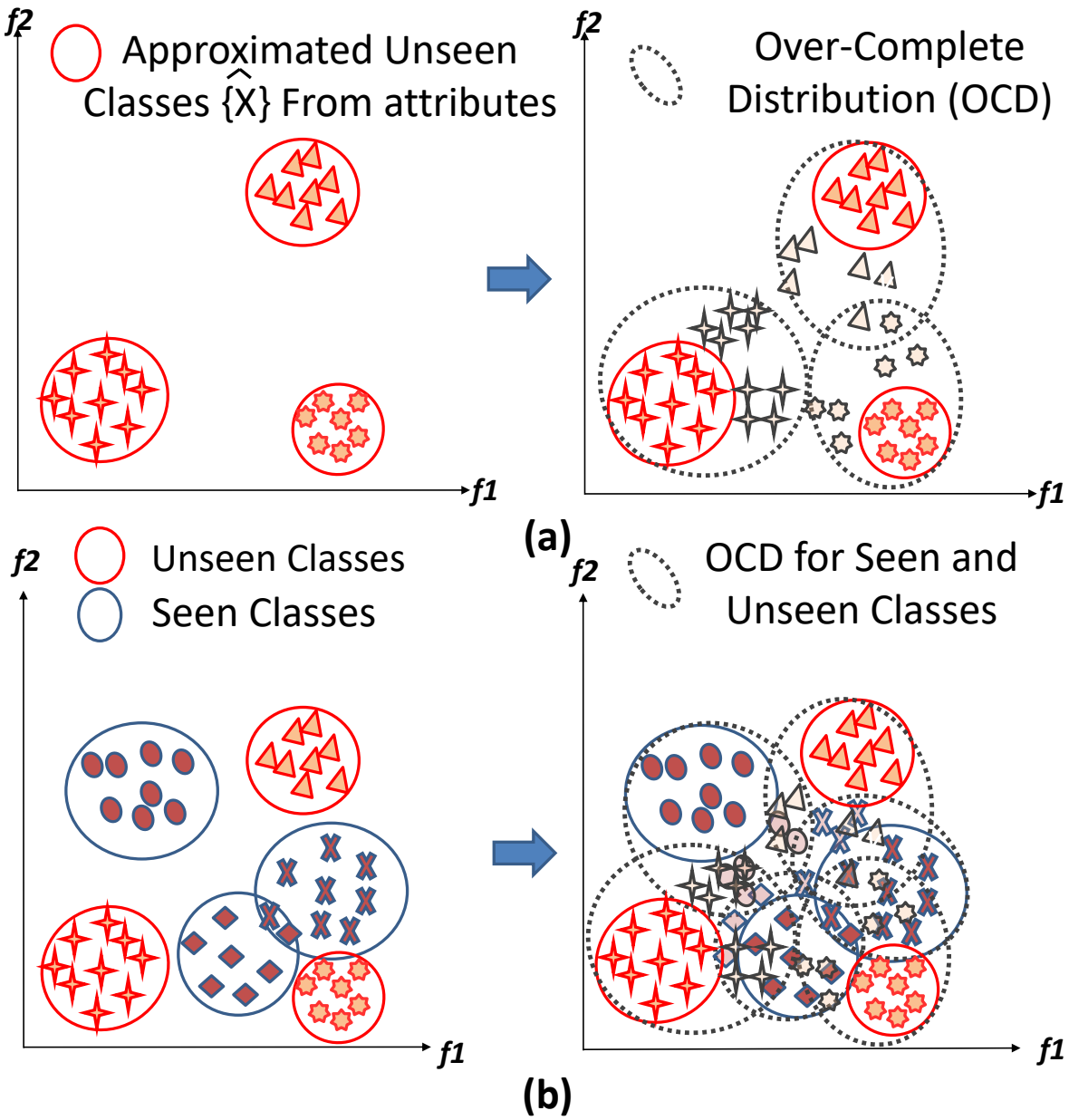



Figure 4-1: Illustration of seen and unseen 2D distribution before and after generation of Over-Complete Distribution (OCD). $f1$ and $f2$ are two dimensions of the data. (a) Shows distributions of three approximated unseen classes and generated OCDs for corresponding classes. (b) Shows three approximated unseen and seen class distributions and generated OCDs for corresponding classes. (Best viewed in colour)

belong to unseen classes, and 2) (Generalized Zero-Shot Learning) GZSL [282]: assign a label for test samples which belong to either seen or unseen classes. In both problems, training a network on seen samples leads to a biased effect toward seen distribution.

One of the main reasons for performance degradation in ZSL is that the testing set contains confusing *hard* samples (closer to another class), and the decision boundary is not optimized on the test set distribution. Therefore, generating hard samples and approximating unseen classes lead the network to reduce the bias. To address the challenges of ZSL, researchers have proposed to generate samples pertaining to unseen distribution synthetically [71], [153], [129], [300]. However, the generated unseen classes fail to mimic the real unseen distribution, which is expected to have hard samples. Hence, utilizing synthetically generated classes for training the discriminative classifier might not improve the performance. In this research, we propose the concept of Over-Complete Distribution (OCD) and then generate samples of classes corresponding to the OCD. Secondly, as shown in Figure 4-1, we propose to utilize Online Batch Triplet Loss (OBTL) to enforce separability between classes and Center Loss (CL) to reduce the spread within the class. We experimentally demonstrate that synthetically generated over-complete distribution allows the classifier to learn a feature space where the separability of seen/unseen classes can be efficiently improved.

## 4.2 Related Work

In ZSL, Larochelle *et al.* [135] have proposed to learn a mapping from the input space view to the model space view. Similarly, Akata *et al.* [2] have suggested embedding each class into the attribute vector space, called Attribute Label Embedding (ALE). Liu *et al.* [153] have proposed a Deep Calibration Network (DCN) for learning the common embedding space between the visual features of an image and the semantic representation of its respective class. A widely used method to tackle the ZSL problem is to learn a mapping between seen observation and the attribute vector space. Lampert *et al.* [134] proposed Direct Attribute Prediction (DAP), where a weighted probabilistic classifier has been trained for each attribute. After learning sample-to-attribute mapping, Baye's rule was applied to map attributes to the class label. Xian *et al.* [280] proposed the more challenging protocol and demonstrated that existing state-of-the-art (SOTA) algorithms do not perform well.

In GZSL, researchers have utilized the generated unseen classes to have representative data in the training set [160], [175]. [129] have proposed a generative model based on conditional varia-

tional autoencoder. They have shown that the synthetically generated unseen distribution is closely approximated to the real unseen data distribution. They have trained supervised linear SVM on synthetically generated data and shown state-of-the-art performance on GZSL protocol. Similarly, Gao *et al.* [71] have proposed synthesizing the unseen data using a joint generative model. They have used (Conditional Variational Autoencoder) CVAE and GAN and observed that preserving the semantic similarities in the reconstruction phase can improve the model's performance.

Zhang *et al.* [300] have proposed a hybrid model consisting of conditional Generative Adversarial Network (cGAN) and Random Attribute Selection (RAS) for the synthesized data generation. They have trained the hybrid model while optimizing the reconstruction loss. Zhang *et al.* [299] observed that the performance of conventional zero-shot learning algorithms suffers due to Class-level Over-fitting (CO) when they are evaluated for the GZSL task. To overcome the CO problem, they have utilized the triplet loss, which significantly outperforms the state-of-art methods. In another research direction, Long *et al.* [160] have proposed Unseen Visual Data Synthesis (UVDS) for generating synthesized classes from semantic attributes information. The authors have also proposed Diffusion Regularization (DR), which helps reduce redundant correlation in the attribute space. Atzmon and Chechik [8] have presented adaptive confidence smoothing for GZSL problem. They have utilized three classifiers, seen, unseen, and gating experts, to improve the model performance. Huang *et al.* [105] have proposed a generative dual adversarial network for learning a mapping function semantic to visual space. Schonfeld *et al.* [220] have proposed to align the distribution generated from VAE and show the improvement in benchmarking databases.

Significant effort has been made to generate an unseen synthetic distribution that has been further utilized for training the model. However, as discussed before, there are still challenges to be addressed to improve the performance on ZSL/GZSL problems, such as generalization of the model on the test set and reducing the bias for both seen and unseen classes.

## 4.3 Proposed Framework

Figure 4-2 demonstrates the steps involved in the proposed framework. For a given input $x$ with associated attribute $a$ and latent variable $z$, there are three modules in the proposed pipeline: (i) an encoder ($p_E(z|x)$) to compute the latent variables $z$ on given $x$, (ii) a decoder ($p_G(\hat{x}|z, a)$) to

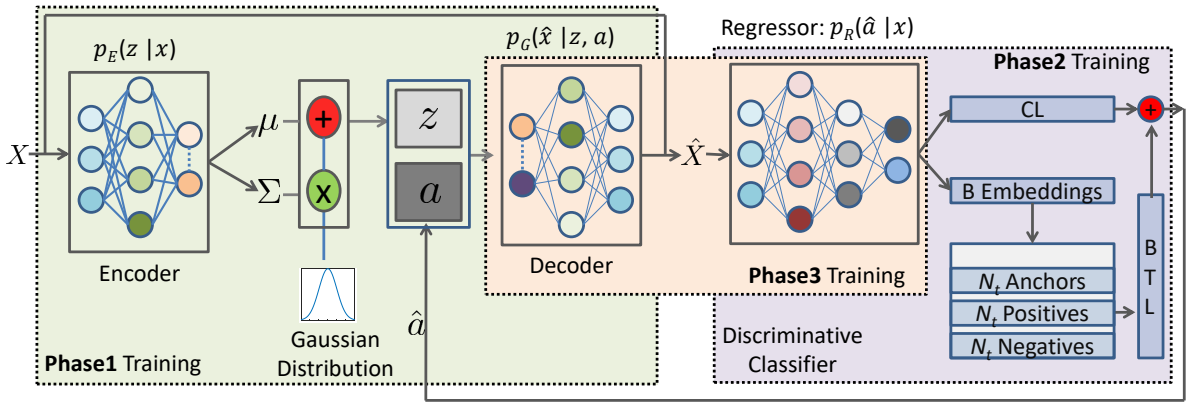



Figure 4-2: Illustration of the proposed OCD-CVAE framework. The framework uses CVAE with encoder $p_E(z|x)$ and decoder $p_G(\hat{x}|z,a)$ modules. The output of CVAE is given to the regressor $p_R(\hat{a}|x)$, where the regressor maps generate samples to their respective attributes. To generate the synthetic unseen data, attributes of unseen samples and randomly sampled $z$ have been provided to the trained decoder.

generate samples $\hat{x}$ on given $z$ and attribute $a$, and (iii) a regressor ($p_R(\hat{a}|\hat{x})$) to map $x$ to their predicted attribute $\hat{a}$.

The proposed framework can be divided into two parts: 1) generation of OCD, and 2) utilization of OBTL and CL to increase the separability between classes and to reduce the spread within the class, respectively. Moreover, training the proposed framework has been divided into three phases. In the first and second training phases, CVAE and regressor are trained and treated as pre-trained models for the third training phase.

### 4.3.1 Preliminaries

The proposed OBTL is a modified version of triplet loss. Along with triplet loss, center loss has been widely utilized in the metric learning literature. Therefore, we first briefly introduce both loss functions.

**Triplet Loss:** The triplet loss can be used to increase the inter-class distance and decrease the intra-class distance. Mathematically, triplet loss can be represented as:

$$\mathcal{L}_t(f^a, f^p, f^n) = \sum_{i=1}^{N} \Big[ ||f_i^a - f_i^p||_2^2 - ||f_i^a - f_i^n||_2^2 + \alpha \Big]_+ \tag{4.1}$$

where, $f$ represents the embedded feature vector, $\mathcal{L}_t$ is the triplet loss, and triplet $(f^a, f^p, f^n)$ is a 3-tuple of anchor, positive, and negative, respectively. A positive sample belongs to the same class as the anchor, and a negative sample is one that does not belong does not belong to the same class as the anchor. $\alpha$ is a margin to control the distance between the positive and negative pairs. $[]_+ = max(0, .)$ represents the hinge loss function.

From Equation 4.1, it can be observed that $\mathcal{L}_t(f^a, f^p, f^n) > 0$ only if $||f^a - f^p||_2^2 + \alpha > ||f^a - f^n||_2^2$. Therefore, hard triplet[1] mining is an essential step to minimize triplet loss.

**Center Loss:** Mapping a sample to their attributes has been used to find a solution for the ZSL problem. To learn the mapping of different samples to the attribute of a class, the standard deviation of a class distribution in the attribute space should be minimal. Therefore, center loss [273] and regressor loss [129] have been utilized to minimize the deviation from the center. As shown in Figure 4-2, regression $p_R(a|x)$ tries to map the approximated $\hat{x}$ to $a$. Minimizing the center loss is required for the over-complete distribution since hard samples increase the standard deviation. For the deviation of samples from the center of the distribution, the discriminative classifier is trained with center loss $\mathcal{L}_{CL}$:

$$\mathcal{L}_{CL} = \frac{1}{2} \sum_{c=1}^{S+U} ||x_c - x_c^{CT}||_2^2 \tag{4.2}$$

where, $x_c$ is a sample of class $c$, and $x_c^{CT}$ is a learned center of the class $c$.

### 4.3.2 Over-Complete Distribution (OCD)

The primary task of the decoder (shown in Figure 4-2) is to generate or approximate a distribution closer to the real unseen data. Since simulating the behavior of real unseen distribution is a challenging problem, we first propose to generate OCD for a class and visually show that the generated OCD simulates the behavior of the real unseen distribution.

Using the given distribution, OCD is generated by mixing a finite number of multiple Gaussian distributions [207] while shifting the mean towards other classes. If the distribution is unknown (in

[1]Inter-class distance $D_{inter}$ (distance between anchor A and negative sample N) should be maximized, and intra-class distance $D_{intra}$ (distance between anchor A and positive sample P) should be minimized. If $D_{inter} < D_{intra}$ then triplet $< A, P, N >$ is a hard triplet.

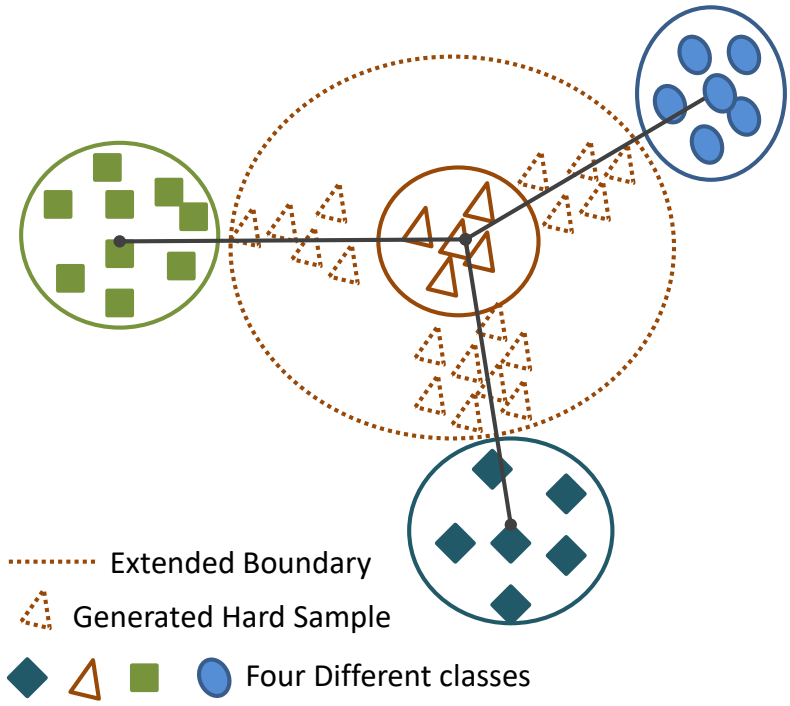



Figure 4-3: Illustration of over-complete distribution generation while generating hard samples between two classes.

the case of unseen classes), the distribution of the class can be approximated by using generative models. The parameters of approximated distribution from the variational inference of a class are represented as $\mu$, $\sigma$ and for the over-complete distribution are $\mu_{OC}$, $\sigma_{OC}$, and $\sigma_{OC} > \sigma$. We have termed it over-complete since the real distribution of a class might lack hard samples for each class of the database.

As shown in Figure 4-3, creating the OCD for a class implies having all the possible hard samples generated closer to other class distributions. Let $\hat{X}$ and $\hat{X}_{OC}$ be the approximated unseen distribution and over-complete distribution, respectively.

$$\hat{X} = p_G(x|\mathcal{N}(\mu_{HP}, \sigma_{HP}), a) \;\; and \;\; \hat{Z} = p_E(z|\hat{x}) \;\; , where \;\; \hat{x} \sim \hat{X}, \;\; \mu_{z|\hat{X}}, \;\; \sigma_{z|\hat{X}} \quad (4.3)$$

$$\hat{X}_{OC} = p_G(x|\mathcal{N}(\mu_{OC}, \sigma^{'}_{HP}), a), \;\; where \;\; \mu_{OC} = \frac{\mu_{z|\hat{X}} + \mu^{'}_{z|\hat{X}}}{2}, \;\; and \;\; \mu^{'}_{z|\hat{X}} = \mu_{z|\hat{X}}[j] \quad (4.4)$$

Equations 4.3 and 4.4 represent the over-complete distribution process. Here, $p_G(.)$ is a generator module of the pipeline. $\mu_{HP}$, and $\sigma_{HP}$ are hyper-parameters for normal distribution. $\mu_{z|\hat{X}}$ and $\sigma_{z|\hat{X}}$ are mean and standard deviation obtained while encoding the data $\hat{X}$ into the latent space $z$. In Equation 4.4, $\sigma^{'}_{HP}$ is a hyper-parameter, and $j$ is a randomly sampled index variable for the shuffling of the parameter $\mu_{z|\hat{X}}$. In both the equations, $\mathcal{N}(.)$ is a Gaussian distribution generator.

In the first part of Equation 4.3, distribution of unseen classes, $\hat{X}$, is generated by randomly sampling $z \sim \mathcal{N}(\mu_{HP}, \sigma_{HP})$ having $\mu_{HP}$, $\sigma_{HP}$ as parameters of the distribution and unseen class attributes $a$. In the second part of Equation 4.3, $\mu_{z|\hat{X}}$ and $\sigma_{z|\hat{X}}$ are estimated by using the encoder module $p_E(.)$.

The first part of the Equation 4.4 represents the generation of over-complete distribution $\hat{X}_{OC}$ in which latent variable $z \sim \mathcal{N}(\mu_{OC}, \sigma^{'}_{HP})$ is randomly sampled from the Gaussian distribution, where, mean of the distribution $\mu_{OC}$ is estimated by the average of the current and each competing class. For example, on a given batch of $\mu$, $j$ is an index variable ranging from $1, ..., batch\ size$ and is randomly sampled without repetition.

In our approach, the decoder/generator $P_G(x|z, a)$ is conditioned on attributes and used in Equations 4.3 and 4.4. In ZSL problems, attributes are assumed to be a good representation of a class and are separated in the attribute space. Within a class, if a sample is away from the center of the distribution, then it can be considered a hard sample. However, the attributes of the sample should be the same as the attributes of a class. Therefore, while generating OCD, attributes of a class are kept unchanged. On the other hand, the latent variable $z$ has been changed based on the mean parameter of other classes.

**Visualization of the Distributions:** As shown in Figure 4-4(a), in the real distribution of unseen classes, some of the samples are closer to another class distribution. As shown in Figure 4-4(b), the unseen distribution predicted/generated via CVAE is well-separated. If the generated distribution fails to mimic the behavior of real distribution, then utilization of such distribution might not be useful in the training phase. Usually, the discriminative classifier trained on such distribution performs poorly on unseen classes. On the other hand, learning class separability by maximizing inter-class distance and minimizing intra-class distance might provide a viable solution when the behavior of the training set is close to the test set. In the case of ZSL, distribution is unknown, and approximating unknown distribution where latent variables are sampled from Gaussian distribution

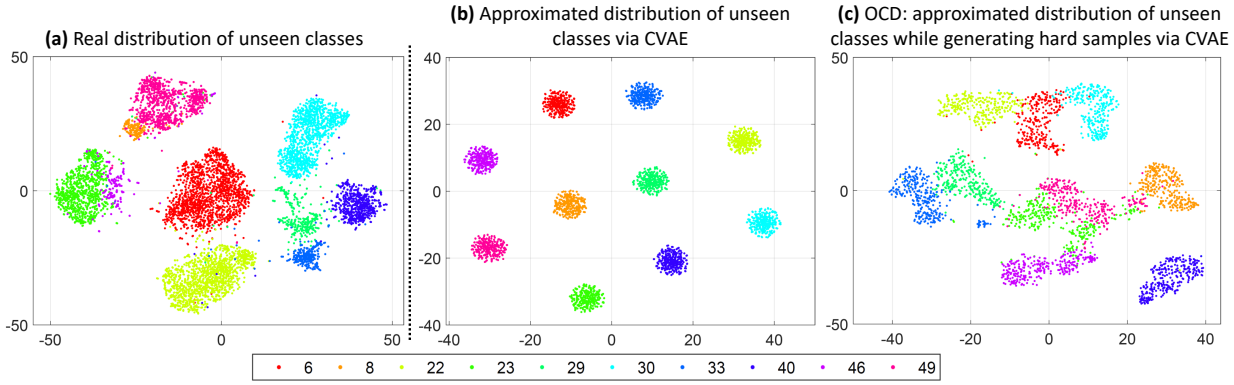



Figure 4-4: Illustration of synthetically generated distributions (a) and (b) for unseen classes of the AWA2 database. (c) is a real unseen distribution of the AWA2 database. Different colors of distribution represent different classes of the AWA2 database when the PS protocol has been used. The distribution plot shows that (b) is a closer representation of the real unseen class distribution (c). (Best viewed in colour)

creates blind spots in the feature space. As observed in Figure 4-4(b), the blind spot is the place where samples are not present, and training a classifier on such separable data would not ensure the model learns transformation so that the real unseen distribution would be separated. Figure 4-4(c) illustrates the OCD, which is an approximated distribution of unseen classes while generating hard samples via CVAE. Experimentally, we have shown that training using such distribution can improve classification performance.

### 4.3.3 Applying Loss Functions to Train the Proposed OCD-CVAE Framework

We propose to train OCD-CVAE framework in three phases. In the first phase, CAVE loss $\mathcal{L}_{CVAE}$, is optimized. In the second phase, OBTL loss along with center loss, i.e., $\mathcal{L}_{OBTL} + \mathcal{L}_{CL}$, is minimized. The trained model is then utilized as a pre-trained model for the third phase, where we propose training the regressor on the generated OCD while minimizing the OBTL and CL losses. The embedding space created from the output of the regressor increases the inter-class distance and reduces the intra-class distance.

### Online Batch Triplet Loss to Maximize Inter Class Distance

In the ZSL problem, generative classifiers suffer from the challenges of bias towards the seen classes. As shown in Figure 4-3, the generated hard samples between two classes lead to generating OCD for a class. The approximated OCD is then utilized for training a discriminative classifier for triplet loss minimization. Selecting $\mathcal{N}_t$ hard triplets in offline mode requires processing all the generated triplets in a single epoch, which is very challenging in real-world ZSL settings. Therefore, proposed $\mathcal{L}_{OBTL}$ minimizes the generated triplets for every batch while training the model. $\mathcal{L}_{OBTL}$ is optimized in a manner similar to $\mathcal{L}_t$ as defined in Equation 4.1.

Processing one batch and generating triplets would reduce the total training time of the deep model[2]. The proposed OBTL loss is inspired by the online triplet loss [6]. Generating triplets batch-wise reduces the search space to find hard negative samples. Synthetically generating hard negatives can improve the learning process of the deep model.

### Learning Phase of the Proposed Model

As shown in Figure 4-2, the learning phase of the proposed framework can be divided into three phases. In the *first phase*, encoder followed by decoder CVAE is trained using KL-divergence and conditional marginal likelihood. In the *second phase*, regressor/classifier is trained using the proposed OBTL along with CL. In the *third phase*, the decoder/generator and the regressor have been trained while minimizing OBTL, CL, and discriminator driven losses [129].

Let the training set contain '$S$' seen classes and the testing set contain '$U$' unseen classes, where their respective class attributes are represented as $\{a_c\}_{c=1}^{S+U}$ where, $a_c \in \mathbb{R}^L$, $L$ is the length of attributes. Training $\mathcal{D}_S$ and testing $\mathcal{D}_U$ sets can be represented as the triplet of data, attributes, and label $\{X_s, a_s, y_s\}_{s=1}^{S}$ and $\{X_u, a_u, y_u\}_{u=1}^{U}$, respectively. On the aforementioned settings, ZSL algorithm aims to build a classification model on $\mathcal{D}_S$ which can learn a mapping function $f : \mathcal{X}_U \rightarrow \mathcal{Y}_U$ where, $\mathcal{X}_U = \{X_u\}_{u=1}^{U}$ is a set of unseen samples and $\mathcal{Y}_U = \{y_u\}_{u=1}^{U}$ is the corresponding class set [2, 133].

[2]For instance, we have 20 samples per class from 10 classes in the dataset. Selecting every combination of 2 images from each class for the anchor and positive images and selecting a hard-negative from the remaining images gives $10 \times (C_2^{20})$ = 1900 triplets. Despite 200 unique samples, it requires 19 forward and backward passes to process 100 triplets simultaneously. In OBTL, these embeddings are mapped to 1900 triplets that are passed to the triplet loss function, and then the derivative is mapped back through to the original sample for the backward network pass - all with a single forward and single backward pass.

**First phase of training:** In the first phase of training, CVAE is trained on $\mathcal{D}_S$, where the input sample is $x_i$ for the encoder which encodes the latent variable $z_i$. The encoded variable is appended with the attributes $a_i$ of the corresponding sample. The appended latent variable $[z_i, a_i]$ is then provided to the generator module that generates the outputs $\hat{x}_i$ for a particular distribution close to the input provided to the encoder module. Trained CVAE allows the decoder to generate synthetic data on given attributes $a$. The CVAE loss $\mathcal{L}_{CVAE}$ can be defined as:

$$\mathcal{L}_{CVAE} = -\mathbb{E}_{p_{E(z|x)}, p_{(a|x)}}[log_{p_G}(\hat{x}|z, a)] + KL(p_E(z|x)||p(z)) \tag{4.5}$$

where, $-\mathbb{E}_{p_{E(z|x)}, p_{(a|x)}}[log_{p_G}(\hat{x}|z, a)]$ is the conditional marginal likelihood and $KL(p_E(z|x)||p(z))$ is the KL-divergence. Inspired from [102], the joint distribution over the latent code $[z, a]$ is factorized into two components $p_E(z|x)$ and $p_R(\hat{a}|x)$ as a disentangled representation.

**Second phase of training:** In the second phase of training, the regressor is trained on $\mathcal{D}_S$ while minimizing two the losses

$$\min_{\theta_R} \mathcal{L}_{OBTL} + \mathcal{L}_{CL} \tag{4.6}$$

The regressor is trained to improve the mapping of generated synthetic data to the corresponding attribute.

**Third phase of training:** From the first phase, we obtain $\theta_G$ (the generator parameter), which is used in the third phase of training. In the third phase, loss $\mathcal{L}_c(\theta_G)$ is based on the discriminator prediction, and $\mathcal{L}_{Reg}(\theta_G)$ is used as a regularizer.

$$\begin{aligned} \mathcal{L}_c(\theta_G) &= -\mathbb{E}_{p_G(\hat{x}|z,a)p(z)p(a)}[log\ p_R(a|\hat{x})] \\ \mathcal{L}_{Reg}(\theta_G) &= -\mathbb{E}_{p(z)p(a)}[log\ p_G(\hat{x}|z, a)] \end{aligned} \tag{4.7}$$

This regularization is used to ensure that the generated OCD from the generator produces class-specific samples even if $z$ is randomly sampled from $p(z)$. The complete objective function of the

third phase can be represented as:

$$\min_{\theta_G,\theta_R} \left(\lambda_c.\mathcal{L}_c + \lambda_{reg}.\mathcal{L}_{Reg} + \mathcal{L}_{OBTL} + \mathcal{L}_{CL}\right) \tag{4.8}$$

where, $\lambda_c$, and $\lambda_{reg}$ are the hyper-parameters.

### 4.3.4 Implementation Details

Experiments are performed on a 1080Ti GPU using Tensorflow-1.12.0 [1]. Hyper-parameters for CVAE learning: $\lambda_c = 0.1$, $\lambda_R = 0.1$, $\lambda_{reg} = 0.1$, and $batch\ size = 256$. To generate hard samples, the value of hyper-parameters $\mu_{HP}$, $\sigma_{HP}$ and $\sigma^{'}_{HP}$ in Equations 4.3 and 4.4 are $0$, $0.12$, and $0.5$, respectively. In our experiments, the size of $\mu$ is $256 \times 100$, and row-wise shuffling is performed. For reproducibility, we will release the model files of the proposed algorithm.

## 4.4 Experimental Results and Analysis

The proposed framework is evaluated on both ZSL and GZSL settings and compared with recent state-of-the-art algorithms for both. This section briefly presents the databases and evaluation protocols, followed by the results and analysis on the AWA2 [134], CUB [272], and SUN [196] benchmarking databases.

### 4.4.1 Database Details

The statistics and protocols of the databases are presented in Table 4.1. All databases have seen/unseen splits and attributes of the corresponding classes. The Animals with Attributes2 (AWA2) [134] is the extension of the AWA [133] database containing $37,322$ samples. The number of classes $(50)$ and size of the attribute vector $(85)$ are consistent with AWA. Human experts manually mark the $85$ dimensional attributes. The Caltech UCSD Bird 200 (CUB) [272] database contains $11,788$ fine-grained images of $200$ bird species. The size of the attribute vector is 312. The SUN Scene Classification (SUN) [196] database contains $14,204$ samples of 717 scenes. It has an attribute vector of $102$ length.

Table 4.1: Databases used in the experiments.

| Dataset | Seen/Unseen Classes | #Image | Attribute/Dim |
|---|---|---|---|
| SUN | 645/72 | 14340 | A/102 |
| CUB | 150/50 | 11788 | A/312 |
| AWA2 | 40/10 | 37322 | A/85 |

### 4.4.2 Evaluation Protocol

In ZSL, OCD for unseen classes have been generated and utilized to train the proposed OCD+CVAE framework. The results are reported on both standard split (SS) given by Larochelle *et al.* [135] proposed split (PS) given by Xian *et al.* [280] protocols. Unseen class classification accuracies have been reported in Table 4.2 for both PS and SS protocols. For GZSL, the seen classes of the dataset are divided into 80-20 train-test ratio to obtain $X^S_{train}$ and $X^S_{test}$ sets. $S+U$ is a set used for the training where the generator module of the proposed framework has synthetically generated $U$. The model is tested on $X^U$ and $X^S_{test}$. In GZSL, as defined in the literature, average class accuracies of protocol **A** and **B** are reported in Table 4.4. Protocol **A** is an average per-class classification accuracy on $X^U_{test}$ where, a classifier is trained on $S+U$ classes ($\mathbf{A}: U \rightarrow S+U$). Protocol **B** is an average per-class classification accuracy on $X^S_{test}$ where, a classifier is trained for $S+U$ classes ($\mathbf{B}: S \rightarrow S+U$). The above mentioned protocols are predefined for AWA2 [134], CUB [272], and SUN [196] databases and widely used to evaluate ZSL/GZSL algorithms.

### 4.4.3 Conventional Zero-Shot Learning (ZSL)

Table 4.2 summarizes the results of conventional zero-shot learning. The train split of all three datasets has been used to optimize the proposed framework. Synthetic hard samples are generated between unseen classes for the ZSL problem. The classification accuracies obtained on the PS protocol on AWA2, CUB, and SUN databases are $71.3\%$, $60.3\%$, and $63.5\%$, respectively. The proposed framework has improved the state-of-art performance on AWA2, SUN, and CUB databases by $1.8\%$, $0.7\%$, and $0.1\%$. The McNemar Test [170] estimates whether this difference is significant. Keeping a significance threshold of 0.05, or 5%, we observed that the null hypothesis is rejected for AWA2 and CUB databases. However, the null hypothesis has been accepted for the SUN database; thereby, the difference is statistically significant in the case of AWA2 and

Table 4.2: Classification accuracy (%) of conventional zero-shot learning for standard split (SS) and proposed split (PS) [280]. (Top two performances are in bold)

| Method | AWA2 | | CUB | | SUN | |
|---|---|---|---|---|---|---|
| | **SS** | **PS** | **SS** | **PS** | **SS** | **PS** |
| CONSE [185] | 67.9 | 44.5 | 36.7 | 34.3 | 44.2 | 38.8 |
| SSE [304] | 67.5 | 61.0 | 43.7 | 43.9 | 54.5 | 51.5 |
| LATEM [279] | 68.7 | 55.8 | 49.4 | 49.3 | 56.9 | 55.3 |
| ALE [2] | 80.3 | 62.5 | 53.2 | 54.9 | 59.1 | 58.1 |
| DEVISE [66] | 68.6 | 59.7 | 53.2 | 52.0 | 57.5 | 56.5 |
| SJE [3] | 69.5 | 61.9 | 55.3 | 53.9 | 57.1 | 53.7 |
| ESZSL [208] | 75.6 | 58.6 | 55.1 | 53.9 | 57.3 | 54.5 |
| SYNC [28] | 71.2 | 46.6 | 54.1 | 55.6 | 59.1 | 56.3 |
| SAE [123] | 80.2 | 54.1 | 33.4 | 33.3 | 42.4 | 40.3 |
| SSZSL [83] | - | - | 55.8 | - | - | - |
| GVRZSC [20] | - | - | 60.1 | - | - | - |
| GFZSL [258] | 79.3 | 67.0 | 53.0 | 49.2 | 62.9 | 62.6 |
| CVAE-ZSL [175] | - | 65.8 | - | 52.1 | - | 61.7 |
| SE-ZSL [129] | **80.8** | 69.2 | **60.3** | **59.6** | 64.5 | **63.4** |
| DCN [153] | - | - | 55.6 | 56.2 | **67.4** | 61.8 |
| JGM-ZSL [71] | - | **69.5** | - | 54.9 | - | 59.0 |
| RAS+cGAN [300] | - | - | - | 52.6 | - | 61.7 |
| **Proposed** | **81.7** | **71.3** | **60.8** | **60.3** | **68.9** | **63.5** |
| **Current Methods** | | | | | | |
| CCD [72] | - | 76.7 | - | 80.3 | - | 64.5 |

CUB databases. On the other hand, it is insignificant in the case of the SUN database. For the SS protocol, the classification accuracy on AWA2, CUB, and SUN databases are $81.2\%$, $60.8\%$, and $68.4\%$, respectively. In general, the proposed algorithm yields one of the best accuracies across the three databases compared to several existing approaches.

### 4.4.4 Ablative Study

The proposed framework OCD-CVAE has utilized multiple loss functions for improving the performance of ZSL/GZSL. An ablation study is conducted to evaluate the effectiveness of each of the components individually and in combination. Table 5.11 summarizes the results of five settings thus obtained OCD+OBTL+CL yields the best results followed by OCD+OBTL. It can be observed that only applying OBTL is performing poorly, and the primary reason is not to have

Table 4.3: Ablative study on 3 datasets with the PS protocol. The reported values are classification accuracy (%).

| | AWA2 | SUN | CUB |
|---|---|---|---|
| OBTL | 65.8 | 56.4 | 54.5 |
| CL | 65.3 | 56.2 | 53.7 |
| OCD+OBTL | 70.9 | 62 | 60.5 |
| OCD+CL | 66.5 | 57.6 | 56.8 |
| OCD+OBTL+CL | **71.3** | **62.1** | **60.9** |

Table 4.4: Average per-class classification accuracy (%) of generalized zero-shot learning when test samples can be from either seen (S) or unseen (U) classes. **A**: U→S+U, and **B**: S→S+U. (**Top two performances are highlighted)**

| **Type** | **Method** | **AWA2** | | | **CUB** | | | **SUN** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **A** | **B** | **H** | **A** | **B** | **H** | **A** | **B** | **H** |
| Non-Generative Models | CONSE [185] | 0.5 | 90.6 | 1.0 | 1.6 | 72.2 | 3.1 | 6.8 | 39.9 | 11.6 |
| | SSE [304] | 8.1 | 82.5 | 14.8 | 8.5 | 46.9 | 14.4 | 2.1 | 36.4 | 4.0 |
| | SJE [3] | 8.0 | 73.9 | 14.4 | 23.5 | 59.2 | 33.6 | 14.7 | 30.5 | 19.8 |
| | ESZSL [208] | 5.9 | 77.8 | 11.0 | 12.6 | 63.8 | 21.0 | 11.0 | 27.9 | 15.8 |
| | SYNC [28] | 10.0 | 90.5 | 18.0 | 11.5 | 70.9 | 19.8 | 7.9 | 43.3 | 13.4 |
| | SAE [123] | 1.1 | 82.2 | 2.2 | 7.8 | 54.0 | 13.6 | 8.8 | 18.0 | 11.8 |
| | LATEM [279] | 11.5 | 77.3 | 20.0 | 15.2 | 57.3 | 24.0 | 14.7 | 28.8 | 19.5 |
| | ALE [2] | 14.0 | 81.8 | 23.9 | 23.7 | 62.8 | 34.4 | 21.8 | 33.1 | 26.3 |
| | DCN [153] | - | - | - | 28.4 | 60.7 | 38.7 | 25.5 | 37.0 | 30.2 |
| | COSMO+LAGO [8] | 52.8 | 80.0 | 63.6 | 44.4 | 57.8 | 50.2 | 44.9 | 37.7 | 41.0 |
| | DEVISE [66] | 17.1 | 74.7 | 27.8 | 23.8 | 53.0 | 32.8 | 16.9 | 27.4 | 20.9 |
| Generative Models | CVAE-ZSL [175] | - | - | 51.2 | - | - | 34.5 | - | - | 26.7 |
| | SE-GZSL [129] | 58.3 | 68.1 | 62.8 | 41.5 | 53.3 | 46.7 | 40.9 | 30.5 | 34.9 |
| | JGM-ZSL [71] | 56.2 | 71.7 | 63.0 | 42.7 | 45.6 | 44.1 | 44.4 | 30.9 | 36.5 |
| | f-CLSWGAN [281] | - | - | - | 43.7 | 57.7 | 49.7 | 42.6 | 36.6 | 39.4 |
| | RAS+cGAN [300] | - | - | - | 31.5 | 40.2 | 35.3 | 41.2 | 26.7 | 32.4 |
| | CADA-VAE [220] | 55.8 | 75.0 | **63.9** | 51.6 | 53.5 | **52.4** | 47.2 | 35.7 | 40.6 |
| | GDAN [105] | 32.1 | 67.5 | 43.5 | 39.3 | 66.7 | 49.5 | 38.1 | 89.9 | **53.4** |
| | **Proposed** | 59.5 | 73.4 | **65.7** | 44.8 | 59.9 | **51.3** | 44.8 | 42.9 | **43.8** |
| | **Current Methods** | | | | | | | | | |
| | MSDN [31] | 62.0 | 74.5 | 67.7 | 68.7 | 67.5 | 68.1 | 52.2 | 34.2 | 41.3 |
| | HSVA [32] | 59.3 | 76.6 | 66.8 | 52.7 | 58.3 | 55.3 | 48.6 | 39.0 | 43.3 |
| | CCD [72] | 68.0 | 82.7 | 74.6 | 73.1 | 78.9 | 75.9 | 43.9 | 44.6 | 44.2 |
| I | SRWGAN [61] | 72.5 | 89.4 | 80.1 | 63.7 | 78.9 | 70.2 | 65.2 | 51.5 | 57.5 |
| T | SRWGAN [61] | 81.2 | 92.9 | 86.6 | 65.3 | 79.1 | 71.6 | 68.8 | 51.8 | 59.1 |

sufficient hard samples for the OBTL loss function to backpropagate the gradient.

### 4.4.5 Generalized Zero-Shot Learning (GZSL)

In GZSL, the testing samples can be from either seen or unseen classes. In Table 4.4, average per-class classification accuracy for the protocols $\mathbf{A}$ and $\mathbf{B}$ have been reported. The final accuracy is the harmonic mean of accuracies (represented as $\mathbf{H}$), which is computed by $2 \times \frac{\mathbf{A} \times \mathbf{B}}{\mathbf{A}+\mathbf{B}}$. This is a challenging setting where the train and test classes are not entirely disjoint, but the samples of the train and test sets are disjoint. Hence, the possibility of overlapping distribution and hard samples increases in the test set. Most of the ZSL algorithms perform poorly on GZSL. We assert that addressing this GZSL requires learning separability in the embedding space (output of the regressor).

Table 4.4 summarizes the results of existing algorithms on the three databases in GZSL settings. The algorithms are segregated into non-generative and generative models. Among the non-generative models, COSMO+AGO [8] yields the best performance. Considering all the algorithms, it can be observed that utilizing the approximated distribution of unseen class by generative models performs better than non-generative models. The proposed method also utilizes CVAE based generative model. We postulate that generating OCD on the train set and utilizing it to optimize the proposed framework leads the network to better generalize on the test set.

It can be observed from Table 4.4 that the proposed framework improves state-of-the-art harmonic mean accuracies $\mathbf{H}$ on the AWA2 dataset by $1.8\%$. The proposed algorithm is among the two best-performing algorithms on the SUN and CUB databases. It is worth mentioning that the GZSL is a challenging problem, and none of the algorithms have consistently outperformed on all three databases.

### 4.4.6 Hyper-Parameter Selection

Figure 4-5(a) shows the performance with increasing synthetically generated samples. It can be observed that when OCD is not used for training the regressor, increasing the number of samples does not affect the performance. With the use of OCD, generating $400$ to $600$ samples leads to improved performance. Tto determine the value of $\sigma^{'}_{HP}$ in Equation 4.4, we have explored the range

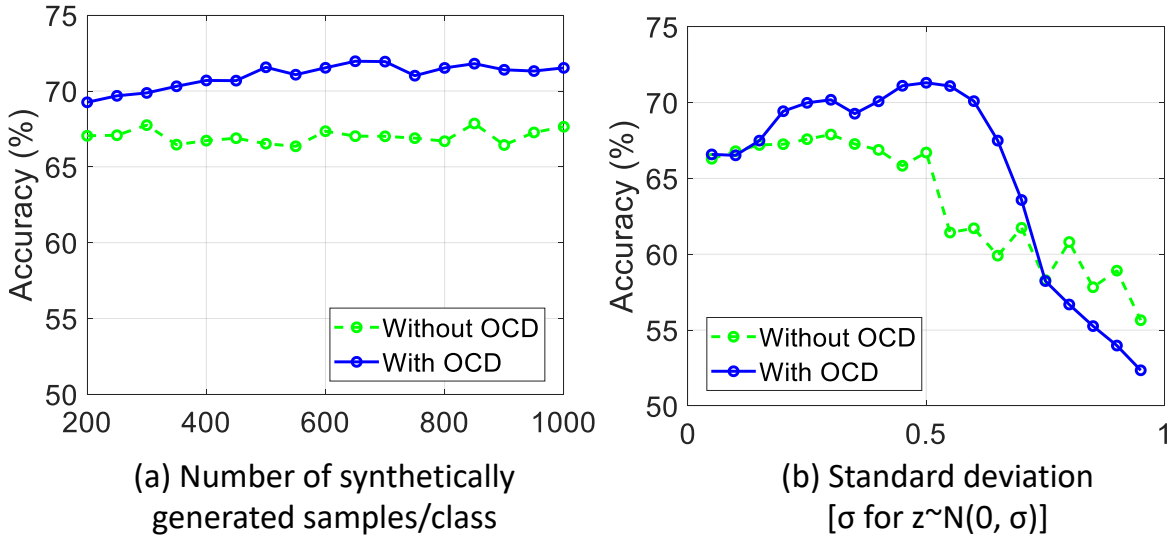



Figure 4-5: Accuracy plots on varying (a) the number of samples and (b) standard deviation on the AWA2 dataset.

of $\sigma'_{HP}$ from $0.05$, to $0.95$. As shown in Figure 4-5(b), it can be observed that the best performance has been achieved on $0.5$ standard deviation. The value of $\sigma_{HP}$ is chosen from a standard normal while $\sigma'_{HP}$ is computed using the PS split train set. The results in Figure 4-5 also demonstrate that the chosen value $(0.5)$ yields the best results. Most of the hyper-parameters are kept consistent with Verma *et al.* [129]. In OBTL loss, $\alpha$ parameter is computed during optimization on the train set and is set as $0.4$.

## 4.5 Summary

This paper addresses the challenge of Zero Shot Learning and Generalized Zero Shot Learning. We propose the concept of over-complete distribution and utilize it to train the discriminative classifier in ZSL and GZSL settings. The over-complete distribution is defined by generating all possible hard samples for a class closer to other competing classes. Experimentally, we have shown that learning a classifier on separable distribution would not generalize on test samples with the class-overlapping distributions. We have observed that over-complete distribution is helpful in ensuring separability between classes and improving classification performance. Experiments on three benchmark databases show that the proposed approach yields improved performance. The concept of OCD can also be utilized in other frameworks, such as Generative Adversarial

Networks, along with optimizing inter-class and intra-class distances.

# Chapter 5

# Guided DropBlock and Filter Augmentation in CNNs for Data Constrained Learning

*"A blocked path also offers guidance."*

— Mason Cooley

## 5.1 Introduction

Data-driven applications strive to automate a variety of real-world challenges by deriving insights and making decisions based on data. However, numerous real-world scenarios are still uncharted, particularly in resource-restricted environments, as highlighted in various studies [143, 265, 286]. These settings often face constraints regarding computational/hardware resources and data accessibility [136]. Hardware limitations may arise due to a lack of computational resources, such as the absence of GPUs or restrictions in network connectivity from edge computing devices to the cloud, often for security reasons associated with edge devices [252]. Conversely, the issue of data scarcity can result from the inherent characteristics of certain applications, such as the identification processes in newborns and pre-post cataract surgery scenarios, or limitations in the sensing devices themselves. This scenario presents a significant hurdle, as having sufficient high-quality training data is a crucial prerequisite for constructing deep learning systems. Over time, as advancements in research have occurred, meeting this prerequisite has become imperative for developing high-

performing deep learning models [91, 103].

Training deep learning models in a data-constrained environment (i.e., the training set has a limited number of samples) often results in a skewed distribution of gradients as the models struggle to fine-tune their millions of parameters. Consequently, these deep models exhibit poor generalization and convergence when applied to tasks with fewer samples in the dataset. This issue is particularly noticeable in scenarios such as small object detection [291], budgeted data classification [64], multi-task learning [277], high-resolution mapping [144], text to signature transformations [251], and distributed semi-supervised learning [154]. A variety of strategies have been proposed to mitigate the challenges posed by data-constrained environments. These include adjusting the adaptive learning rate [151], facilitating the transfer of learnable features [156], implementing rotation equivariant CNNs [257], augmenting the dataset to increase sample numbers [42, 127, 193, 215, 228, 239], utilizing models pre-trained on extensive datasets [118], adopting knowledge distillation techniques [92, 119], engaging in metric learning [75], and applying regularization methods [73, 233]. These approaches aim to alleviate the undergeneralization issue and enhance the performance of deep models in data-constrained scenarios.

Few-shot learning presents a unique challenge in data-constrained scenarios, where it is defined as an $m$-shot $n$-way classification task. In this context, $m$ represents the count of labeled instances for each novel category, and $n$ signifies the number of unfamiliar categories to be classified. Various strategies have been proposed to address this challenge, including approaches based on probability density [172], metric learning [163, 259], memory augmentation [219], and the generation of latent features [165]. A comprehensive review by [267] encapsulates the problem of few-shot learning, pinpointing the main issue as the model's poor generalization capabilities when only limited samples are available. There are two principal strategies to navigate the constraints imposed by limited data availability: A) augmenting the dataset to increase the number of samples and B) implementing robust regularization techniques. These approaches enhance the model's performance, making it more adaptable and reliable even when operating under data scarcity.

**A) Data Augmentation** directly manipulates the data before it is inputted into the model, allowing the model to extract relevant features from various transformations or perspectives of the data. Common augmentation techniques encompass rotation, flipping, histogram equalization,

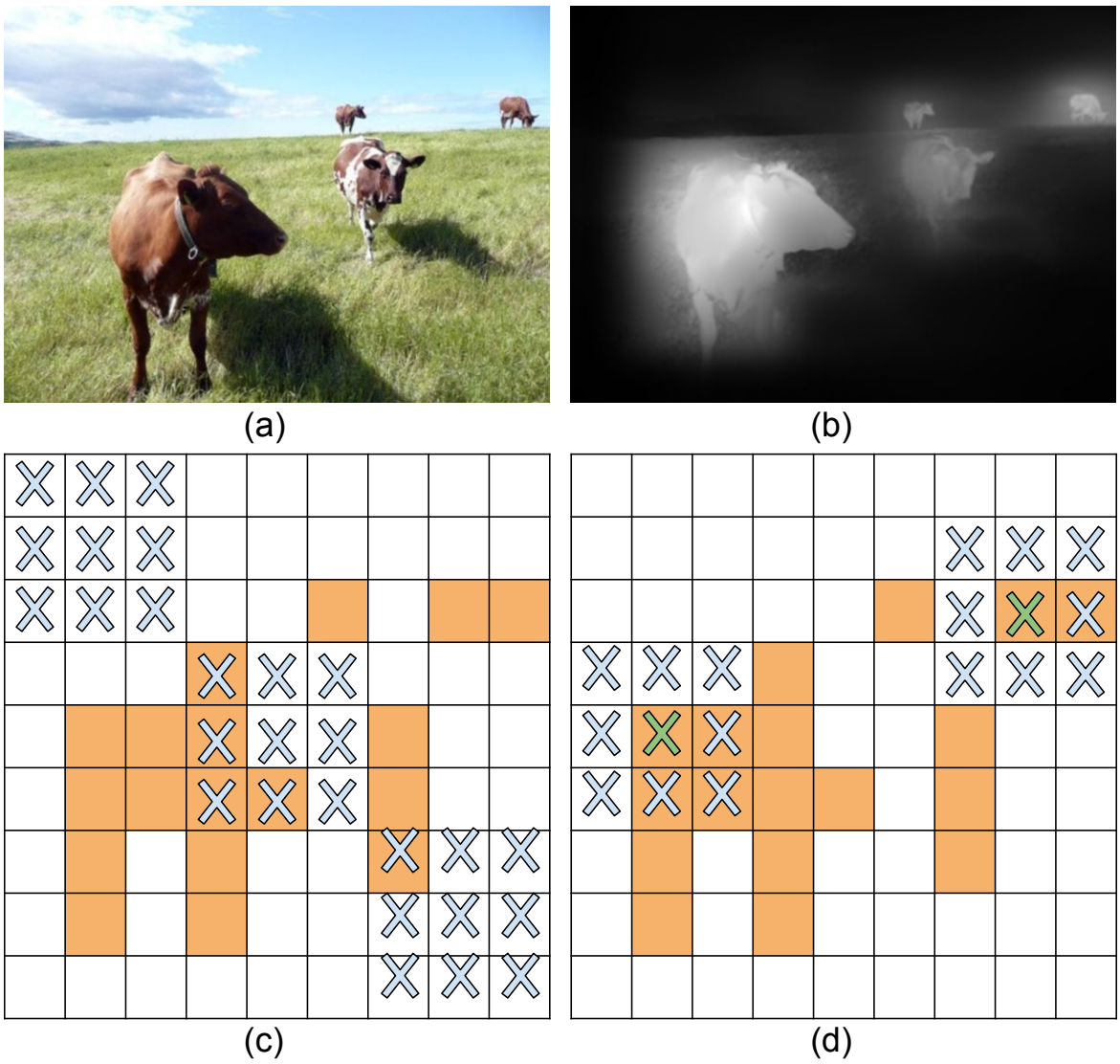



Figure 5-1: Figure Illustration of differences between DropBlock and Guided DropBlock. (a) an input image for a CNN model. The yellow regions in (c) and (d) include the activation units, which contain semantic information about cows shown in the input image (a) and the activation map (b). (c) In DropBlock, continuous regions containing semantic information like head or feet can be removed. If the object's size varies (as shown in (a), which has four cows at different distances, hence different sizes within an image), then dropping a random block can drop the whole object or miss it. Thus, it will lead to inefficiency in dropping semantic information. (d) Proposed Guided DropBlock, where continuous regions are selected from the active features shown in feature map (b).

cropping, CutMix, MixUp, and pixel corruption [161] operations. Enhancements in model performance are consistently observed across the board when these approaches are employed, as they enable the model to capture variability properties from both the original and augmented data

sets [43, 62, 148]. Nevertheless, it's crucial to note that CNNs generally struggle to handle variations in object views, often misclassifying objects when they appear in rotated forms [4]. There has been an exploration into adjusting the orientation and scale of filters [40, 74], aiming to equip CNNs with the capability to better recognize and classify objects, regardless of their orientation or scale in the input data.

**B) Regularization** plays a crucial role in mitigating poor generalization in deep learning models. Various adaptations of the traditional dropout regularization method [233] have been introduced in the literature. Among these, DropBlock [73] stands out as a regularization technique designed to eliminate semantic information, making it particularly suitable for use after convolutional layers in modern CNN architectures [125, 210, 264, 298]. To illustrate, consider Figure 5-1(a), which depicts an image featuring four cows at varying distances and, consequently, different sizes. Figure 5-1(b) represents the corresponding activation map. After translating the activation map into binary form, we derive the highlighted cells shown in Figure 5-1(c) and (d). In Figure 5-1(c), a $3\times3$ block in the top-left corner is randomly chosen. In the standard DropBlock method, continuous regions of the image, possibly containing vital semantic information like parts of the object, could be randomly removed. This method can be less efficient, especially when objects vary in scale, and the image has a significant background portion. To address these limitations, we introduce a modified version of the DropBlock regularizer named Guided DropBlock, aiming to enhance efficiency through guidance. As depicted in Figure 5-1(d), Guided DropBlock selectively removes continuous regions based on the active features highlighted in the feature map (Figure 5-1(b)), providing a semantic guide for more effective information dropping. Moreover, we incorporate fine-tuning a pre-trained model, freezing the initial layers of the CNN, and utilizing Guided DropBlock for regularization to optimize performance.

**Contributions:** Fine-tuning a pre-trained model in data-limited environments (previously trained on a comprehensive dataset) serves as a viable strategy to mitigate overfitting when data is scarce. As illustrated in Figure 5-2(a), a model undergoes training on a source database abundant with data adequate for training the model. Adapting the pre-trained model involves updating only the last few layers, making it challenging to determine the optimal filters for the target database. Conversely, in

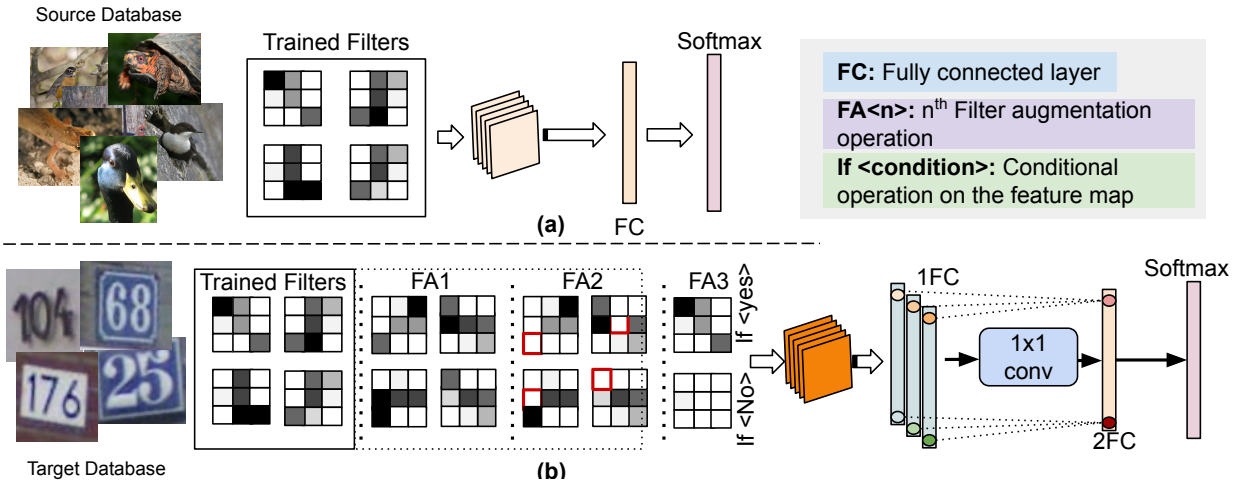



Figure 5-2: Illustrating CNN and the structure of trained filters. CNN contains multiple convolutional blocks followed by Fully Connected ($FC$) and $Softmax$ layers. (a) Visualization of trained filters in CNN framework, trained on the source database. (b) Visualization of three filter augmentation operations on pre-trained filters along with two FC layers. FA1: is a $90^{\circ}$ clockwise rotation operation, FA2: randomly drops some of the weight in a filter after FA1, and FA3: is a conditional operation applied when the condition is satisfied. Each Filter Augmentation (FA) operation leads to training the corresponding feature vector shown in the first FC layer. All feature vectors are combined via $1 \times 1$ $conv$ layer and produce the second FC layer used as an input for the softmax layer.

our proposed framework, as shown in Figure 5-2(b), the pre-trained model undergoes fine-tuning on the target dataset while undergoing three operations: FA1, FA2, and FA3[1], resulting in behavior akin to an ensemble model. This research focuses on the following:

1. Extraction of pertinent features for the target database, even when filters are frozen, to prevent overfitting due to limited data availability.

2. Addressing the research question: *"Is it feasible to implement augmentation operations on convolutional filters, enabling the extraction of apt features for the target database, even with a scarce number of samples?"*

3. We propose the integration of Guided DropBlock and Filter Augmentation (FA) within the CNN framework, introducing minimal overhead through the application of augmentation operations on filters.

[1]The number of FA operations can vary based on the selection of functions deemed suitable for the filters.

4. To substantiate our claims, extensive experiments were conducted, yielding results across seven databases and two real-world applications. Comparative analyses with existing state-of-the-art algorithms were also carried out to demonstrate the effectiveness of the proposed method.

## 5.2 Related Work

The related literature can be divided into two parts: 1) advancement in few-shot learning and 2) architectural similarity-based approaches.

**Advancements in Few-Shot Learning:** Generally, few-shot learning protocols are defined in terms of base and novel classes. Base classes usually have sufficient samples to train the model, and novel classes have fewer training examples. Therefore, the model should be generalized enough to perform well on novel classes. In literature, multiple promising solutions have been proposed. [219] have suggested using a memory-augmented-based neural network. Their approach utilizes gradient descent and slowly learns an abstract method for obtaining valuable representations of raw data and binding with a single presentation through an external memory module. [259] have proposed a non-parametric solution for few-shot learning, which utilizes a neural network (augmented with memory) and builds a function between small sample data to its classes.

[163] have utilized a CNN architecture and employed supervised and entropy losses to achieve multi-level domain transfer. Specifically, they have a semantic transfer by minimizing the pairwise entropy's similarity between unlabeled/labeled target images [118] have proposed a strength-based adaptation of CNN filters. While training, they froze the filters and learned only one parameter corresponding to each filter. The results showed improved performance and reduced the requirement for extensive training data. [199] have proposed a structured margin loss with a context-aware query embedding encoder and generates a highly discriminative feature space for discriminative embeddings. Their proposed task-dependent features help the meta-learner to learn a distribution over tasks more effectively.

[101] have proposed a novel transfer-based method that builds on two successive steps: 1) transform the feature vectors to Gaussian-like distributions, and 2) utilizing this pre-processed vectors via an optimal-transport inspired method which uses the Wasserstein distance [44]. [168]

have proposed an S2M2 framework that learns relevant feature manifolds for few-shot tasks using self-supervision and regularization techniques. They observed that regularizing the feature manifold, enriched via self-supervised methods, with Manifold Mixup significantly improves the performance of few-shot learning.

[306] have suggested utilizing adaptive kernels with random Fourier features in the meta-learning framework for few-shot learning called MetaVRF. They have formulated the optimization of MetaVRF as a variational inference problem and proposed a context inference of the posterior, established by an Long Short-Term Memory Networks (LSTM) architecture to integrate the context information of the previous task. [229] have proposed a framework by introducing subspace-based dynamic classifiers that use few samples. They have experimentally shown the efficacy of the proposed modeling on the task of supervised and semi-supervised few-shot classification. [284] have submitted Task-specific Discriminative Embeddings for Few-shot Learning (TDE-FSL) via utilizing the dictionary learning method for mapping the feature to a more discriminative subspace. It allows them to use unlabeled data. They obtained excellent performance on five benchmark image datasets.

**Architectural Similarity:** Analyzing the filters while applying operations such as orientation, scale, and structural change are widely utilized in computer vision [78] and now used to study CNN filters as well [290]. We want to mention some of the regularizations that could be related to "*filter augmentation*". In the *filter augmentation*, operations on the filters have been applied that change the network architecture and require a dedicated feature vector in the FC layer. [39] have proposed Group equivariant CNNs (G-CNNs) where a new layer called G-convolution has been used. In this new type of convolution, reflection, rotation, and translation operations have been utilized to have robust features. Extending the work of G-CNNs, Cohen, and Welling [40] have proposed steerable CNNs where a cyclic shift of convolutional kernel can extract rotation-invariant features. Similarly, [74] have utilized scale-steerable operation on the filters to make the CNN scale and rotation invariant.

[113] have proposed a locally scale-invariant CNN model and shown improvement on the MNIST-scale dataset. In some cases of *filter augmentation*, operations are not directly affecting network architecture and do not require a dedicated feature vector in the FC layer such as dropconnect [262], dropout [233] and its variants [9, 68, 116, 121]. Patch Gaussian Augmenta-

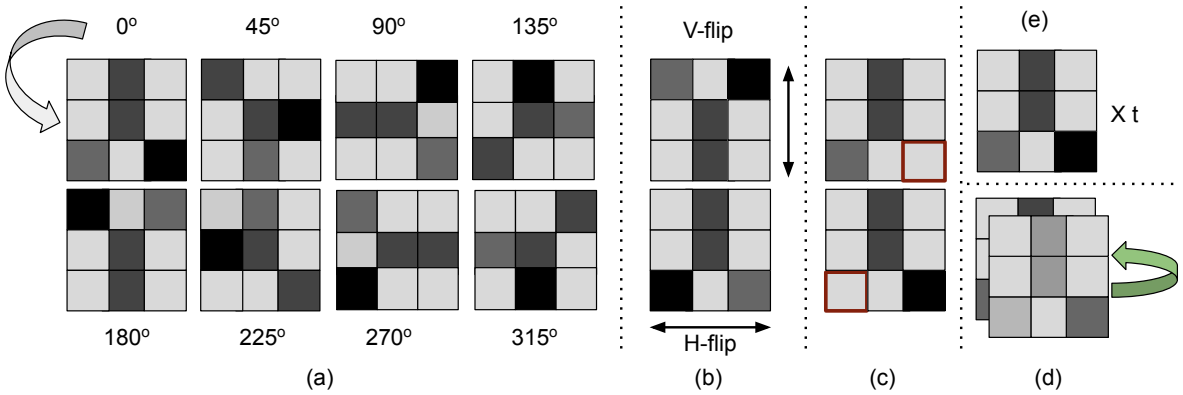



Figure 5-3: Illustration of filter augmentation operations in $3 \times 3$ filters: (a) Visualization of eight anti-clockwise rotation operations of intervals $45°$, (b) horizontal and vertical flip operation, (c) randomly drop some of the weight in a filter, (d) channel shuffling operation, (e) learning strength of filters.

tion [161] regularizers behave as an ensemble of filter augmentation operations when the deep models are trained for many epochs. Apart from filter augmentation operation, model adaptation is also widely used when the target database has limited samples [52, 223].

## 5.3 Proposed Guided DropBlock and Filter Augmentation

In this research, we introduce two novel contributions: 1) Guided DropBlock, a new regularization technique, and 2) filter augmentation, which involves applying transformations to pre-trained or previously learned filters. The proposed filter augmentation (FA) technique encompasses five distinct operations: 1) variation in strength, 2) rotation, 3) flipping, 4) channel shuffling, and 5) random elimination of filter weights. These proposed FA operations are visually depicted in Figure 5-3. This section is structured to provide preliminary information first, followed by detailed descriptions of the proposed Guided DropBlock, filter augmentation, and the associated implementation specifics.

### 5.3.1 Preliminary

Dropout is a prominent and extensively utilized regularization technique to address overfitting [233], with numerous variations introduced in subsequent literature. DropBlock [73] represents a novel iteration of this concept, tailored for semantic-aware learning applications. This method is specifi-

cally intended for use following the convolutional layer, where it operates by randomly eliminating block(s) of semantic information from the 2D input.

**Dropout and DropBlock:** In dropout [233], some of the nodes from the hidden layers are dropped at every epoch with probability $(1-\theta)$. Let $\ell$ be the $\ell^{th}$ layer of a network, where the value of $\ell$ ranges from 0 to $L$, $\ell_0$ represents the input layer, and $L$ is the number of hidden layers. Let the intermediate output of the network be $z^{(\ell)}$.

$$\widetilde{a}_j^{(\ell+1)} = r_j^{(\ell+1)} \odot a_j^{(\ell+1)} \tag{5.1}$$

where, $a_j^{(\ell+1)} = f(z_j^{(\ell+1)})$, and $z_j^{(\ell+1)} = w_{j\times i}^{(\ell+1)} a_i^{\ell} + b_j^{(\ell+1)}$. $i \in [1, ..., N_{in}]$, and $j \in [1, ..., N_{out}]$ are index variables for $N_{in}$ and $N_{out}$ at the $(\ell+1)^{th}$ layer, respectively. $f(.)$ is the $ReLU$ activation function. The conventional dropout randomly drops nodes using $Bernoulli$ distribution, where $\widetilde{a}$ is the masked output, $a^{(\ell)}$ is the intermediate output, $\odot$ is the element-wise multiplication, and $r^{(\ell)}$ is the dropout mask sampled from $Bernoulli$ distribution.

In case of "DropBlock", the outputs $\widetilde{\mathbf{a}}$ has been modified by $\mathbf{r} \odot \mathbf{a}$, where, $\mathbf{r}$, and $\mathbf{a}$ are four-dimensional tensors. $\mathbf{r}$ is randomly sampled from $Bernoulli$ distribution such that selected pixels of the features map along with neighboring pixels are dropped in a block-wise manner [73].

### 5.3.2 Guided DropBlock

First, we establish a connection between traditional dropout and "Filter Augmentation", demonstrating that DropBlock serves as an innovative approach to implement FA on convolutional filters. Subsequently, we delve into the functionality of "Guided DropBlock," which we introduce as a novel filter augmentation method. This technique is employed to train the model, as illustrated in Figure 5-3(h). The dropout Eq. 5.1 can be rewritten as: $\widetilde{a}_j^{(\ell+1)} = r_j^{(\ell+1)} \odot f(w_{j\times i}^{(\ell+1)} a_i^{\ell})$. The $Bernoulli$ variable $r$ is element-wise multiplied by the output of function $f$. Directly modifying the weight parameter $W$ with dropout mask $r$, the equation can be further derived as:

$$\widetilde{a}_j^{(\ell+1)} = f(\mathbf{W'}_j^{(\ell+1)} a_i^{\ell}), \quad \mathbf{W'} = \begin{cases} \mathbf{W} & if \quad r = 1 \\ 0 & if \quad r = 0 \end{cases} \tag{5.2}$$

**Algorithm 2** Guided DropBlock

**Input:** output of $\ell^{th}$ layer is $A$, $block\_size$, $drop\_prob$, $mode$
**Output:** masked output of $A$
**if** $mode == Inference$ **then**
  return $A$
**else**
  mask $M_{i,j} : A > mask\_th$
  $list$= $find(M_{i,j} == 1)$
  $drop\_list = Random(list, drop\_prob)$ ; `// randomly select n items from list with drop probability p`
  **while** *i',j'*$\in drop\_list$ **do**
    k,l=$spatial\_square\_mask\_index(i', j', block\_size)$
    $M_{k,\ell} = 0$
  **end**
  $A = A \times M$
  $A = A \times count(M)/count\_ones(M)$
**end**

where, $\mathbf{W'}_j^{(\ell+1)} = r_j^{(\ell+1)} \odot \mathbf{W}_j^{(\ell+1)}$ and $\mathbf{W}_j$ is the column vector $w_i$, $\forall\ i$. In the case of convolution operation, the modified filters can behave as a conditional operation over $Bernoulli$ variable $r$ as described in equation 5.2.

Dropout is not commonly utilized following the convolutional layer. This is primarily due to the lack of semantic information at this stage, and the random elimination of pixel data may not effectively contribute to the variability in the sample. To overcome this limitation, Ghiasi *et al.* [73] introduced DropBlock, a novel approach that systematically eliminates semantic information from feature maps in a block-wise fashion. This can be advantageous when the blocks removed contain partial object information. However, the method may not be as effective if it removes blocks that predominantly contain background information. To address this issue, we propose leveraging the feature maps to create a mask highlighting crucial activated regions. This mask is then used to guide the block removal process, ensuring that the blocks dropped contain crucial semantic information. We refer to this approach as *"Guided DropBlock"*.

Algorithm 2 describes the proposed Guided DropBlock. Let $A = a_i^\ell$, $block\_size$ be the height and width of a spatial square mask with the center $M_{i,j}$. $M_{i,j}$ has been derived from the masking of $A$ by iteratively scheduling masking threshold ($mask\_th$) at every epoch. $M_{k,\ell}$ is set to be zero

based on randomly selected block$\{k, \ell\}$. The function $Spatial\_square\_mask\_index()$ takes a block as input and returns a list of all pixel locations of the block[2]. Finally, generated $M$ is applied to mask the feature map $A$. After that, $A$ is normalized by the total number of ones in $M$.

### 5.3.3 Filter augmentation (FA)

FA enhances the diversity of filters by applying a series of operations to the existing ones, which can be derived from predefined mathematical functions [197] or pre-trained CNN models [290]. This paper delves into various innovative operations to manipulate pre-trained CNN filters. FA operations can be integrated into each training epoch, transforming the resulting model into an ensemble of multiple classifiers influenced by different augmentation operations applied to the filters. This approach is particularly beneficial when adapting pre-trained models to a limited dataset that significantly differs from the source data. Previous work has introduced steerable CNN, which generate filters through rotational operations and initialize them at the outset. However, this method incurs a computational overhead, requiring $n$ times more learning parameters at each convolutional block for $n$ steerable functions and, consequently, $n$ times more resources to train the model. In contrast, this research proposes incorporating an augmentation module within the convolutional block of CNN models, empowering the network to learn invariant features tailored to the augmentation methods. Although the proposed method applies to any CNN model, this paper focuses on two specific ResNet architectures, 50 and 101. It is important to note that since this research emphasizes data-constrained environments, more parameterized ResNet architectures are not preferred. As shown in Figure 5-4, (a) represents the conventional training of the ResNet model, and (b) proposed framework for the ResNet model in which every convolutional block has a filter augmentation module ($< FA >$).

An infinite number of filters can be generated by changing the values of pre-existing filters. However, only some filters might be helpful for the target database. In this paper, six types of augmentation methods have been considered: 1) Guided DropBlock 5.3.2 2) learning strength of filters, 3) rotation operation, 4) flip operation, 5) channel shuffling, and 6) generating new filters while dropping connections. The five operations, 2 to 6, can be directly applied to the filters and

[2] $k$,$\ell$=$Spatial\_square\_mask\_index(i', j', block\_size)$, where, $i'$ and $j'$ as a center-pixel of the squared block of size $block\_size$. $k$ and $\ell$ have a list of all pixel locations of the block

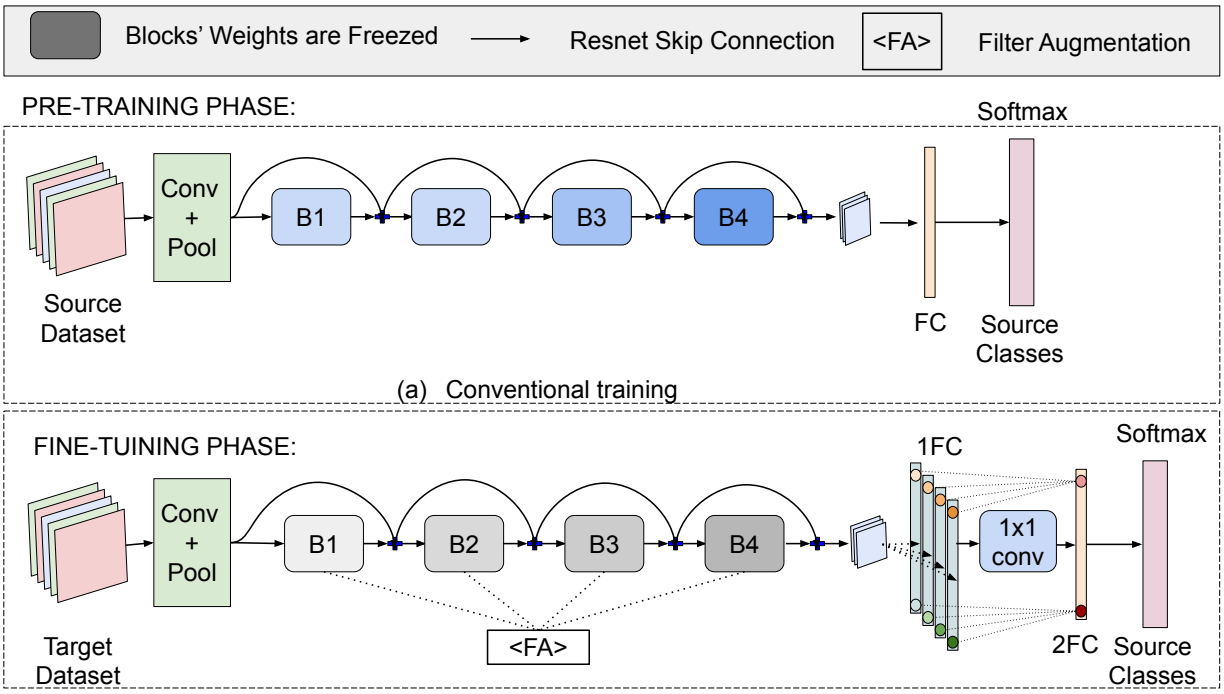



Figure 5-4: Illustration of ResNet model and proposed fine-tuning phase: (a) represents the conventional training of the ResNet model, (b) Proposed framework for fine-tuning of ResNet model in which every convolutional block has filter augmentation module ($< FA >$). While fine-tuning the model, convolutional blocks are frozen, and only the Fully Connected (FC) layer has been trained along with augmentation operations.

likely cover most ways of generating new filters from existing ones. For example, a $3 \times 3$ filter can only have eight new filters by applying rotation operation.

Similarly, horizontal and vertical axes have been chosen to use the filter's flip function without changing the filter structure. Also, the strength learning operation allows us to have a new filter while keeping the exact structure of the pre-trained filter [118]. Since filters are a 3D kernel, where the depth of the filter is many channels, a new 3D kernel can be obtained by applying channel shuffling; also, a new filter can be generated by changing any cell value of $3 \times 3$ filter.

**Rotation operation:** In the rotation operation (shown in Figure 5-3(a)), eight angles ($\theta \in [0°, 45°, ..., 315°]$) have been utilized to apply rotation on the convolutional filters. Out of eight possible angles, any of the angles is applied on the filters anticlockwise. The 2D affine matrix has

been computed to find the transformed index variable $i'$ and $j'$ and used to obtain the rotated filters.

$$M = \begin{bmatrix} \alpha & \beta & (1-\alpha) \times \omega_1 - \beta \times \omega_2 \\ -\beta & \alpha & \beta \times \omega_1 + (1-\alpha) \times \omega_2 \end{bmatrix} \tag{5.3}$$

$$\begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix} = M \times \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{5.4}$$

where, $\alpha = s \times cos(\theta)$, $\beta = s \times sin(\theta)$, and $s$ is the scaling parameter. $i$ and $j$ are index variables of the $w_{i \times j}$ parameter known as $height$ and $width$ for 2D filters, where, $i \in [0, ..., height]$ and $j \in [0, ..., width]$.

**Flip operation:** Two horizontal and vertical flips have been utilized (shown in Figure 5-3(b)). In case of horizontal flip, $w_{i \times j}$ can be written as $w_{i \times j'}$, where, $j' \in [width, ..., 0]$. Similarly, in the case of a vertical flip, the obtained filter is $w_{i' \times j}$, where $i' \in [height, ..., 0]$.

**Learning strength of filters:** In this type of operation (shown in Figure 5-3(e)), an augmented filter is generated by learning the strength of a filter and scaling the filter by element-wise multiplication and mathematically can be represented as $\mathbf{W}' = t \odot \mathbf{W}$ [118].

**Channel shuffling:** CNN filters are defined in four-dimensional tensor $w_{i \times j \times c \times n}$, where, $i$, $j$, $c$, and $n$ are height, width, number of channels, and number of filters, respectively. To apply the channel shuffling operation on the 4D tensor, channel index variable $c$ has been shuffled to obtain $c' = shuffle(c)$ (shown in Figure 5-3(d)).

**Randomly drop filter weight(s):** Unlike the dropout regularization, dropping weights are similar to dropconnect [262] which is directly applied on weights of CNN filters. Mathematically, it can be expressed as: $\widetilde{a_j}^{(\ell+1)} = f((r_{j \times i}^{(\ell+1)} \odot w_{j \times i}^{(\ell+1)}) a_i^{\ell})$, where, $\mathbf{r}$ is randomly sampled from $Bernoulli$ distribution and the dimension of $\mathbf{r}$ is same with weight parameter $\mathbf{W}$. In dropping filter weight (shown in Figure 5-3(c)), randomly sampled $Bernoulli$ variable $r_{i \times j}$ is applied on $w_{i \times j}$. It can be observed that applying drop-connect operation on the weight of filter $w$ generates new filters as a feature extractor, which leads the network to extract robust features from the sample.

### 5.3.4 Fully Connected Layer and loss function

As shown in Figure 5-4(b), each filter augmentation operation creates a new classifier, and the output feature vector $m$ represents numerical scores (pre-softmax logits). We are assigning $f : \mathbb{R}^d \mapsto \mathbb{R}^m$ to represent a basic classifier that receives a $d$-dimensional input. These $m$-dimensional feature vectors are combined by $1 \times 1$ $conv$ layer[3], mathematically it can be represented as $\bar{f}(\boldsymbol{x}) = \sum_{l=0}^{k} \Psi_l \times A^l(f(\boldsymbol{x}))$ where, $\bar{f}$ and $f$ are the output of ensemble of classifiers after applying $A^l$ augmentation and a selected classifier when no augmentation $A^0$ applied, respectively. The feature vector of the individual classifier then fuses with either averaging all logits [1FC] or using the second FC layers [2FC]. In all the experiments, both settings are used to compute the performance and compare with SOTA methods. $k$ is the number of augmented filters operations, $\Psi_l$ is the weight associated with each feature vector. In case of [1FC], $\Psi_l$ is a uniform distribution $\frac{1}{k}$. $\boldsymbol{y} = \bar{f}(\boldsymbol{x})$ will lead the model to smooth the decision boundary and essential while finetuning a model on the target dataset. The output of the FC layer is embeddings for the softmax layer. The loss function for the model can be written as $\mathcal{L}(\boldsymbol{y}, c) = -\varphi_c log \frac{exp(y_{n,c})}{\sum_{i=1}^{C} exp(y_{n,i})}$, where, $\boldsymbol{y}$ is the logits for the softmax layer, at $l = 0$ means un-augmented classifier. $c$ is the target, $\varphi$ is the weight, $C$ is the number of classes, and $N$ spans the mini-batch dimension.

### 5.3.5 Why Filter Augmentation Work on Data Constrained?

The fundamental issue deep learning encounters when dealing with small datasets is generalization. Deep models memorize the training samples rather than learning the underlying patterns when provided with a limited amount of data. As a result, the model exhibits high variance on the test set, demonstrating poor performance due to the insufficient smoothing of the decision boundary [99]. Let us assume pre-softmax logits $f^l(\boldsymbol{x}) =: \boldsymbol{y}^l \in \mathbb{R}^m$ as the sum of two random variables $\boldsymbol{y}_t^l = \boldsymbol{y}_d^l + \boldsymbol{y}_s^l$, where, $\boldsymbol{y}_s^l$ represents classifier's behavior on the source domain data, and $\boldsymbol{y}_d^l$ represents the effect of domain shift. The distribution of the source target component $\boldsymbol{y_s}$ over classifiers with mean $\boldsymbol{c} = \mathbb{E}_l \left[\boldsymbol{f}^l(\boldsymbol{x})\right]$, and covariance $\Sigma_c \in \mathbb{R}^{m \times m}$ characterizing randomness in the training process. We assume the distribution of the target data domain shift effect $y_d$ to be zero-mean (following from local linearization and zero mean perturbations) and to have covariance

[3]if feature vector size is $512 \times 1$ and many FA operations are eight then all feature vector has been concatenated, and resultant vector size is $512 \times 1 \times 8$, and $conv$ size would be $1 \times 1 \times 8 \times 1$

$\Sigma_d \in \mathbb{R}^{m \times m}$. Another way, it can be represented as $\Sigma_{ii} = \sigma_i^2$ and $\Sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$, for standard deviations $\sigma_i$ and correlations $\rho_{ij}$. In the case of the ensemble model, we parametrize the covariance between $\boldsymbol{y_s^i}$ and $\boldsymbol{y_s^j}$ for classifiers $i \neq j$ with $\zeta_s \sum_s$ and similarly between $\boldsymbol{y_d^i}$ and $\boldsymbol{y_d^j}$ with $\zeta_d \sum_d$ for $\zeta_s, \zeta_d \in [0, 1]$. With these correlation coefficients $\zeta_s$ and $\zeta_d$, this construction captures the range from no correlation $(\zeta = 0)$ to perfect correlation $(\zeta = 1)$.

**Variance Reduction** explained by [99] is also applicable in source and target finetuning, where multiple filter augmentation creates an ensemble model. Mathematically, for both variance component ratios $\sigma_s^2(k)/\sigma_s^2(1)$ and $\sigma_d^2(k)/\sigma_d^2(1)$ can be defined as[4]:

$$\frac{\sigma^2(k)}{\sigma^2(1)} = \frac{(1 + \zeta(k-1))(\sigma_1^2 + \sigma_i^2 - 2\rho_{1,i}\sigma_1\sigma_i)}{k(\sigma_1^2 + \sigma_i^2 - 2\rho_{1,i}\sigma_1\sigma_i)} = \frac{1 + \zeta(k-1)}{k} \tag{5.5}$$

where, $\zeta$ is correlation coefficients, $\sigma^2(1)$ covariance matrix for majority class. From equation 5.5, it can be observed that there is a direct relationship between the variance component ratio and the correlation coefficients within the ensemble network, culminating in a more resilient decision boundary when dealing with small datasets.

### 5.3.6 Implementation Details

Experiments are performed on a workstation with two 1080Ti GPUs under PyTorch [195] programming platform. Also, some of the experiments are performed on Amazon SageMaker p3.8xlarge, which has four V100-SXM2 GPUs [147]. The program is distributed on GPUs. Hyperparameters such as epochs, learning rate, and batch size are kept as 200, $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$ (after every $50$ epoch, the learning rate has been reduced by the factor 10), and 32, respectively for all the experiments. FA operations are fixed and lead to training corresponding feature vectors in the first FC layer. In guided dropblock, mask-threshold ($mask\_th$) has been scheduled zero till 50 epochs. After that, 150 equally spaced mask thresholds are drawn from the minimum to maximum value of the feature map. Obtained 150 mask-threshold values are then iteratively initialized from epochs 51 to 200. In the case of FA-ResNet [1FC], the network is trained independently, and softmax layer scores have been fused with equal weight. In FA-ResNet [2FC], feature vectors are com-

[4] subscript is omitted

bined with the learned weight of $1 \times 1$ $conv$ layer utilized before the final FC layer, as shown in Figure 5-4(b). There are two modes of making an inference[5]: 1) when we trained all the layers, then the best-trained model was utilized. 2) when we trained only the last layers, all the filters were the same except the final two FA layer's features vector of the model. In our proposed methods, the FA module is a set of operations defined in the convolutional block. The functions are applied directly to the convolutional filters. Rotation operation has been utilized from the library given by Riba *et al.* [206]. The other operations are available in the PyTorch library.

Empirically, we have obtained four filters of angle [$0°$, $45°$, $90°$, $135°$] in rotation operation and horizontal, vertical flip operations. Regarding channel shuffling, we have utilized one channel shift clockwise. Therefore, the total number of feature vectors is nine (four rotations, two flips, one strength learning, one channel shuffling, and one for all randomly dropping weights.)

## 5.4 Experiments and Results

The proposed FA approach has been evaluated with two source and target database settings: 1) intra-database and 2) cross-database. Table 5.3 presents the details of the experiments. For the intra-database experiments, ResNet50 and ResNet101 have been utilized. ResNet50, which is pre-trained on ImageNet, has been used for the data-constrained cross-database experiments. The pre-trained ResNet50 is then fine-tuned on the randomly sampled target dataset. The protocol is similar to few-shot learning; therefore, we evaluated the proposed model on publicly available few-shot databases and predefined protocols. We have used two architectures as the backbone, ResNet and ConvMixer, for evaluation.

### 5.4.1 Database and Protocols

The performance of the proposed algorithm is demonstrated on five classification benchmarking datasets: CIFAR10 [138], CIFAR100 [138], TinyImageNet [247], SVHN [180], MNIST [138], and for few-shot experiments CUB [261], and Omniglot [130] have been utilized. We have also conducted experiments on real-world applications such as newborn [15] and cataract pre-post surgery CMPD [114] identification. Some of the samples are shown in Figure 5-5. The details of the

[5]The augmentation operations have been applied when the model is getting trained and at the inference.

Table 5.1: Summarizing the protocols of the publicly available databases in terms of the number of classes in training and testing sets.

| Database | | Classes | Train | Test |
|---|---|---|---|---|
| **Source** | TinyImageNet | 200 | 100,000 | 10,000 |
| | ImageNet | 1,000 | 1,331,167 | 100,000 |
| **Target** | CIFAR10 | 10 | 50,000 | 10,000 |
| | CIFAR100 | 100 | 50,000 | 10,000 |
| | MNIST | 10 | 60,000 | 10,000 |
| | SVHN | 10 | 73257 | 26,032 |

Table 5.2: Summarizing the protocols of the publicly available databases for zero-shot experiments and real-world applications as a use-case in terms of the number of classes in training, validation, and testing sets.

| Dataset | Train | Validation | Test |
|---|---|---|---|
| Omniglot | 30 | - | 20 |
| CUB | 100 | 50 | 50 |
| CIFAR-FS | 64 | 16 | 20 |
| Cataract Mobile Periocular Database (CMPD) | 132 | - | 56 |
| Newborn Face | 10 | - | 86 |

Table 5.3: Summarizing the Protocol for inter/cross database experimental setup. The source database is used for training, and the target database is used for testing. (CIFAR10 and CIFAR100 mention as C10 and C100, respectively.)

| Experiments | Source Database | Target Database |
|---|---|---|
| **A: Intra Database Filter Augmentation** | C10 | C10 |
| | C100 | C100 |
| | TinyImageNet | TinyImageNet |
| | SVHN | SVHN |
| | MNIST | MNIST |
| **B: Cross Database Filter Augmentation** | ImageNet | C10 |
| | | C100 |
| | | MNIST |
| | | SVHN |
| | Tiny-ImageNet | C10 |
| | | C100 |
| | | MNIST |
| | | SVHN |

respective databases and experiments are mentioned in Tables 5.1, 5.2, and 5.3. ImageNet and Tiny-ImageNet are source databases in these experiments, and CIFAR-10, CIFAR-100, MNIST,

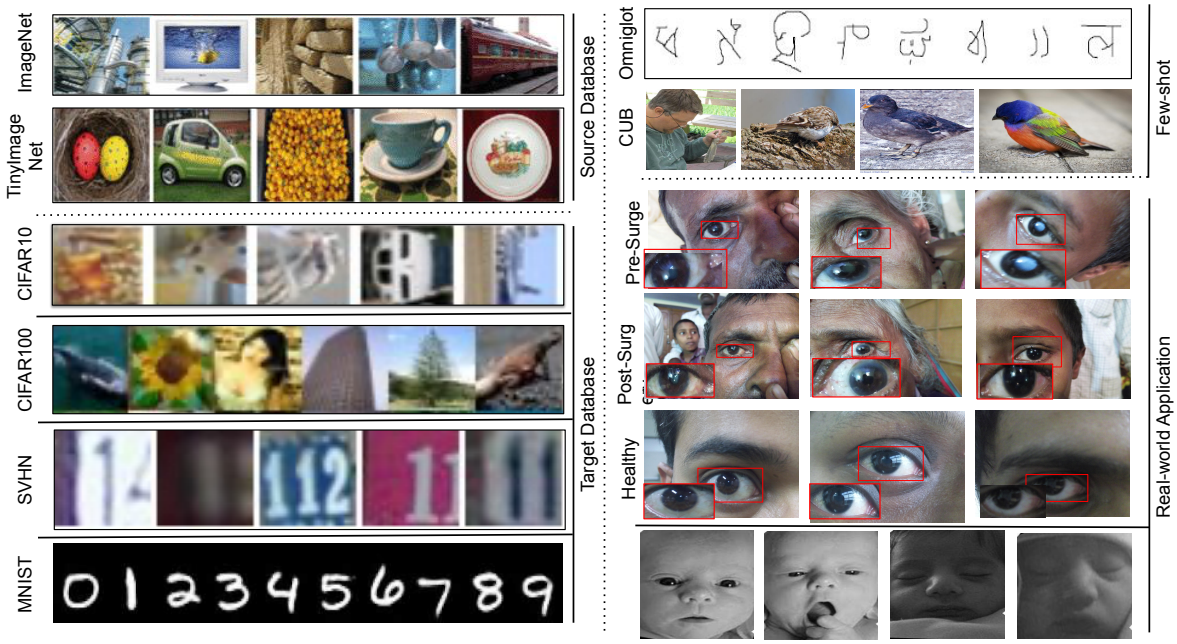



Figure 5-5: Shows a few samples from datasets used in our experiments. **Left** side of databases are bench-marking datasets, summarized in table 5.1. **Right** side of databases are used for data-constrained experiments such as few-shot, and real-world applications as a use case, summarized in table 5.2. We have also conducted experiments with the bench-marking dataset by reducing per class sample.

and SVHN are target databases. In the data-constrained cross-database experiments, the training set of the target databases is randomly selected with $1, 5, 10, 25, 50, 100$ samples per class. Five-fold cross-validation has been performed for data-constrained experiments, and the mean of test accuracies are shown in Figure 5-6. The testing set for the target databases is kept unchanged. Finetuning is performed on the model pre-trained on the source dataset, i.e., ImageNet.

We have also conducted few-shot experiments with 1-shot, 5-way, and 5-shot, 5-way settings on CUB, Omniglot, and CIFAR100 datasets. The protocols for all three datasets are predefined. Omniglot has 1623 handwritten characters of 50 different alphabets. The split of the dataset is adopted from Vinyalset *et al.* [259]. The background database has 30 alphabets, and the evaluation set has 20. In the case of CIFAR100, the new split defined as CIFAR-FS by Bertinetto *et al.* [13] is adopted. They divided the dataset into the train, validation, and test sets of $64$, $16$, and $20$. The resolution of the dataset is kept unchanged, which is $32 \times 32$. CUB has 11,788 images of resolution $84 \times 84$ and contains $200$ classes. The protocol is defined by Hu *et al.* [100]. The train set has 100 classes, 50 for validation and 50 novel classes for testing. All the experiments are performed with

Table 5.4: Intra-database results when the complete dataset has been used to train all layers of CNN models. Test accuracy (%) on five databases using two different depths of ResNet architectures. For the ConvMixer model, two architectures † h128-d4, ‡ h256-d20 have been utilized, which are not ResNet architecture and are highlighted with a light gray color. (The top two accuracies are in bold).

| Method | ResNet-50 | | | | | ResNet-101 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | C10 | C100 | SVHN | MNIST | Tiny ImageNet | C10 | C100 | SVHN | MNIST | Tiny ImageNet |
| ResNet [91] | 93.01 | 73.02 | 93.11 | 99.43 | 59.25 | 93.39 | 73.5 | 93.11 | 99.44 | 59.38 |
| SteerCNN [40] | 93.26 | 73.25 | 94.42 | 99.34 | 60.17 | 93.26 | 73.49 | 94.42 | 99.34 | 60.17 |
| SS-ResNet [118] | 92.89 | 72.38 | 94.11 | 99.7 | 60.05 | 92.89 | 72.14 | 94.11 | 99.7 | 60.05 |
| BiT [124] | **95.22** | **76.52** | **96.57** | **99.68** | **62.84** | **95.95** | **76.54** | **96.26** | **99.71** | **63.73** |
| ConvMixer [250] | 91.26† | 70.84† | 91.33† | 99.52† | 58.75† | **96.67**‡ | 76.29‡ | 95.03‡ | 99.59‡ | 62.42‡ |
| **FA-BiT** [1FC] | 95.08 | 76.15 | 96.25 | 99.51 | 62.55 | 95.41 | 76.06 | 95.88 | 99.7 | 62.55 |
| **FA-BiT** [2FC] | **96.18** | **77.08** | **97.25** | **99.73** | **63.69** | **96.63** | **77.17** | **97.38** | **99.73** | **64.84** |
| **FA-ConvMixer** [1FC] | 92.56† | 73.67† | 92.34† | 99.54† | 59.23† | - | - | - | - | - |
| **FA-ConvMixer** [2FC] | 93.78† | 74.03† | 93.89† | 99.65† | 60.51† | - | - | - | - | - |

five-fold cross-validation, and average accuracies are reported, which we explain in the following subsection.

### 5.4.2 Intra Database Results

The proposed filter augmentation-based finetuning has been evaluated on five databases, and experimental protocols are summarized in Tables 5.1 and 5.3. In Table 5.3, **row A:** Intra Database Filter Augmentation has the same source and target database. A training set of the source database is used for training, and a test set is used for evaluation. Two different ResNet of depth 50 and 101 are utilized to assess the performance of proposed *FA-ResNet*. The proposed method is then compared with baseline methods ResNet [91], steerableCNN [40] and SS-ResNet [118] and state-of-the-art (SOTA) methods ConvMixer [250] and BiT [124]. SteerableCNN and SS-ResNet methods are considered baseline because they have modified Reset's architecture like our proposed method. Specifically, steerableCNN and SS-ResNet are changing the convolutional layer.

For our proposed method, we have conducted experiments with different architectures and observed consistent improvement; here, we are reporting FA-BiT and FA-ConvMixer as a proposed modification of BiT and ConvMixer. Also, the proposed [1FC] and [2FC] FA operations are fixed and shared weights of the CNN model and then applied on all layers where one and two FC layers have been utilized, respectively. ConvMixer did not have ResNet as the base model; therefore, h128-d4 and h256-d20 have been utilized and in Table 5.4 represented as light gray color with symbol † and ‡, respectively. ConvMixer-based models need high computational resources, and the

computationally heavy model with our proposed modification is not feasible. However, we have computed results for our proposed FA-ConvMixer with base ConvMixer h128-d4 and observed improvement of 1% to 2%.

Experiments with the different base models with different depths of respective models are conducted to showcase the applicability of the proposed framework across robust and SOTA CNN architectures. It can be observed that FA-BiT-50 [2FC] is able to achieve $96.18\%$, $77.08\%$, $97.35\%$, $99.73\%$, $63.69\%$ on CIFAR10, CIFAR100, SVHN, MNIST, and TinyImageNet, respectively. And FA-BiT-101 [2FC] is able to achieve $96.63\%$, $77.17\%$, $97.38\%$, $99.73\%$, $64.84\%$ on CIFAR10, CIFAR100, SVHN, MNIST, and TinyImageNet, respectively. FA-BiT-50 [2FC] outperforms in comparison to SOTA methods reported in Table 5.4.

### 5.4.3 Cross Database Results

The proposed method is also evaluated on cross-database experiments, and experimental protocols are summarized in Tables 5.1, and 5.3. In Tables 5.3, **row B:** Cross Database Filter Augmentation has different source and target databases. A training set of the source database is used for training, and a test set is used for evaluation. In this setting, the model has been pre-trained on the source database and finetuned on the target database to adapt the model. Pre-trained ResNet models on ImageNet [47] with different depths are available and widely used to finetune the target dataset. Since pre-trained steerable CNN on ImageNet is unavailable, TinyImageNet has been used as a source database for pre-training the model and finetuned on the target databases. The protocol has been consistent for BiT and proposed FA-BiT as well. Two sets of experiments are performed to evaluate the proposed model on cross-database. 1) Use complete data, and 2) use data constrained settings for finetuning explained in section 5.4.4.

**Complete Data:** With complete training data, experiments can be divided into two subparts. 1) Finetune all the layers of the pre-trained model, and 2) Finetune the last layer of the pre-trained model. The results are reported in Tables 5.5 and 5.6. Pre-training with ImageNet as a source database is superior to TinyimageNet. From the Tables 5.5 and 5.6, it can be observed that compared to SOTA methods, the proposed method yields higher performance by 1% to 5% when only the last layer is finetuned.

Table 5.5: Cross-database results when a complete dataset has been used to finetune all the layers and the last layer of models. ImageNet has been used as a source database for test results with cross-database experiments. (The top two accuracies are in bold)

| | Method | ResNet-50 | | | | ResNet-101 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | C10 | C100 | SVHN | MNIST | C10 | C100 | SVHN | MNIST |
| All Layer | ResNet [91] | 84.73 | 55.77 | 95.39 | 99.55 | 85.66 | 56.18 | 96.32 | 99.64 |
| | SS-ResNet [118] | 84.71 | 55.64 | 95.73 | 99.65 | 85.70 | 57.12 | 96.54 | 99.68 |
| | BiT [124] | **97.20** | 86.50 | 97.90 | **99.73** | **98.50** | 90.80 | 98.24 | **99.83** |
| | **FA-BiT [1FC]** | 97.13 | **86.57** | **98.06** | 99.72 | 98.17 | **91.15** | **98.58** | 99.68 |
| | **FA-BiT [2FC]** | **98.13** | **87.56** | **98.55** | **99.74** | **98.64** | **91.26** | **98.62** | **99.79** |
| Last Layer | ResNet [91] | 78.78 | 49.30 | 44.68 | 92.64 | 67.13 | 47.84 | 39.92 | 90.68 |
| | SS-ResNet [118] | 80.05 | 55.02 | 58.19 | 94.74 | 68.71 | 48.87 | 45.92 | 92.22 |
| | BiT [124] | 82.3 | 63 | 68.81 | 94.04 | 85.2 | 67.4 | 72.15 | 98.13 |
| | **FA-BiT [1FC]** | **85.08** | **65.54** | **69.59** | **95.35** | **86.36** | **68.32** | **74.82** | **98.45** |
| | **FA-BiT [1FC]** | **86.45** | **65.81** | **71.38** | **96.38** | **87.66** | **70.27** | **75.04** | **99.12** |

Table 5.6: Cross-database results when a complete dataset has been used to finetune all the layers and the last layer of models. TinyImageNet has been used as a source database for test results with cross-database experiments. (The top two accuracies are in bold)

| | Method | ResNet-50 | | | | ResNet-101 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | C10 | C100 | SVHN | MNIST | C10 | C100 | SVHN | MNIST |
| All Layer | ResNet [91] | 80.99 | 50.99 | 90.93 | 98.05 | 81.35 | 52.77 | 92.53 | 98.26 |
| | Steerable ResNet [40] | 80.5 | 51.03 | 93.34 | 98.82 | 80.86 | 54.22 | 93.29 | 98.98 |
| | SS-ResNet [118] | 80.68 | 51.45 | 93.83 | 99.02 | 81.32 | 54.55 | 93.48 | 99.09 |
| | BiT [124] | 93.3 | **83.69** | 94.57 | 99.03 | **94.60** | 87.36 | 94.27 | 99.14 |
| | **FA-BiT [1FC]** | **93.47** | 83.11 | **95.13** | **99.14** | 94.56 | **87.88** | **95.85** | **99.28** |
| | **FA-BiT [2FC]** | **94.66** | **84.51** | **95.72** | **99.25** | **94.69** | **88.89** | **96.02** | **99.47** |
| Last Layer | ResNet [91] | 74.8 | 45.12 | 42.17 | 91.19 | 63.48 | 44.11 | 37.29 | 91.38 |
| | Steerable ResNet [40] | 77.19 | 52.76 | 55.47 | 94.15 | 66.23 | 45.71 | 43.22 | 94.42 |
| | SS-ResNet [118] | 78.19 | 55.41 | 63.97 | 93.49 | 77.92 | 58.45 | 67.21 | 93.64 |
| | BiT [124] | 80.85 | 59.46 | 65.97 | 94.46 | 81.84 | 63.26 | 69.08 | 94.71 |
| | **FA-BiT [1FC]** | **81.74** | **61.16** | **67.45** | **95.55** | **83.68** | **65.04** | **71.37** | **95.71** |
| | **FA-BiT [2FC]** | **82.93** | **62.69** | **75.99** | **95.68** | **84** | **67.23** | **74.78** | **95.81** |

An improvement ranging from 0.1% to 1% is observed when finetuning all layers on target datasets. These results highlight that the availability of comprehensive data and ample training samples enables the effective training of all model layers. When sufficient data is present, the models' absolute performance is superior when training all layers compared to just training the last layer. Nonetheless, it is essential to emphasize that *the proportional enhancement compared to the SOTA is more pronounced when only the last layer of our proposed FA-BiT model is finetuned*.

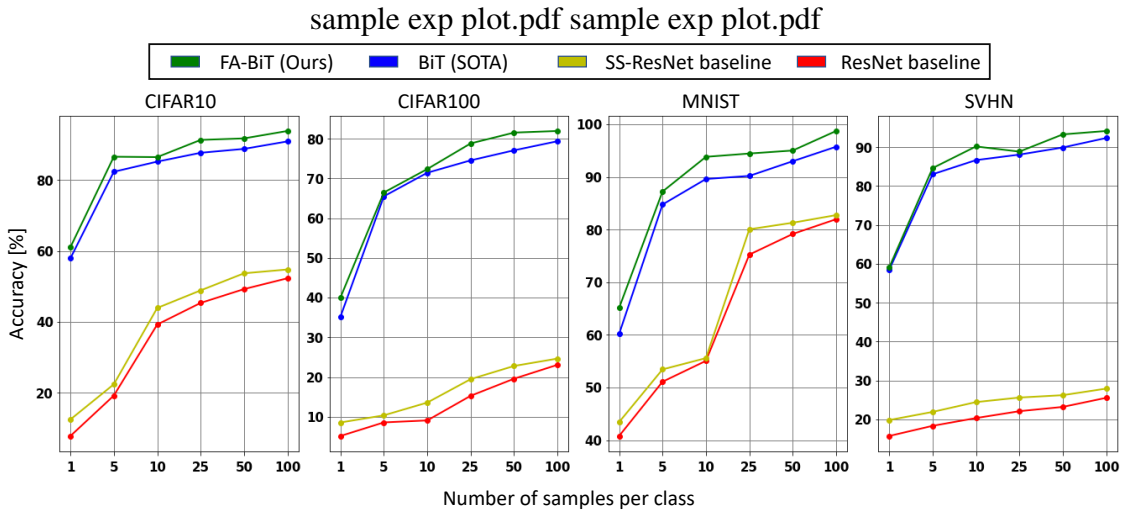



Figure 5-6: Summarizing mean of test accuracy over five random folds (represented on the Y-axis) vs. respective dataset sampling size. On the X-axis, the number represents pre-class sample(s) selected randomly for finetuning. We have employed two baselines, ResNet and SS-ResNet, and BiT SOTA methods to compare with our proposed FA-BiT [2FC] approach. All methods base model ResNet50 is kept the same, and the ImageNet database has been utilized for pre-training. (best viewed in color)

### 5.4.4 Data Constrained Problem

In data-constrained problems, there are either too few samples or classes available, making it challenging to capture the entire distribution of the data accurately:

1. We executed experiments on benchmark databases, decreasing the number of samples across all categories, and assessed the class activation maps of the fine-tuned models.

2. We benchmarked our proposed method against state-of-the-art techniques in few-shot learning scenarios.

3. We demonstrated the efficacy of our approach through real-world applications, specifically in identifying individuals in newborns and in pre-post cataract surgery scenarios.

#### Reduced Samples in Bench-marking Databases

In this experiment, all of the results were obtained by fine-tuning only the last layer of the pre-trained model while keeping the rest of the layers fixed. We noticed a decrease in accuracy when attempting to train all the layers of the model with a limited number of samples. Consequently, for

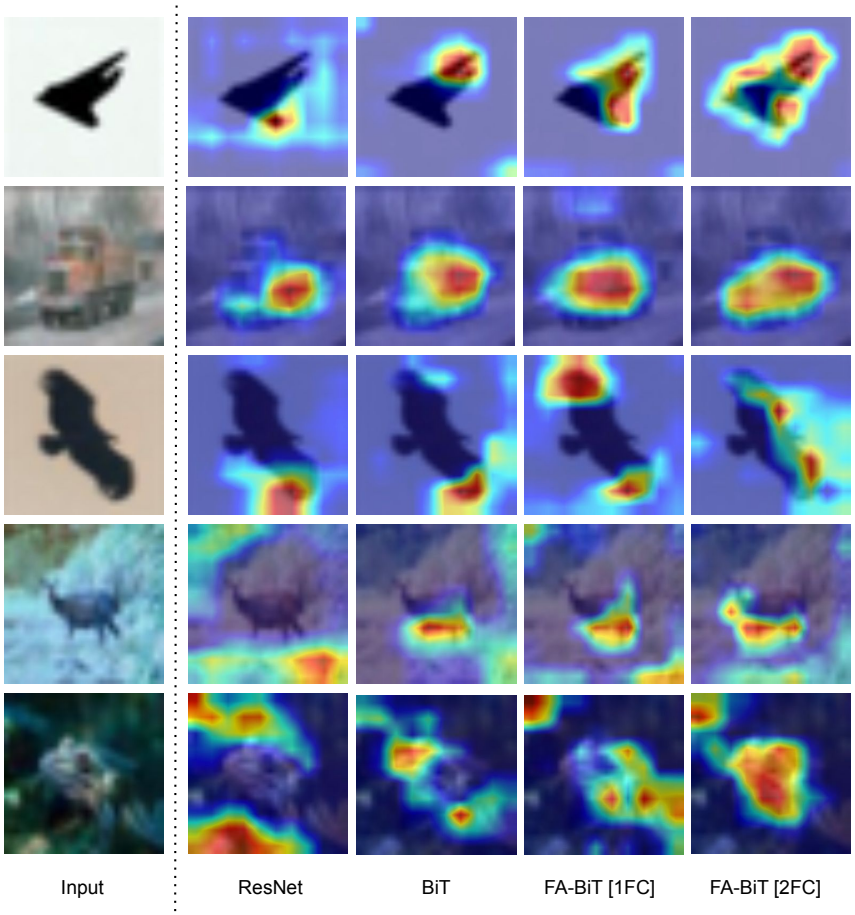



Figure 5-7: Illustration of Class Activation Map (CAM) from the model trained on ImageNet and finetuned all the layers using the CIFAR10 dataset. The first column is a CIFAR10 test image that feeds into the model. The second to fifth columns are the CAM output of ResNet, BiT, FA-BiT [1FC], and FA-BiT [2FC]. From the fifth column, it can be observed that there are more active regions around the object for FA-BiT [2FC].

protocols with data constraints, we have chosen not to include comparisons with experiments that trained all model layers. The results that were obtained were benchmarked against our proposed method, FA-BiT [2FC]. All evaluations were carried out using ImageNet as the source database and CIFAR10, CIFAR100, MNIST, and SVHN as target databases. As illustrated in Figure 5-6, it is evident that our FA-BiT [2FC] method not only surpasses baseline methods such as ResNet and SS-ResNet but also outperforms the state-of-the-art BiT method. In scenarios with only one sample per class, our method achieved test accuracies of $61.03\%$, $40.02\%$, $65.12\%$, and $59.03\%$ on CIFAR10, CIFAR100, SVHN, and MNIST, respectively. These results show improvements of $3.13\%$, $4.82\%$, $4.95\%$, and $0.62\%$ over the state-of-the-art BiT method on the respective target

Table 5.7: Test accuracy (%) on the Omniglot dataset. In the BiT and proposed method, backbone ResNet50 has been utilized for the evaluation. (The top two results are in bold, ‡ results are not reported in the paper, computed using the official code)

| Method | 1-shot, 5-way | 5-shot, 5-way |
|---|---|---|
| [219] | 82.8 | 94.9 |
| [122] | 97.3 | 98.4 |
| [259] | 98.1 | 98.9 |
| SS-ResNet34 [118] | 97.6 ± 0.84 | 98.3 ± 1.03 |
| ‡PT-MAP [101] | 99.1±0.32 | 99.6±0.15 |
| [199] | **99.7** | **99.9** |
| BiT [124] | 98.7 ± 1.29 | 99.0 ± 1.27 |
| **FA-BiT [1FC]** | 99.4 ± 0.57 | 99.5 ± 0.22 |
| **FA-BiT [2FC]** | **99.8 ± 0.15** | **99.9 ± 0.18** |

databases.

**Visualization:** Figure 5-7 illustrates the class activation map of different methods used in Table 5.6. The first row is an airplane class; it can be observed that the conventional ResNet focuses on a smaller portion of the airplane than BiT. However, our proposed methods, FA-BiT [1FC] and FA-BiT [2FC], almost concentrate on the whole object. The pattern can be seen consistently in other classes as well. The activation map supports our assertion that augmented filters allow the model to learn more convenient features.

### Few-shot learning

Table 5.7 summarizes the results of the proposed algorithm on the Omniglot database. It can be observed that utilizing BiT of depth 101 with Filter Augmentation (FA-BiT [2FC]) outperforms methods for 1-shot, 5-way protocol. In the 5-shot, 5-way protocol, the mean accuracy of the proposed method is the same, with a standard deviation of $0.18$ compared to SOTA. We have also evaluated and compared the proposed model on CUB [261] and CIFAR-FS [13] datasets. Table 5.8 summarizes the results of the proposed algorithm and compares them with existing SOTA algorithms. In the case of CIFAR-FS datasets, it can be observed that the proposed FA-BiT [2FC] is the second-best performing algorithm for the 5-shot, 5-way protocol. However, In the case of the CUB dataset, the proposed method performs better than the SOTA method when experimenting with the 1-shot, 5-way protocol. In 5-shot, 5-way for CUB, and 1-shot, 5-way for CIFAR-FS,

Table 5.8: Test accuracy (%) on CUB [261], and CIFAR-FS [13] datasets with few shot setup. In the BiT and proposed method, backbone ResNet50 has been utilized for the evaluation. (The top two results are in bold)

| Method | CUB | | CIFAR-FS | |
|---|---|---|---|---|
| | 1-shot, 5-way | 5-shot, 5-way | 1-shot, 5-way | 5-shot, 5-way |
| MetaVRF [306] | - | - | 63.1±0.70 | 76.5±0.90 |
| S2M2R [168] | 80.7±0.81 | 90.9±0.44 | 74.8±0.19 | 67.5±0.13 |
| PT-MAP [101] | **91.6±0.19** | **94.0±0.10** | **87.7±0.23** | **90.7±0.15** |
| Steerable CNN [40] | 81.2±0.50 | 91.4±0.60 | 80.4±0.25 | 83.0±0.44 |
| SS-ResNet34 [118] | 84.9±0.84 | 92.8±0.71 | 81.9±0.58 | 83.1±0.81 |
| DNS-MR [229] | - | - | 78.0±0.90 | 87.3±0.60 |
| TDE-FSL [284] | - | - | 84.68±1.14 | 89.24±0.66 |
| BiT [124] | 88.6±0.83 | 92.4±1.79 | 83.2±0.98 | 88.9±1.45 |
| **FA-BiT [1FC]** | 89.6±0.05 | 93.3±0.02 | 84.8±0.08 | 89.6±0.03 |
| **FA-BiT [2FC]** | **91.8±0.08** | **94.0±0.06** | **87.7±0.09** | **90.2±0.04** |

the proposed FA-BiT [2FC] performs similarly with reduced variance. The reduced variance is obtained because of the nature of the ensemble explained in section 5.3.5.

### Real-world Application

To validate the effectiveness of our proposed method, we have applied it to two specific applications: (i) Identification Before and After Cataract Surgery Using Periocular Information: The periocular region, encompassing the eye and its surrounding area, undergoes noticeable changes post-cataract surgery, potentially impairing the performance of identification systems [184]. Our method addresses these challenges and improves the system's resilience to such changes. (ii) Newborn Face Identification: This application is particularly challenging due to the scarce availability of extensive datasets, posing difficulties in developing robust identification models [15]. Both of these applications inherently deal with data-constrained environments, making them apt for demonstrating the strengths of our proposed method. For experimental validation, we adhered strictly to predefined protocols across all tests. Performance was evaluated using rank-1 identification accuracy (in percentage) on both the newborn and CMPD datasets, with the results detailed in Tables 5.9 and 5.10. The results indicate that our FA-BIT [2FC] model performs better than the SOTA methods across various gallery settings in the newborn dataset. Additionally, our method

Table 5.9: Identification accuracy (%) on newborn. Gallery in newborn experiments is a set of representative image(s) of each class from the test set. If one representative image is kept, the gallery number is represented as one, and experiments are repeated from one to four galleries. † represents that the model is not finetuned on newborn and CMPD dataset. FaceViT model trained on MS-Celeb-1M dataset. (The top two results are in bold)

| **Method** | **Newborn** | | | |
|---|---|---|---|---|
| | **Number of Gallery Images** | | | |
| | **1** | **2** | **3** | **4** |
| Gabor+scatNet+DSIFT | - | - | - | - |
| SS-VGG-face | 70.4±0.5 | 81.4±1.59 | 86.5±1.20 | 90.0±1.53 |
| Steerable CNN | 73.4±2.34 | 82.8±2.05 | 85.8±2.45 | 91.4±2.51 |
| PT-MAP | 77.2±1.59 | 84.1±1.11 | **86.6±1.56** | 92.8±1.65 |
| DNS-MR | 74.7±1.88 | 83.5±1.23 | 85.5±1.62 | 90.3±1.72 |
| Inception-Resnet-v1 | 74.5±1.15 | 82.8±1.05 | 85.4±0.95 | 91.5±0.84 |
| BiT | 76.2±1.36 | 83.6±1.02 | 85.9±0.99 | 92.8±1.53 |
| **FA-BiT [1FC]** | **79.3±0.11** | **85.2±0.10** | 86.1±0.12 | **93.1±0.63** |
| **FA-BiT [2FC]** | **80.4±0.07** | **86.7±0.09** | **87.8±0.18** | **93.8±0.34** |
| FaceViT | 27.1±0.02† | 37.2±0.02† | 44.4±0.01† | 49.9±0.02† |

demonstrates enhanced performance in both modalities (L and R) and registered and unregistered periocular scenarios for the periocular dataset. It is important to note that four eye points are aligned across all images in the registered periocular scenario.

## 5.5 Ablation Study

Through our empirical observations, we have found that utilizing four filters at angles [$0°$, $45°$, $90°$, $135°$] for the rotation operation yields performance almost equivalent to using eight filters. This is because the combination of four filters and horizontal and vertical flip operations can generate filters similar to those produced with eight filters. Given our focus on data-constrained learning, the ablation results are based on a random selection of $100$ samples. Moreover, we have individually assessed all six FA operations (refer to serial numbers 3 to 8 in Table 5.11), following a procedure of five-time random subsampling. The results, including means and standard deviations, are provided in Table 5.11. Among all the FA operations applied, Guided DropBlock stands out as the most effective, and the results from FA-BiT [2FC] indicate that a combination of these FA operations leads to enhanced performance. Contrary to approaches that add filters to models, which increases training overhead, our recommendation is to implement augmentation operations directly on the filters and finetune only the CNN models' last layer. This approach maintains

Table 5.10: Identification accuracy (%) on CMPD datasets. L and R represent left-left and right-right periocular matching from sessions S1 and S2. In the BiT and proposed method, backbone ResNet50 has been utilized for the evaluation. † represents that the model is not finetuned on newborn and CMPD dataset. FaceViT model trained on MS-Celeb-1M dataset. (The top two results are in bold)

| **Method** | **CMPD S1-S2 (pre-post surgery)** | | | |
|---|---|---|---|---|
| | **Unregistered** | | **Registered** | |
| | **L** | **R** | **L** | **R** |
| Gabor+scatNet+DSIFT | 16.3 | 15.5 | 30.1 | 22.4 |
| SS-VGG-face | 52.4 | 60.1 | 68.7 | 66.6 |
| Steerable CNN | 51.8 | 50.3 | 64.2 | 61.8 |
| PT-MAP | 56.7 | 54.5 | 69.8 | 67.3 |
| DNS-MR | 54.9 | 52.2 | 66.7 | 62.7 |
| Inception-Resnet-v1 | 55.6 | 53.9 | 67.3 | 65.7 |
| BiT | 56.2 | 54.1 | 68.7 | 67.2 |
| **FA-BiT [1FC]** | **61.2** | **60.8** | **71.9** | **69.8** |
| **FA-BiT [2FC]** | **63.9** | **62.4** | **74.2** | **71.2** |
| FaceViT | 43.2† | 41.6† | 54.6† | 51.3† |

efficiency while improving the model's performance in data-constrained scenarios.

## 5.6 Summary

Finetuning is a prevalent technique in deep learning, mainly when dealing with data-constrained challenges. The approach involves adjusting the extent to which the learning parameters of a model are either kept static or adapted based on the size of the available training dataset. In this paper, we introduce Guided Dropblock and Filter Augmentation to enhance image feature extraction. Guided Dropblock is motivated from Chapter 3 and [73], and FA is achieved by augmenting the filters within convolutional layers while maintaining the current number of learnable parameters. The proposed methodology enables the augmentation of specific filters that are pertinent to the target dataset, employing augmentation operations on filters that have been pre-trained. Through extensive experimentation, we have validated that applying augmentation operations to convolutional filters, combined with the regularization effect of Guided Dropblock, leads to improved performance. This is particularly evident in cross-dataset experiments and scenarios characterized by limited data availability.

Table 5.11: Test accuracy (%) on CIFAR10 and SVHN datasets as **Ablation study**. ResNet-50 has been used as the backbone model for BiT, pre-trained on the ImageNet, and fine-tuned on the target dataset. BiT+ is the model that has been trained while augmenting the training data while utilizing 1) rotation, 2) flip, 3) channel shuffling, 4) generating new samples while randomly dropping image pixels, 5) blocks, and 6) change intensity. The number of augmentation operations is six.

| **S.No.** | **FA operation** | **CIFAR10** | **SVHN** |
|---|---|---|---|
| 1 | BiT | 85.16±1.6 | 86.67±0.79 |
| 2 | BiT+ | 85.36±1.03 | 86.8±0.58 |
| 3 | Learning Strength | 85.58±1.30 | 87.50±1.51 |
| 4 | Rotation | 85.86±1.91 | 87.43±1.12 |
| 5 | Flip | 85.44±0.92 | 86.04±0.69 |
| 6 | Channel Shuffling | 85.20±0.31 | 86.12±0.56 |
| 7 | Dropping Weights | 85.93±0.54 | 86.81±0.43 |
| 8 | Guided DropBlock | 86.07±0.16 | 89.74±0.25 |
| 9 | FA-BiT [1FC] | 86.27±0.12 | 89.19±0.34 |
| 10 | FA-BiT [2FC] | 86.43±0.41 | 90.19±0.28 |

# Chapter 6

# Conclusions and Future Work

This dissertation explores the application of deep learning models in constrained learning settings, especially when data availability is limited. As shown in Figure 6-1, Constrained Learning (CL) encompasses two primary categories: (A) Data Constrained and (B) Resource Constrained. This research explicitly addresses the data-constrained scenario, providing innovative strategies across the deep learning model's input, model, and feature spaces. These novel methods have showcased enhanced results across various standard benchmark datasets, surpassing the performance of existing state-of-the-art techniques.

- **CNN training on Data Constrained Settings:** We propose Structure and Strength-Filtered CNN as a framework for learning a CNN model with small training databases. Also, empirically, using unsupervised dictionary filters to initialize CNN's filters can help the model

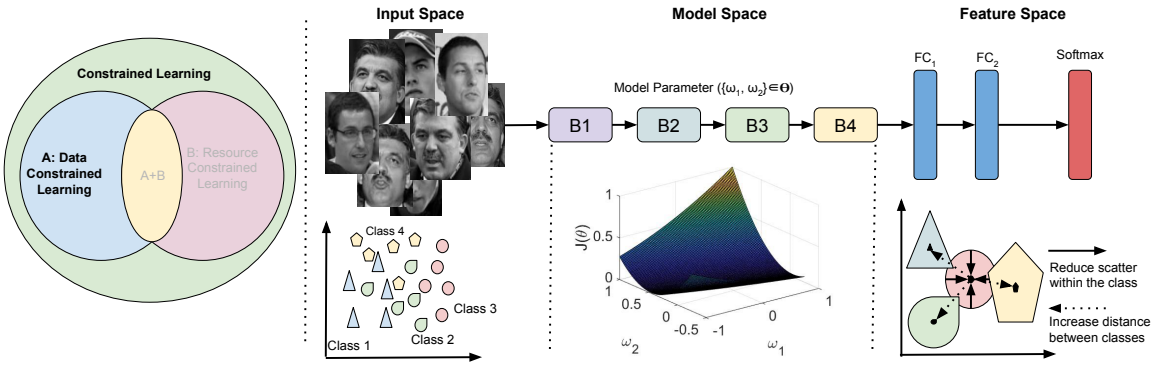



Figure 6-1: Illustration of constrained learning and proposed solutions for deep architecture pipeline's input, model, and feature space.

training, even if data is limited. We next suggest learning the strength of the filters with the given training data. Using different architectures and experiments, we demonstrate the efficacy of the proposed approach. Specifically, in the case of newborn face recognition, remarkable improvement in accuracy is achieved with the proposed method. The effectiveness of the proposed model has been demonstrated on multiple object classification databases and a real-world newborn face recognition problem.

- **Guided Dropout Regularization:** We introduce a novel approach, a guidance-based variant of dropout, designed to deactivate highly activated nodes in every iteration. This encourages nodes that are either inactive or possess low strength to adapt and learn distinctive features. Throughout the training phase, these low-strength nodes play a more significant role in the learning process, aiming to minimize loss and ultimately enhance their strength. We have assessed the performance of this guided dropout using both dense neural networks and convolutional neural networks. For all conducted experiments, benchmark databases were employed, and the outcomes have demonstrated the efficacy of the proposed guided dropout method.

- **Over-Complete Distribution for Generalized Zero-shot Learning:** We introduced the concept of over-complete distribution, applying it to train discriminative classifiers in Zero-Shot Learning (ZSL) and Generalized Zero-Shot Learning (GZSL) contexts. An over-complete distribution is crafted by generating all potentially challenging samples for a category that lies in close proximity to other competing categories. Through experimental validation, we have demonstrated that a classifier trained on a separable distribution may fail to generalize effectively on test samples drawn from distributions where classes overlap. Implementing the over-complete distribution has enhanced class separability and boosted classification accuracy. The efficacy of the proposed method is evident from experiments conducted on three benchmark databases, where it consistently outperformed existing approaches. Additionally, the Over-Complete Distribution (OCD) principle, alongside optimizing inter-class and intra-class distances, holds potential for application in other computational frameworks, such as Generative Adversarial Networks (GANs).

- **Guided Dropblock and Filter Augmentation:** We have introduced Guided Dropblock and

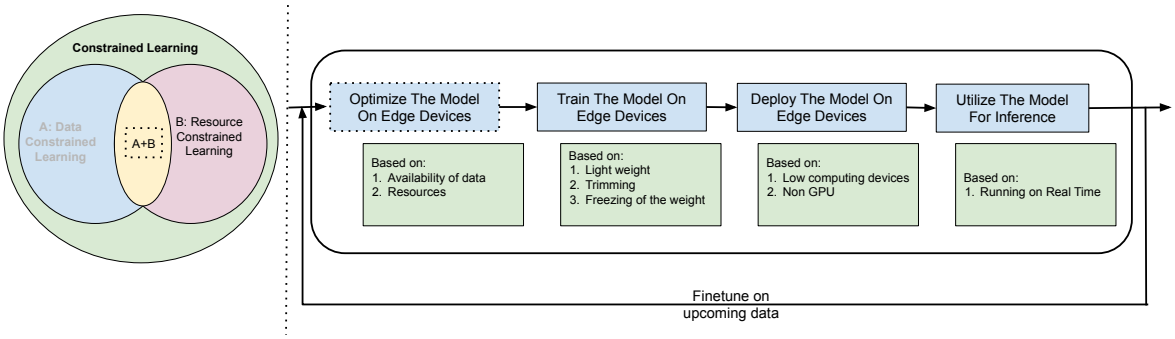



Figure 6-2: Illustration of future work when resource, data, and both constraints are applicable. The dotted block represents the optional module based on data and resource applicability.

Filter Augmentation, methodologies that enhance the robustness of features extracted from images. This is achieved by augmenting the filters in convolutional layers without an accompanying increase in the number of parameters that require learning. Our approach enables the selective augmentation of filters pertinent to the target data by applying augmentation operations to pre-trained filters. Through extensive experimentation, we have validated that combining filter augmentation and Guided Dropblock regularization leads to improved performance, particularly noticeable in cross-dataset experiments and few-shot learning scenarios.

Although the research presented in this dissertation makes significant strides in deep learning under data-constrained conditions, both from theoretical and technological perspectives, ample opportunities for further progress remain. Leveraging the groundwork laid by this dissertation, Figure 6-2 offers a visual representation of potential avenues for future research. Below, we delineate several specific directions for prospective studies:

- **Deep Learning in Resource-Constrained Settings:** In addition to limitations imposed by data scarcity, resource constraints serve as another impediment to advancing deep learning. Efficient training and inference demand the availability of a parallel processing unit equipped with a substantial internal memory. Lin et al. [149] introduced Quantization-Aware Scaling alongside Sparse Update to optimize their bespoke microcontroller, demonstrating that it operates efficiently with 256KB SRAM and 1MB Flash, utilizing less than 1/1000 of the memory without the need for auxiliary storage. Despite this progress, there remains a critical need for ongoing research to ascertain the universal adaptability of deep learning models

across diverse hardware configurations and resource allocations, ensuring their widespread applicability.

- **Training and Inference of the Deep Models on the Edge Devices:** deep learning models are typically trained in cloud environments before being implemented on edge devices. This practice restricts the model's capacity for fine-tuning post-deployment and hinders the application of online joint knowledge distillation. Gandelsman et al. [70] has proposed a method for unsupervised test-time fine-tuning after deployment. Addressing the challenges posed by resource limitations in deep learning will facilitate the adoption of test-time fine-tuning, resulting in enhanced model performance and a more transparent understanding of deep learning operations beyond the "Black Box" paradigm.

- **Out-of Distribution Generalization:** In data-constrained settings, capturing the entirety of a distribution is challenging, leading to issues with Out-of-Distribution (OOD) Generalization when training deep models. This dissertation contributes to mitigating the impacts of OOD problems by introducing novel architectures and regularization techniques. Nevertheless, this domain demands further exploration to effectively utilize the sparse available samples, aiming to overcome data limitations while training deep models.

In conclusion, it is crucial to emphasize that the research conducted in this dissertation and the future work proposed centers on employing deep learning algorithms in scenarios characterized by limitations in resources, data, or both. Under such constraints, traditional deep models tend to memorize and overfit the training data. The contributions made through this dissertation are pivotal in broadening the spectrum of complex machine learning challenges where deep models can be effectively applied while still achieving outstanding performance.

# Bibliography


[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2), 2015. 68

[2] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 819–826, 2013. 59, 66, 70, 71

[3] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936, 2015. 70, 71

[4] Michael A. Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 78

[5] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3(4):283–293, 2017. 19

[6] Brandon Amos. Openface 0.2.0: Higher accuracy and halved execution time. https://swami1995.github.io/2016/01/19/openface-0.2.0/, last accessed: 01/2024. 66

[7] Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. In *International Society for Music Information Retrieval (ISMIR)*, pages 657–662, 2011. 25, 26, 34, 36, 38

[8] Yuval Atzmon and Gal Chechik. Adaptive confidence smoothing for generalized zero-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11671–11680, 2019. 13, 60, 71, 72

[9] Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 3084–3092. Curran Associates, Inc., 2013. 41, 49, 81

[10] Mehrtash Harandi Baktashmotlagh, Mahsa and Mathieu Salzmann. Distribution-matching embedding for visual domain adaptation. *Journal of Machine Learning Research (JMLR)*, 17(1):3760–3789, 2016. 11

[11] Samy Bengio. Sharing representations for long tail computer vision problems. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 1–1. ACM, 2015. 5

[12] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ACM International conference on machine learning (ICML)*, pages 41–48. ACM, 2009. 12

[13] L Bertinetto, J Henriques, P Torr, and A Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019. xx, 92, 98, 99

[14] Samarth Bharadwaj, Himanshu S Bhatt, Richa Singh, Mayank Vatsa, and Sanjay K Singh. Face recognition for newborns: A preliminary study. In *IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6, 2010. 23

[15] Samarth Bharadwaj, Himanshu S Bhatt, Mayank Vatsa, and Richa Singh. Domain specific

learning for newborn face recognition. *IEEE Transactions on Information Forensics and Security (TIFS)*, 11(7):1630–1641, 2016. xix, 15, 23, 26, 32, 37, 38, 55, 90, 99

[16] Aparna Bharati, Richa Singh, Mayank Vatsa, and Kevin W Bowyer. Detecting facial retouching using supervised deep learning. *IEEE Transactions on Information Forensics and Security (TIFS)*, 11(9):1903–1913, 2016. 17

[17] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006. 13

[18] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3722–3731, 2017. 9, 12, 13

[19] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research (JMLR)*, 2(Mar):499–526, 2002. 54

[20] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Generating visual representations for zero-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2666–2673, 2017. 70

[21] Shaofeng Cai, Yao Shu, Gang Chen, Beng Chin Ooi, Wei Wang, and Meihui Zhang. Effective and efficient dropout for deep convolutional neural networks. *arXiv preprint arXiv:1904.03392*, 2019. 42

[22] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *AGR*, pages 67–74, 2018. 14

[23] Xiaohuan Cao, Jianhua Yang, Yaozong Gao, Yanrong Guo, Guorong Wu, and Dinggang Shen. Dual-core steered non-rigid registration for multi-modal images via bi-directional image synthesis. *Medical image analysis (MIA)*, 41:18–31, 2017. 13

[24] Dominique Cardon, Jean-Philippe Cointet, and Antoine Mazieres. Neurons spike back: The invention of inductive machines and the artificial intelligence controversy. *Reseaux*, 36:173–220, 2018. xiii, 2

[25] Rich Caruana. Multitask learning. *Machine learning (ML)*, 28(1):41–75, 1997. 57

[26] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 24

[27] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing (TIP)*, 24(12):5017–5032, 2015. 25, 26, 34, 36

[28] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5327–5336, 2016. 70, 71

[29] Agisilaos Chartsias, Thomas Joyce, Mario Valerio Giuffrida, and Sotirios A Tsaftaris. Multimodal mr synthesis via modality-invariant latent representation. *IEEE Transactions on Medical Imaging (TMI)*, 37(3):803–814, 2017. 13

[30] Liyan Chen, Philip Gautier, and Sergul Aydore. Dropcluster: A structured dropout for convolutional networks. *arXiv preprint arXiv:2002.02997*, 2020. 42

[31] Shiming Chen, Ziming Hong, Guo-Sen Xie, Wenhan Wang, Qinmu Peng, Kai Wang, Jian Zhao, and Xinge You. Msdn: Mutually semantic distillation network for zero-shot learning. *arXiv preprint arXiv:2203.03137*, 2022. 71

[32] Shiming Chen, Guosen Xie, Yang Liu, Qinmu Peng, Baigui Sun, Hao Li, Xinge You, and Ling Shao. Hsva: Hierarchical semantic-visual adaptation for zero-shot learning. *Advances in Neural Information Processing Systems (NIPS)*, 34, 2021. 71

[33] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person Re-identification. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 9, 18

[34] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2172–2180, 2016. 12

[35] Saheb Chhabra, Puspita Majumdar, Mayank Vatsa, and Richa Singh. Data fine-tuning. In *Proceedings of conference the Association for the Advancement of Artificial Intelligence (AAAI)*, volume 33, pages 8223–8230, 2019. 9, 14

[36] LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4088–4098, 2017. 12

[37] Brian Chu, Vashisht Madhavan, Oscar Beijbom, Judy Hoffman, and Trevor Darrell. Best practices for fine-tuning visual classifiers to new domains. In *European conference on computer vision (ECCV)*, pages 435–442. Springer, 2016. 15

[38] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015. 25

[39] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *PMLR International conference on machine learning (ICML)*, pages 2990–2999, 2016. 81

[40] Taco S Cohen and Max Welling. Steerable CNNs. *International Conference on Learning Representations (ICLR)*, 2016. 78, 81, 93, 95, 99

[41] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 1

[42] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 76

[43] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019. 78

[44] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NIPS)*, 26, 2013. 80

[45] Cyxtera;. Building ai applications using deep learning. https://blog.easysol.net/building-ai-applications/, last accessed: 10-2019. xiii, 3

[46] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015. 17

[47] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. xix, 14, 37, 94

[48] Zhijie Deng, Hao Zhang, Xiaodan Liang, Luona Yang, Shizhen Xu, Jun Zhu, and Eric P Xing. Structured generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3899–3909, 2017. 12

[49] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1269–1277, 2014. 45

[50] Tejas I Dhamecha, Aastha Nigam, Richa Singh, and Mayank Vatsa. Disguise detection and face recognition in visible and thermal spectrums. In *IAPR International Conference on Biometrics (ICB)*, pages 1–8, 2013. 40

[51] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012. 10

[52] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *PMLR International conference on machine learning (ICML)*, pages 647–655, 2014. 82

[53] Xuanyi Dong, Liang Zheng, Fan Ma, Yi Yang, and Deyu Meng. Few-example object detection with model communication. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. 12

[54] Lixin Duan, Ivor W Tsang, Dong Xu, and Stephen J Maybank. Domain transfer SVM for video concept detection. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1375–1381, 2009. 16

[55] Lixin Duan, Dong Xu, and Shih-Fu Chang. Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1338–1345, 2012. 16

[56] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2780–2789, 2018. 20

[57] ER Eggenberger and JH Pula. Neuro-ophthalmology in medicine. In *Aminoff's Neurology and General Medicine*, pages 479–502. Elsevier, 2014. 4

[58] Kjersti Engan, Sven Ole Aase, and J Hakon Husoy. Method of optimal directions for frame design. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (ICASSP)*, volume 5, pages 2443–2446, 1999. 29

[59] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, 11(Feb):625–660, 2010. 25

[60] Jiashi Feng and Trevor Darrell. Learning the structure of deep convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2749–2757, 2015. 25

[61] Liangjun Feng, Chunhui Zhao, and Xi Li. Bias-eliminated semantic refinement for any-shot learning. *IEEE Transactions on Image Processing (TIP)*, 31:2229–2244, 2022. 71

[62] Yaogong Feng, Xiaowen Huang, Pengbo Yang, Jian Yu, and Jitao Sang. Non-generative generalized zero-shot learning via task-correlated disentanglement and controllable samples synthesis. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9346–9355, 2022. 78

[63] Ziyong Feng, Lianwen Jin, Dapeng Tao, and Shuangping Huang. Dlanet: A manifold-learning-based discriminative feature learning network for scene classification. *Neurocomputing*, 157:11–21, 2015. 25

[64] Benjamin Feuer, Ameya Joshi, Minh Pham, and Chinmay Hegde. Distributionally robust classification on a data budget. *Transactions on Machine Learning Research (TMLR)*, 2023. 76

[65] D Foley. Considerations of sample and feature size. *IEEE Transactions on Information Theory (TIT)*, 18(5):618–626, 1972. 2, 3

[66] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2121–2129, 2013. 70, 71

[67] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016. 42

[68] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3584–3593, 2017. 42, 49, 81

[69] Yanhai Gan, Jun Liu, Junyu Dong, and Guoqiang Zhong. A PCA-based convolutional network. *arXiv preprint arXiv:1505.03703*, 2015. 25, 26

[70] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems (NIPS)*, 2022. 106

[71] Rui Gao, Xingsong Hou, Jie Qin, Li Liu, Fan Zhu, and Zhao Zhang. A joint generative model for zero-shot learning. In *European Conference Computer Vision (ECCV)*, pages 631–646. Springer, 2018. 59, 60, 70, 71

[72] Yi Gao, Chenwei Tang, and Jiancheng Lv. Cluster-based contrastive disentangling for generalized zero-shot learning. *arXiv preprint arXiv:2203.02648*, 2022. 70, 71

[73] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 10727–10737, 2018. 42, 76, 78, 82, 83, 84, 101

[74] Rohan Ghosh and Anupam K Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1906.03861*, 2019. 78, 81

[75] Soumyadeep Ghosh, Richa Singh, and Mayank Vatsa. On learning density aware embeddings. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 19, 76

[76] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (ICAIS)*, volume 9, pages 249–256, 2010. xiv, 30, 34

[77] Lamha Goel, Mayank Vatsa, and Richa Singh. LC-DECAL: Label consistent deep collaborative learning for face recognition. In *IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8, 2019. 17

[78] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 2009. 81

[79] Gaurav Goswami, Romil Bhardwaj, Richa Singh, and Mayank Vatsa. MDLFace: Memorability augmented deep learning for video face recognition. In *IEEE international joint conference on biometrics (IJCB)*, pages 1–7, 2014. 40

[80] Gaurav Goswami, Paritosh Mittal, Angshul Majumdar, Mayank Vatsa, and Richa Singh. Group sparse representation based classification for multi-feature multimodal biometrics. *Information Fusion*, 32:3–12, 2016. 17

[81] Joseph A Greenwood and Marion M Sandomire. Sample size required for estimating the standard deviation as a per cent of its true value. *Journal of the American Statistical Association*, 45(250):257–260, 1950. 2

[82] Mislav Grgic, Kresimir Delac, and Sonja Grgic. Scface–surveillance cameras face database. *Multimedia tools and applications (MTA)*, 51:863–879, 2011. 19

[83] Yuchen Guo, Guiguang Ding, Jungong Han, and Yue Gao. Zero-shot learning with transferred samples. *IEEE Transactions on Image Processing (TIP)*, 26(7):3277–3290, 2017. 70

[84] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006. 9, 18

[85] Tao Han, Junyu Gao, Yuan Yuan, and Qi Wang. Unsupervised semantic aggregation and deformable template matching for semi-supervised learning. *Advances in Neural Information Processing Systems (NIPS)*, 33:9972–9982, 2020. 36

[86] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. 38

[87] Ben Harwood, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2821–2829, 2017. 20

[88] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 820–828, 2016. 12

[89] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. 5

[90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. xiv, 30, 34

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 9, 14, 20, 23, 29, 32, 33, 38, 76, 93, 95

[92] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 16, 76

[93] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 41

[94] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 15

[95] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 41

[96] Tin Kam Ho. Random decision forests. In *ICDAR*, volume 1, pages 278–282, 1995. 1

[97] Robert V Hogg, Elliot A Tanis, and Dale L Zimmerman. *Probability and statistical inference*, volume 993. Macmillan New York, 1977. 2

[98] Seunghoon Hong, Suha Kwak, and Bohyung Han. Weakly supervised learning with deep convolutional neural networks for semantic segmentation: Understanding semantic layout of images with minimum human supervision. *IEEE Signal Processing Magazine (SPM)*, 34(6):39–49, 2017. 5

[99] Miklós Z Horváth, Mark Niklas Müller, Marc Fischer, and Martin Vechev. Boosting randomized smoothing with variance reduced classifiers. *arXiv preprint arXiv:2106.06946*, 2021. 88, 89

[100] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Exploiting unsupervised inputs for accurate Few-Shot classification. *arXiv preprint arXiv:2001.09849*, 2020. 92

[101] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Leveraging the feature distribution in transfer-based Few-Shot learning. *International Conference on Artificial Neural Networks (ICANN)*, 2021. 80, 98, 99

[102] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International conference on machine learning (ICML)*, pages 1587–1596. JMLR. org, 2017. 67

[103] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 9, 14, 20, 23, 32, 38, 76

[104] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, 07-49, University of Massachusetts, Amherst, 2007. 38

[105] He Huang, Changhu Wang, Philip S Yu, and Chang-Dong Wang. Generative dual adversarial network for generalized zero-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 801–810, 2019. 9, 12, 60, 71

[106] Dat Huynh and Ehsan Elhamifar. A shared multi-attention framework for multi-label zero-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8776–8786, 2020. 17

[107] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *PMLR International conference on machine learning (ICML)*, 2015. 17

[108] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2146–2153, 2009. 25

[109] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the ACM international conference on Multimedia (ICM)*, pages 547–556, 2014. 12

[110] Wei Jiang, Eric Zavesky, Shih-Fu Chang, and Alex Loui. Cross-domain learning methods for high-level visual concept classification. In *IEEE International Conference on Image Processing (ICIP)*, pages 161–164, 2008. 16

[111] Thomas Joyce, Agisilaos Chartsias, and Sotirios A Tsaftaris. Robust multi-modal mr image synthesis. In *International Conference Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 347–355. Springer, 2017. 13

[112] Laveen Kanal and B Chandrasekaran. On dimensionality and sample size in statistical pattern classification. *Pattern recognition (PR)*, 3(3):225–234, 1971. 2

[113] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. Locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1412.5104*, 2014. 81

[114] Rohit Keshari, Soumyadeep Ghosh, Akshay Agarwal, Richa Singh, and Mayank Vatsa. Mobile periocular matching with pre-post cataract surgery. In *IEEE International Conference on Image Processing (ICIP)*, pages 3116–3120, 2016. 37, 39, 55, 90

[115] Rohit Keshari, Soumyadeep Ghosh, Saheb Chhabra, Mayank Vatsa, and Richa Singh. Unravelling small sample size problems in the deep learning world. In *BigMM*, pages 134–143, 2020. 2

[116] Rohit Keshari, Richa Singh, and Mayank Vatsa. Guided dropout. In *Proceedings of conference the Association for the Advancement of Artificial Intelligence (AAAI)*, volume 33, pages 4065–4072, 2019. 9, 81

[117] Rohit Keshari, Richa Singh, and Mayank Vatsa. Generalized zero-shot learning via overcomplete distribution. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 20

[118] Rohit Keshari, Mayank Vatsa, Richa Singh, and Afzel Noore. Learning structure and strength of CNN filters for small sample size training. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9349–9358, 2018. 15, 49, 76, 80, 86, 87, 93, 95, 98, 99

[119] Youngeun Kim, Donghyeon Cho, Kyeongtak Han, Priyadarshini Panda, and Sungeun Hong. Domain adaptation without source data. *IEEE Transactions on Artificial Intelligence (TAI)*, 2(6):508–518, 2021. 76

[120] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2575–2583, 2015. 41, 49

[121] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 971–980, 2017. 41, 49, 81

[122] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *International conference on Machine Learning Workshop (MLW)*, volume 2. Lille, 2015. 40, 98

[123] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3174–3183, 2017. 70, 71

[124] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In *European Conference Computer Vision (ECCV)*, pages 491–507. Springer, 2020. 93, 95, 98, 99

[125] Xiangtao Kong, Xina Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Reflash dropout in image super-resolution. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6002–6012, 2022. 78

[126] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. xix, 26, 32, 49, 50, 53

[127] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 9, 15, 16, 17, 23, 76

[128] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2539–2547, 2015. 11

[129] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4281–4289, 2018. 59, 62, 66, 70, 71, 73

[130] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011. xix, 26, 32, 37, 39, 40, 90

[131] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017. 4

[132] Aditya Lakra, Pavani Tripathi, Rohit Keshari, Mayank Vatsa, and Richa Singh. SegDenseNet: Iris segmentation for Pre-and-Post cataract surgery. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 3150–3155, 2018. 17

[133] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 951–958, 2009. 66, 68

[134] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(3):453–465, 2014. 59, 68, 69

[135] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *Proceedings of conference the Association for the Advancement of Artificial Intelligence (AAAI)*, volume 1, page 3, 2008. 57, 59, 69

[136] Siddique Latif, Muhammad Usman, Sanaullah Manzoor, Waleed Iqbal, Junaid Qadir, Gareth Tyson, Ignacio Castro, Adeel Razi, Maged N Kamel Boulos, Adrian Weller, et al. Leveraging data science to combat COVID-19: A comprehensive review. *IEEE Transactions on Artificial Intelligence (TAI)*, 1(1):85–103, 2020. 75

[137] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 23

[138] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. xix, 26, 32, 49, 53, 90

[139] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999. 1

[140] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–104, 2004. 26, 32

[141] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the connectionist models summer school*, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988. 30

[142] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999. 29

[143] Enlin Li, Liwei Wang, Qiuju Xie, Rui Gao, Zhongbin Su, and Yonggang Li. A novel deep learning method for maize disease identification based on small sample-size and complex background datasets. *Ecological Informatics*, 75:102011, 2023. 75

[144] Lianfa Li. High-resolution mapping of aerosol optical depth and ground aerosol coefficients for mainland china. *Remote Sensing (RS)*, 13(12):2324, 2021. 76

[145] Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. Visual question generation as dual task of visual question answering. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6116–6124, 2018. 13

[146] Jian Liang, Ran He, Zhenan Sun, and Tieniu Tan. Aggregating randomized clustering-promoting invariant projections for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(5):1027–1042, 2018. 13

[147] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, et al. Elastic machine learning algorithms in Amazon sagemaker. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 731–737, 2020. 89

[148] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6662–6672, 2019. 78

[149] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256KB memory. *Advances in Neural Information Processing Systems (NIPS)*, 2022. 105

[150] Liang Lin, Keze Wang, Deyu Meng, Wangmeng Zuo, and Lei Zhang. Active self-paced learning for cost-effective and progressive face identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(1):7–19, 2017. 12

[151] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 76

[152] Minghui Liu, Tianshu Xie, Xuan Cheng, Jiali Deng, Meiyi Yang, Xiaomin Wang, and Ming Liu. Focuseddropout for convolutional neural network. *Applied Sciences*, 12(15):7682, 2022. 42

[153] Shichen Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Generalized zero-shot learning with deep calibration network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2009–2019, 2018. 59, 70, 71

[154] Ying Liu, Zhen Xu, and Chen Zhang. Distributed semisupervised partial label learning over networks. *IEEE Transactions on Artificial Intelligence (TAI)*, 3(3):414–425, 2022. 76

[155] Zhenhua Liu, Jizheng Xu, Xiulian Peng, and Ruiqin Xiong. Frequency-domain dynamic pruning for convolutional neural networks. *Advances in neural information processing systems (NIPS)*, 31, 2018. 10

[156] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015. 16, 17, 76

[157] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2200–2207, 2013. 11

[158] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer joint matching for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1410–1417, 2014. 11

[159] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 136–144, 2016. 16

[160] Yang Long, Li Liu, Fumin Shen, Ling Shao, and Xuelong Li. Zero-shot learning using synthesised unseen visual data with diffusion regularisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(10):2498–2512, 2018. 59, 60

[161] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D Cubuk. Improving robustness without sacrificing accuracy with patch Gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019. 77, 82

[162] Zelun Luo, Jun-Ting Hsieh, Lu Jiang, Juan Carlos Niebles, and Li Fei-Fei. Graph distillation for action detection with privileged modalities. In *European Conference Computer Vision (ECCV)*, pages 166–183, 2018. 16

[163] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 165–177, 2017. 76, 80

[164] Lei Ma, Hongliang Li, Fanman Meng, Qingbo Wu, and King Ngi Ngan. Discriminative deep metric learning for asymmetric discrete hashing. *Neurocomputing*, 380:115–124, 2020. 18

[165] Peirong Ma, Hong Lu, Bohong Yang, and Wu Ran. Gan-mvae: A discriminative latent feature generation framework for generalized zero-shot learning. *Pattern Recognition Letters (PRL)*, 155:77–83, 2022. 76

[166] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research (JMLR)*, 11(Jan):19–60, 2010. 29

[167] Angshul Majumdar, Richa Singh, and Mayank Vatsa. Face verification via class sparsity based supervised encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(6):1273–1280, 2016. 1

[168] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2218–2227, 2020. 80, 99

[169] Rongfu Mao, Haichao Zhu, Linke Zhang, and Aizhi Chen. A new method to assist small data set neural network learning. In *IEEE international conference on intelligent systems design and applications (ISDA)*, volume 1, pages 17–22, 2006. 25

[170] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. 69

[171] Poorya Mianjy, Raman Arora, and Rene Vidal. On the implicit bias of dropout. *arXiv preprint arXiv:1806.09777*, 2018. 44, 45, 47, 48

[172] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 464–471, 2000. 76

[173] Erik Gundersen Miller. *Learning from one example in machine vision by sharing probability densities*. PhD thesis, Massachusetts Institute of Technology, 2002. 57

[174] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 25

[175] Ashish Mishra, Shiva Krishna Reddy, Anurag Mittal, and Hema A Murthy. A generative model for zero shot learning using conditional variational autoencoders. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 2188–2196, 2018. 59, 70, 71

[176] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. 6

[177] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4500–4509, 2018. 12

[178] Shruti Nagpal, Maneet Singh, Richa Singh, Mayank Vatsa, Afzel Noore, and Angshul Majumdar. Face sketch matching via coupled deep transform learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5429–5438, 2017. 40

[179] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010. 41

[180] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in neural information processing systems workshop (NIPSW)*, volume 2011, page 5, 2011. xix, 49, 50, 53, 90

[181] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1279–1287, 2010. 25

[182] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *International Conference on Learning Representations (ICLR)*, 2021. 24

[183] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems (NIPS)*, 34, 2021. 24

[184] Ishan Nigam, Rohit Keshari, Mayank Vatsa, Richa Singh, and Kevin Bowyer. Phacoemulsification cataract surgery affects the discriminative capacity of iris pattern recognition. *Scientific reports*, 9(1):11139, 2019. 99

[185] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013. 70, 71

[186] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992. 41

[187] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016. 20

[188] Emmanuel Okafor, Rik Smit, Lambert Schomaker, and Marco Wiering. Operational data augmentation in classifying single aerial images of animals. In *IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 354–360, 2017. 11

[189] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 864–873, 2016. 5

[190] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. *arXiv preprint arXiv:1703.08961*, 2017. 25, 34, 36, 38

[191] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks (TNN)*, 22(2):199–210, 2010. 11

[192] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359, 2009. 13, 17

[193] Chanwoo Park, Sangdoo Yun, and Sanghyuk Chun. A unified analysis of mixed sample data augmentation: A loss function perspective. *Advances in Neural Information Processing Systems (NIPS)*, 2022. 76

[194] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference (BMVC)*, 2015. 38

[195] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Advances in neural information processing systems workshop (NIPSW)*, 2017. 48, 89

[196] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2751–2758, 2012. 68, 69

[197] Juan C Pérez, Motasem Alfarra, Guillaume Jeanneret, Adel Bibi, Ali Thabet, Bernard Ghanem, and Pablo Arbeláez. Gabor layers enhance network robustness. In *European Conference Computer Vision (ECCV)*, pages 450–466. Springer, 2020. 85

[198] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4119–4128, 2018. 16

[199] Alireza Rahimpour and Hairong Qi. Class-discriminative feature embedding for meta-learning based few-shot classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3179–3187, 2020. 80, 98

[200] Shafin Rahman, Salman Khan, and Fatih Porikli. Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts. *arXiv preprint arXiv:1803.06049*, 2018. 5

[201] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *ACM International conference on machine learning (ICML)*, pages 759–766. ACM, 2007. 57

[202] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3567–3575, 2016. 12, 13

[203] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3236–3246, 2017. 11

[204] Sarunas J Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, (3):252–264, 1991. 2, 3

[205] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 15

[206] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for PyTorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020. 90

[207] Sylvia Richardson and Peter J Green. On Bayesian analysis of mixtures with an unknown number of components (with discussion). *RSS: series B (statistical methodology)*, 59(4):731–792, 1997. 62

[208] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *PMLR International conference on machine learning (ICML)*, pages 2152–2161, 2015. 70, 71

[209] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 16

[210] Arunabha M Roy and Jayabrata Bhaduri. Real-time growth stage detection model for high degree of occultation using DenseNet-fused YOLOv4. *Computers and Electronics in Agriculture (CEA)*, 193:106694, 2022. 78

[211] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5

[212] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016. 6

[213] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 15, 17

[214] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference Computer Vision (ECCV)*, pages 213–226. Springer, 2010. 11

[215] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal processing letters (SPL)*, 24(3):279–283, 2017. 11, 76

[216] Hojjat Salehinejad, Shahrokh Valaee, Tim Dowdell, and Joseph Barfett. Image augmentation using radial transform for training deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3016–3020, 2018. 11

[217] Anush Sankaran, Gaurav Goswami, Mayank Vatsa, Richa Singh, and Angshul Majumdar. Class sparsity signature based restricted Boltzmann machine. *Pattern Recognition*, 61:674–685, 2017. 17

[218] Anush Sankaran, Mayank Vatsa, Richa Singh, and Angshul Majumdar. Group sparse autoencoder. *IVC*, 60:64–74, 2017. 17

[219] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *PMLR International Conference on Machine Learning (ICML)*, pages 1842–1850, 2016. 40, 76, 80, 98

[220] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8247–8255, 2019. 12, 60, 71

[221] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. 9, 18

[222] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2110–2118, 2016. 16, 17

[223] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 806–813, 2014. 82

[224] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3400–3409, 2017. 16

[225] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 761–769, 2016. 20

[226] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2107–2116, 2017. 12

[227] Sahar Siddiqui, Mayank Vatsa, and Richa Singh. Face recognition for newborns, toddlers, and pre-school children: A deep learning approach. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 3156–3161, 2018. 17

[228] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, 2003. 76

[229] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for Few-Shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4136–4145, 2020. 81, 99

[230] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 9, 23, 38

[231] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4077–4087, 2017. 19

[232] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1857–1865, 2016. 18

[233] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of*

*Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014. 9, 17, 41, 49, 51, 76, 78, 81, 82, 83

[234] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep Boltzmann machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2222–2230, 2012. 13

[235] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1433–1440, 2008. 11

[236] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7251–7259, 2019. 20

[237] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1988–1996, 2014. 18

[238] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1199–1208, 2018. 19

[239] Saksham Suri, Anush Sankaran, Mayank Vatsa, and Richa Singh. On matching faces with alterations due to plastic surgery and disguise. In *IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–7. IEEE, 2018. 9, 76

[240] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 1, 9, 14, 20, 23

[241] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016. 12

[242] Snigdha Tariyal, Angshul Majumdar, Richa Singh, and Mayank Vatsa. Deep dictionary learning. *IEEE Access*, 4:10096–10109, 2016. 15, 26, 29

[243] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011. 17

[244] Alexander V Terekhov, Guglielmo Montone, and J Kevin O'Regan. Knowledge transfer in deep block-modular neural networks. In *Proceedings of the International Conference on Biomimetic and Biohybrid Systems (BBS)*, pages 268–279. Springer, 2015. 15, 17

[245] Rajat M Thomas, Willem Bruin, Paul Zhutovsky, and Guido van Wingen. Dealing with missing data, small sample sizes, and heterogeneity in machine learning studies of brain disorders. In *Machine Learning*, pages 249–266. Elsevier, 2020. 4

[246] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, pages 640–646, 1996. 57

[247] TinyImageNet. Tiny ImageNet tiny imagenet visual recognition challenge. https://tiny-imagenet.herokuapp.com/, 2018. Accessed: 21st-Jan-2020. xix, 49, 50, 53, 90

[248] Angel Torrado-Carvajal, Joaquin L Herraiz, Eduardo Alcain, Antonio S Montemayor, Lina Garcia-Canamaque, Juan A Hernandez-Tamames, Yves Rozenholc, and Norberto Malpica. Fast Patch-Based Pseudo-CT Synthesis from T1-Weighted MR Images for PET/MR Attenuation Correction in Brain Studies. *Journal of Nuclear Medicine (NM)*, 57(1):136–143, 2016. 13

[249] Ivana Tošić and Pascal Frossard. Dictionary learning. *IEEE Signal Processing Magazine (SPM)*, 28(2):27–38, 2011. 26, 29

[250] Asher Trockman and J Zico Kolter. Patches are all you need? *Transactions on Machine Learning Research (TMLR)*, 2023. 93

[251] Dimitrios Tsourounis, Ilias Theodorakopoulos, Elias N Zois, and George Economou. From text to signatures: Knowledge transfer for efficient deep feature learning in offline signature verification. *Expert Systems with Applications (ESA)*, 189:116136, 2022. 76

[252] Lukas Tuggener, Mohammadreza Amirian, Fernando Benites, Pius von Däniken, Prakhar Gupta, Frank-Peter Schilling, and Thilo Stadelmann. Design patterns for resource-constrained automated deep-learning methods. *AI*, 1(4):510–538, 2020. 75

[253] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4068–4076, 2015. 16

[254] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7167–7176, 2017. 12

[255] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 11

[256] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009. 17

[257] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. In *International Conference Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 210–218. Springer, 2018. 76

[258] Vinay Kumar Verma and Piyush Rai. A simple exponential family framework for zero-shot learning. In *Joint European conference on machine learning and knowledge discovery in databases (ECMLKDD)*, pages 792–808. Springer, 2017. 70

[259] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3630–3638, 2016. 19, 38, 40, 76, 80, 92, 98

[260] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5495–5504, 2018. 13

[261] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The CalTech-USCD birds-200-2011 dataset. 2011. xx, 90, 98, 99

[262] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *PMLR International conference on machine learning (ICML)*, pages 1058–1066, 2013. 9, 17, 33, 41, 81, 87

[263] Jason Wang, Luis Perez, et al. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit (CNNVR)*, 11(2017):1–8, 2017. 11

[264] Jie Wang, Jianqing Liang, Jiye Liang, and Kaixuan Yao. GUIDE: Training deep graph neural networks via guided dropout over edges. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022. 78

[265] Joseph Wang, Kirill Trapeznikov, and Venkatesh Saligrama. Efficient learning by directed acyclic graph for resource constrained prediction. *Advances in Neural Information Processing Systems (NIPS)*, 28, 2015. 75

[266] Sida Wang and Christopher Manning. Fast dropout training. In *International Conference on Machine Learning (ICML)*, pages 118–126, 2013. 41

[267] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on Few-Shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020. 76

[268] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2471–2480, 2017. 15

[269] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems (NIPS)*, pages 7029–7039, 2017. 5

[270] Yunhe Wang, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. Cnnpack: Packing convolutional neural networks in the frequency domain. *Advances in neural information processing systems (NIPS)*, 29, 2016. 10

[271] Zhuoyi Wang, Yigong Wang, Bo Dong, Sahoo Pracheta, Kevin Hamlen, and Latifur Khan. Adaptive margin based deep adversarial metric learning. In *Big Data Security*, pages 100–108, 2020. 20

[272] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD birds 200. 2010. 68, 69

[273] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference Computer Vision (ECCV)*, pages 499–515. Springer, 2016. 62

[274] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–534, 2011. 38

[275] Dan Wu, Jiasong Wu, Rui Zeng, Longyu Jiang, Lotfi Senhadji, and Huazhong Shu. Kernel principal component analysis network for image classification. *arXiv preprint arXiv:1512.06337*, 2015. 25

[276] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the ACM international conference on management of data (ICMD)*, pages 1301–1316, 2018. 13

[277] Shiling Wu and Dunlu Peng. Pre-smats: A multi-task learning based prediction model for small multi-stage seasonal time series. *Expert Systems with Applications (ESA)*, page 117121, 2022. 76

[278] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light CNN for deep face representation with noisy labels. *arXiv preprint arXiv:1511.02683*, 2015. 38

[279] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 69–77, 2016. 70, 71

[280] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. xix, 59, 69, 70

[281] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5542–5551, 2018. 71

[282] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4582–4591, 2017. 58

[283] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems (NIPS)*, 33:6256–6268, 2020. 36

[284] Lei Xing, Shuai Shao, Weifeng Liu, Anxun Han, Xiangshuai Pan, and Bao-Di Liu. Learning task-specific discriminative embeddings for few-shot image classification. *Neurocomputing*, 488:1–13, 2022. 81, 99

[285] Wei Xiong, Bo Du, Lefei Zhang, Ruimin Hu, and Dacheng Tao. Regularizing deep convolutional neural networks with a structured decorrelation constraint. In *International Conference on Data Mining (ICDM)*, 2016. 25

[286] Pengcheng Xu, Xiaobo Ji, Minjie Li, and Wencong Lu. Small data machine learning in materials science. *npj Computational Materials*, 9(1):42, 2023. 75

[287] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 11

[288] Pew-Thian Yap, Xudong Jiang, and Alex Chichung Kot. Two-dimensional polar harmonic transforms for invariant image representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(7):1259–1270, 2009. 11

[289] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4133–4141, 2017. 16

[290] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014. 15, 81, 85

[291] Xiang Yuan, Gong Cheng, Kebing Yan, Qinghua Zeng, and Junwei Han. Small object detection via coarse-to-fine proposal generation and imitation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6317–6327, 2023. 76

[292] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 814–823, 2017. 20

[293] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ACM International conference on machine learning (ICML)*, page 114, 2004. 13

[294] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference Computer Vision (ECCV)*, pages 818–833. Springer, 2014. 34

[295] Rui Zeng, Jiasong Wu, Lotfi Senhadji, and Huazhong Shu. Tensor object classification via

multilinear discriminant analysis network. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015. 25, 26

[296] Yao Zeng, Xusheng Liu, Lintan Sun, Wenzhong Li, Yuchu Fang, and Sanglu Lu. Iterative deep model compression and acceleration in the frequency domain. In *Asian Conference on Machine Learning (ACML)*, pages 331–346. PMLR, 2021. 10

[297] Yuyuan Zeng, Tao Dai, and Shu-Tao Xia. Corrdrop: Correlation based dropout for convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3742–3746, 2020. 42

[298] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2736–2746, 2022. 78

[299] Haofeng Zhang, Yang Long, Yu Guan, and Ling Shao. Triple verification network for generalized zero-shot learning. *IEEE Transactions on Image Processing (TIP)*, 28(1):506–517, 2019. 60

[300] Haofeng Zhang, Yang Long, Li Liu, and Ling Shao. Adversarial unseen visual feature synthesis for zero-shot learning. *Neurocomputing*, 329:12–20, 2019. 12, 59, 60, 70, 71

[301] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision. *Advances in Neural Information Processing Systems (NIPS)*, 2022. 36

[302] Xiaofeng Zhang, Zhangyang Wang, Dong Liu, Qifeng Lin, and Qing Ling. Deep adversarial data augmentation for extremely low data regimes. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 31(1):15–28, 2020. 12

[303] Xin-Yu Zhang, Taihong Xiao, Haolin Jia, Ming-Ming Cheng, and Ming-Hsuan Yang. Semi-supervised learning with meta-gradient. *arXiv preprint arXiv:2007.03966*, 2020. 36

[304] Ziming Zhang and Venkatesh Saligrama. Learning joint feature adaptation for zero-shot recognition. *arXiv preprint arXiv:1611.07593*, 2016. 70, 71

[305] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *International Conference on Learning Representations (ICLR)*, 2021. 24

[306] Xiantong Zhen, Haoliang Sun, Yingjun Du, Jun Xu, Yilong Yin, Ling Shao, and Cees Snoek. Learning to learn kernels with variational random features. In *PMLR International conference on machine learning (ICML)*, pages 11409–11419, 2020. 81, 99

[307] Wenzhao Zheng, Jiwen Lu, and Jie Zhou. Deep metric learning via adaptive learnable assessment. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2960–2969, 2020. 18

[308] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. 33