ENTROPY-BASED OPTIMIZATION STRATEGIES FOR CONVOLUTIONAL NEURAL NETWORKS

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisor

Assoc. Prof. Roopak Sinha

Assoc. Prof. Wei Qi Yan

Prof. Stephen MacDonell

20th January 2021

By

Nidhi Gowdra

School of Engineering, Computer and Mathematical Sciences

Abstract

Deep convolutional neural networks are state-of-the-art for image classification and significant strides have been made to improve neural network model performance which can now even outperform human-level abilities. However, these gains have been achieved through increased model depths and rigorous specialized manual fine-tuning of model HyperParameters (HPs). These strategies cause considerable over-parameterization and elevated complexity in Convolutional Neural Network (CNN) model training. Training over-parameterized CNN models tend to induce afflictions like overfitting, increased sensitivity to noise and decreased generalization ability which contribute to deterioration of model performance. Furthermore, training over-parameterized CNN models require specialized regimes and vast computing power subsequently increasing the complexity and difficulty of training.

In this thesis, we develop several novel entropy-based techniques to abate the effects of over-parameterization, reduce the number of manually tuned HPs, increase generalization ability and, enhance performance of CNN models. Specifically, we examine information propagation and feature extraction/generation & representation in CNNs. Armed with this knowledge, we develop a heuristic and several optimization strategies to simplify model training and improve model performance by addressing the problem of over-parameterization in CNNs. We cultivate the techniques in this thesis utilizing quantitative metrics such as Shannon's Entropy (SE), Maximum Entropy (ME) and Signal-to-Noise (SNR) ratio.

Our methodology involves a multi-faceted approach of incorporating iterative and continuous integration of quantitatively defined feedback loops which allows us to test numerous research hypotheses efficiently using the design science research framework. We start off by exploring and understanding the hierarchical feature extraction & representational capabilities of CNNs. Through our experimentation we were able to explore the sparsity of feature representations and analyze the underlying learning mechanisms in CNNs for non-convex optimization problems such as image classification. Equipped with this knowledge, we were able to experimentally demonstrate and validate the notion that for low and high quality input data (determined through ME and SNR measures) using deeper and shallower networks could lead to the phenomena of information underflow and overflow respectively, degrading classification performance.

To mitigate the negative effects of information underflow and overflow in the context of kernel saturation, we propose and evaluate a novel hypothesis of augmenting the data distribution of the input dataset with negative images. Our experimental results generated a classification accuracy increase of 3-7% on various datasets. One of the limitations argued against the validity of our novel augmentation was model training time, in particular, models require large amounts of computing power and time to train. In order to address these criticisms, we theorize a SE-based heuristic to resolve the problem of over-parameterization by forcing feature abstractions in the convolutional layers up to its theoretic limit as defined by their SE measure. The SE-based model trained 45.22% faster without compromising classification accuracy when compared to deeper models. Further arguments were posed relating to model training afflictions such as overfitting and generalizability.

To mitigate the speculations raised around model training afflictions such as overfitting and generalizability in deep CNN models, we introduce a Maximum Entropy-based Learning raTE enhanceR (MELTER), to dynamically schedule and adapt model learning during training, and a Maximum Categorical Cross-Entropy (MCCE) loss function

derived from the commonly used Categorical Cross-Entropy (CCE) loss function, to reduce model overfitting. MELTER and MCCE utilize a priori knowledge of the input data to curtail a few risks encountered during model training which affect performance such as, sensitivity to random noise, overfitting to the training data and lack of generalizability to new unseen data. To this extent, MELTER outperforms manually tuned models by 2-6% on various benchmarking datasets by exploring a larger solution space. MCCE-trained models showed a reduction in overfitting by up to 5.4% and outperform Categorical Cross-Entropy (CCE) trained models in terms of classification accuracy by up to 6.17% on two facial (ethnicity) recognition datasets, colorFERET and UTKFace, along with standard benchmarking datasets such as CIFAR-10 and MNIST.

Through these series of experiments, we can conclude that, entropy-based optimization strategies for tuning HPs of deep learning models are viable and either maintain or outperform baseline classification accuracies achieved by networks trained using traditional methods. Furthermore, the entropy-based optimization methods outlined in this thesis also mitigate several well-known training afflictions such as overfitting, lack of generalizability and rate of convergence while eliminating manual fine-tuning of HPs.

Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of candidate

Publications

Gowdra, N. & Sinha, R. & MacDonell, S. (2020) "Examining convolutional feature extraction using Maximum Entropy (ME) and Signal-to-Noise Ratio (SNR) for image classification", IECON 2020-46th Annual Conference of the IEEE Industrial Electronics Society, IEEE.

Gowdra, N. & Sinha, R. & MacDonell, S. (2020) "Examining and Mitigating Kernel Saturation in Convolutional Neural Networks using Negative Images", IECON 2020-46th Annual Conference of the IEEE Industrial Electronics Society, IEEE.

Gowdra, N. & Sinha, R. & MacDonell, S. & WeiQi, Y. (2021) "Mitigating severe over-parameterization in deep convolutional neural networks through forced feature abstraction and compression with an entropy-based heuristic", Pattern Recognition Journal, Elsevier Ltd.

Acknowledgements

Firstly, I would like to start-off by quoting Steve Jobs,
'Great things are never done by one person; they're done by a team of people'.

To that extent, I would like to express my sincere gratitude to the team of incredible souls who supported me in producing this piece of literary work, this thesis is dedicated to you all!.

Obviously, the first set of amazing people I would like to thank are my family; my loving parents (Dr. Nijagunappa G.M and Mrs. Nanda K.R) and caring brother (Mr. Nakul Gowdra), for shielding me from some of the harsh realities of life. My sister-in-law for her continued support in my endeavors.

My sincere appreciation goes towards my team of advisors, Assoc. Prof. Roopak Sinha and Prof. Stephen MacDonell, who stepped up at the most difficult of times with their unwavering support in helping me through this arduous journey. I would also like to express my recognition of the wonderful support of Assoc. Prof. Wei Qi Yan, who agreed to join the team with enthusiasm.

My heartfelt gratitude goes out to one of my best friends and colleague Subash Humagain, who has been invaluable in helping me navigate through this grueling phase of my life. I would like to thank Barry/Badger Dowdeswell, a positively fantastic friend and colleague who has been immensely encouraging throughout the Ph.D. and for the past 5 years. Additionally, I would like to express my honest appreciation to Matthew Kuo, a colleague and budding friend for the years I spent assisting his teaching of undergraduate courses.

My sincere gratefulness goes out to Prof. Ajit Narayanan for his precious guidance and insightful feedback in shaping my research along with the Auckland University of Technology (AUT) who provided me an opportunity to study with a partial scholarship. I would also like to heartily acknowledge the Graduate Research School, it's former Dean, Prof. Marion Jones; current Dean, Prof. Mark Orams and manager, Martin Wilson along with the Head of School (Engineering, Computer and Mathematical Sciences), Dr. Rosser Johnson for their wisdom and imparting knowledge around boardroom courtesy, motions, proceedings, university procedures and protocols. It was a pleasure meeting you all!

Finally, I would like to express my immense admiration for the New-Zealand government, its crown institutions such as, Callaghan Innovation under the Ministry of Business, Innovation and Employment (MBIE), its people, James Muir; and Auckland Tourism, Events and Economic Development (ATEED), especially, Paula Cooper and Deb Wai. These institutions and all the amazing people were critical in securing funding to launch and grow my own startup InfuseAI Limited, which provided the majority of computational resources utilized in this thesis along with the New Zealand e-Science Infrastructure (NeSI). Without these resources, none of the experiments presented in this thesis simply would not have been possible considering the immense computational resource constraint at AUT.

To quote Helen Keller,
'Alone we can do so little; together we can do so much'

Thank you all for your understanding and cooperation! We have had some great times. We have endured and persevered through it all to come out the other side stronger! Professionally, achieving a doctorate degree is not as challenging as most make it out to be but, personally, this journey has taken its toll. The silver lining is that this is a once-in-a-lifetime journey and hopefully it will prove to be worth the effort.

Signing off with the words of Rob Siltanen, enshrined in the famous Apple Inc. advertisement (1997),

Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.

Chapter 1

Introduction

In the late 20th century, an epochal and expeditious shift from an era shaped by the Industrial Revolution to economies based largely upon information technology was witnessed. The computer/digital/information age as it is referred to, relies primarily on a knowledge economy that rewards critical thinking and problem solving to generate economic value. Since the advent of the first micro-processors in 1971 and fuelled by modern inventions such as the fiber-optic cable and the internet, we generate an estimated 2.5 quintillion bytes of digital data per year (Marr, 2018). Analyzing this stored data is beneficial for numerous real-world applications, such as marketing, enhancing industrial and commercial processes, demand forecasting, improving healthcare services and broadening the reach of educational courses. These are just a few areas where Big-Data analysis is valuable. However, the predominant challenge in exploiting Big-Data to provide advantageous business insights is the need for complex processing in specific application domains.

Recent trends in technological innovation extensively relies on automating the process of analyzing digital data using complex computer algorithms. Traditional approaches using statistical analyses or signal processing are incapable of accommodating such large amounts of data (D. J. Becker et al., 1995) and lack adequate information

extraction capabilities (Wiatowski & Bölcskei, 2018). Therefore, to quote Marshall McLuhan (McLuhan & Gordon, 2013),

'Faced with information overload, we have no alternative but pattern recognition.'

Writing a pattern recognition algorithm for computers is especially challenging since the underlying patterns in digital data are indistinct and unnoticeable using traditional techniques. Humans are especially proficient in pattern recognition and in some instances going so far as to decipher patterns in random noise. To quote Michael Shermer (Shermer, 2011),

'Humans evolved brains that are pattern recognition machines, adept at detecting signals that enhance or threaten survival amid a very noisy world, but there is only one surefire method of proper pattern recognition, and that is science.'

Collectively, humans can help analyze the vast amount of data generated, as evidenced by the Amazon Mechanical Turk project where digital images are manually annotated by humans, but this process is financially infeasible. Enabling computers to learn patterns from a given set of digital data through the use of learning algorithms can help with the economic feasibility of analyzing data.

Over the past decade, through the utilization of increased high-compute infrastructure (primarily Graphics Processing Units (GPUs)), availability of Big-Data and development of novel machine learning algorithms, the world has witnessed an explosion in applications of machine learning pattern recognition systems (Ardakani, Condo, Ahmadi & Gross, 2018). Law enforcement, social networking websites, internet search engines, weather predictions, disaster mitigation and credit scoring systems all deploy some form of machine learning. The task of digital image classification/recognition is widely used in disciplines such as facial detection, medical diagnoses and automotive driving-automation systems which have huge social, economic and cultural significance. The implementation of machine learning algorithms for these diverse set of application domains has led to the acceleration in complexity of machine learning algorithms.

Continuous increments in the complexity of machine learning algorithms and models has led to a real-world problem of exponentially growing computational requirements. The complexity of algorithms is a function of the input data presented to the algorithms. As applications of machine learning models become increasingly varied and complicated, a subsequent growth in the complexity of the machine learning algorithms and models is required to satisfactorily identify any implicit patterns in the data. Computational overhead is the penalty and trade-off for analyzing progressively increasing complex data. Compression methods can help alleviate such trade-offs using tried and tested mathematical techniques (Shannon, Weaver & Burks, 1951), but they are imprecise and could result in undesired information losses. Active research is being undertaken in exploring alternate techniques for big-data processing.

Entropy can help distinguish useful information from random noise (Donoho, Johnstone, Stern & Hoch, 1990), which is critical for successful signal processing (Gull & Skilling, 1984). Furthermore, different entropy measures such as Shannon's Entropy (SE) (Shannon et al., 1951) and Maximum Entropy (ME) (Petrovici, Damian & Coltuc, 2018) are used in various applications to determine the theoretical limit for digital compression and signal reconstruction (Tsai, Lee & Matsuyama, 2008) respectively. ME is especially useful to reconstruct signal information from lossy sources (Macqueen & Marschak, 1975), or as a monitoring function to correct variational drifts in data (Rathore, Kumar, Rajasegarar & Palaniswami, 2017) warranting its application across diverse domains. ***In this thesis, we primarily focus and constrain our research to explore how SE and ME measures can be utilized to improve machine learning pattern recognition.***

1.1 Overview of the Research Domain

Taking inspiration from the human brain which is especially proficient in pattern recognition, mathematical abstractions of biological neurons and their neural connections paved the way for a new computing paradigm, dubbed Artificial Neural Networks (ANNs) illustrated in Figure 2.1. ANNs have enabled rapid progress to be made in the field of pattern recognition and gave rise to the current state-of-the-art Neural Networks for computer vision using convolution operations (or simply, Convolutional Neural Networks (CNNs)), illustrated in Figure 2.2 and explained thoroughly in Chapter 2. Leading CNN architectures and models outperform even human-level image recognition ability (LeCun, Bengio & Hinton, 2015).

Although state-of-the-art CNN models can outperform human-level abilities in image recognition, they have numerous limitations, such as misclassification, generalization errors (where the network cannot accurately identify new unseen images) and overfitting (where the network performance is significantly reduced for slight variances in the underlying dataset data distribution in the same set of image classes). Such limitations are addressed in this thesis, but it is imperative that we look at a high-level overview of the domain.

Challenges: Images are encoded in arrays of brightness intensities as pixels, which range from a few hundred to millions of pixels in an image. Machine learning algorithms need to analyze, transform and interpret patterns inside these arrays of brightness intensities to generate high-level semantic concepts to recognize images such as dogs or cats. The problem becomes more complex when similar patterns form part of the same high-level semantic concepts (as an example, let's say a feature, tail is part of a dog and cat). In these instances, fine-grained hierarchical features are needed (as an example, the fur, shape or size of the tail, might help distinguish the two classes). High-level semantic information can be obtained by combining multiple learning layers making

the model 'deep'.

Progress: There has been encouraging progress in the area of computer vision with state-of-the-art deep CNN architectures such as Residual Networks (ResNets) (K. He et al., 2016), aggregated residual transformations (ResNeXt) (Xie, Girshick, Dollár, Tu & He, 2017) and densely connected convolutional networks (DenseNets) (Huang et al., 2017) (discussed in Section 2.3.1) maintaining or even outperforming human-level abilities in classification tasks.

Unresolved challenges: Although there has been rapid progress in the field of computer vision, there are a few drawbacks that persist. Current CNN architectures are highly susceptible to structural noise and sensitive to the data distributions of the input dataset. CNN models are sparse in their feature representations and due to their sparsity, exploration of larger parameter spaces is required, adding to the immense computational power demanded for model training. Furthermore, manual optimization of HyperParameters, such as the model learning rate, is both time consuming and can lead to convergence issues for improper configurations. Finally, all CNN models suffer from overfitting on the training dataset to a certain degree which could become problematic for unrefined tasks on complex datasets such as ethnicity detection on facial datasets, where a propensity for misclassification is greatly induced. To test our novel hypotheses we restrict the scope of evaluation on a limited set of datasets, which are selected for their representational characteristics for various application domains.

Datasets: Several standard image benchmarking datasets have been collated and are used to measure relative performance improvements for machine learning algorithms. In this thesis, we primarily utilize three types of benchmarking datasets, namely character recognition, natural image recognition and facial (race/ethnicity) recognition. The MNIST dataset is a modified version of all available documents (to include only numbers from 0-9) sourced from the National Institute of Standards and Technology (NIST) used for handwritten character recognition. Various natural image datasets (CIFAR-10/100,

STL-10 and ImageNet32) are employed for image classification which include natural images of cats, dogs, automobiles and so on. Finally, two facial recognition datasets (UTKFace and colorFERET) are used for ethnicity recognition. The datasets used for each set of experiments are included in individual chapters addressing unresolved issues such as sensitivity to structural noise and data distributions (Chapters 3 - 4), sparse feature representations (Chapter 5), optimization of HyperParameters (Chapter 6) and reducing the tendency of CNN models to overfit on the training data (Chapter 7).

1.2 Motivation

The rapid progress in CNN model performance has been achieved through increasingly complex architectures and training regimes. The trade-off from this attrition-based approach on increasing performance is that the barriers for entry into deep learning research are getting higher and harder to reach. Research publications in top-tier conferences and practice is increasingly becoming a function of access to adequate computing and data resources along with narrower domain expertise which are progressively being concentrated into a few research institutions and high-technology companies. The research in this thesis was born out of necessity as our access to large computing infrastructures was severely limited. Therefore, the research in this thesis sought to exploit the opportunity to reduce the computational burden required for CNN model training. In our endeavor to reduce computational resources required for model training, we explore and propose several CNN optimization techniques using entropy (SE and ME) as the cornerstone for our research. To quote Albert Einstein (Einstein, 1933),

'It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.'

The question now resolves into, how can CNN models be optimized without compromising representational/learning capacity? Using entropy measures is quite intuitive since Shannon and Maximum Entropy measures quantitatively determine the theoretical compression limits in digital systems and the total amount of information in a given series of data respectively. Using apriori knowledge of these entropy measures, we employ ME-based monitoring functions to enhance CNN model training, which is both logically and mathematically sound. To quote Stephen Hawking (Hawking, 2009),

'Just like a computer, we must remember things in the order in which entropy increases. This makes the second law of thermodynamics almost trivial. Disorder increases with time because we measure time in the direction in which disorder increases. You can't have a safer bet than that!'

Therefore, the core premise for our research is to answer the following underlying question: *how could entropy measures, specifically Shannon and Maximum entropy, be utilized to optimize CNN model training?*. This high-level research question can be deconstructed to investigate several lower-level issues in the domain of computer vision. The topics for examination in this thesis addresses how entropy measures could be utilized to enhance classification performance, decrease model training times, accelerate the rates of convergence and reduce overfitting without adversely impacting model performance. Our research methodology involves a multi-faceted approach incorporating an iterative and continuous integration of quantitatively driven feedback loops, which allows us to test numerous research hypotheses efficiently using the design science research framework. Detailed descriptions of the experimental design and CNN model HyperParameters (HPs) are included in each of the individual experimental chapters (i.e. Chapters 3-7).

1.3 Contributions and Thesis Outline

Through this thesis, we contribute to the body of knowledge several CNN optimization techniques/strategies/methods that address some of the persistent challenges of current CNN models, detailed as follows:

In Chapter 2, we provide a mathematical foundation for pattern recognition using deep learning. We introduce early Artificial Neural Networks (ANNs) and describe model loss calculation for classification tasks. We also discuss high-level optimization paradigms using first order derivatives, Stochastic Gradient Descent (SGD) and Back-Propagation (BP) algorithms for non-convex optimization problems such as image classification. We then introduce regularization and normalization methods to help improve model performance. We examine the benefits and drawbacks of Convolutional Neural Networks (CNNs), analyze their high classification performance and conclude by reviewing several state-of-the-art deep CNN architectures.

In Chapter 3, we quantitatively examine the feature extraction and information propagation properties of CNNs using two metrics, Maximum Entropy (ME) and Signal-to-Noise Ratio (SNR). Using ME and SNR measures, we identify two characteristic phenomena (information overflow and underflow) that affect classification performance. CNN models are able to extract only a certain degree of information and if the extracted feature information is lower or higher than the input dataset it can lead to inaccurate model convergence. The two key takeaway points in this chapter are that high SNR and ME measures help in information propagation within the hidden layers and as such a CNN model's size should be tailored to the input dataset.

In Chapter 4, we examine the relatively unexplored phenomenon of kernel saturation in the convolutional block of a CNN model. Kernel saturation is well understood for ANNs but remains largely unstudied for convolutional kernels. We utilize the same ME and SNR metrics to quantify and mathematically hypothesize that augmenting standard

datasets with structurally and semantically similar negative images will help mitigate the effect of kernel saturation on classification accuracy. This chapter strengthens the findings presented in Chapter 3.

In Chapter 5, we explore CNN model over-parameterization and specifically investigate the sparsity of feature extraction in CNNs. We propose a heuristic based on Shannon's Entropy (SE) measure to force feature abstraction and artificially compress the solution space. Restricting unnecessary exploration of the search space should significantly decrease model training time and increase model convergence efficiency. This chapter further strengthens the findings presented in Chapter 3 along with providing insights into the relative sparsity of the network, suggesting further optimization might be warranted.

In Chapter 6, we propose an automatic entropy-based inner-loop optimization of CNN models, specifically the learning rate HyperParameter (HP). Our novel automatic optimization increases the rate of model convergence and delivers a more efficient exploration of the search space using the ME and SNR metrics. This chapter presents evidence to suggest that CNN models do not undertake a complete exploration of the solution space and thus indicates CNN models might not generalize well for datasets where the underlying data distributions do not represent a rich feature set, strengthening the findings presented in Chapter 5.

In Chapter 7, we propose a novel L_2 regularization term based on ME measures as a noise-robust technique to the commonly used Categorical Cross-Entropy (CCE) loss function to enhance model performance and mitigate overfitting. We hypothesize and verify that the CCE loss function is highly susceptible to structural noise in the input data. This Chapter addresses one of the main challenges faced through all of the previous optimizations proposed in the previous chapters (Chapters 3 - 6)

Finally, in Chapter 8, we conclude the thesis, we identify limitations in our experimental study and discuss potential areas for future contributions.

Chapter 2

Background

A summary list of notations adopted in this thesis is provided in Appendix A along with hyperlinks to access the source code.

2.1 Deep Learning

Most deep learning tasks involve mapping a function $f : \mathbf{X} \rightarrow Y$, where $\mathbf{X}$ is the input space and Y is the output space. In computer vision tasks like image classification, the input space $\mathbf{X}$ is complex, comprised of n number of d-dimensional input vectors i.e. digital images which can be represented in the form $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i = \langle x_i^1, x_i^2, \cdots, x_i^d \rangle | \mathbf{x}_i^j \in \mathbf{X}$ and $i, j \in \mathbb{Z}_{>0}$ (Maiorov, 2006). The output space Y is the probability interval from [0,1] for an input image vector $\mathbf{x}_i$ which classifies the input image. Simplified, $\mathbf{X}$ is the input dataset with n number of d-dimensional input images i.e. $\mathbf{x}_{1-n}^{1-d}$ with a corresponding class label $y_{1-n} \in Y$, i is the index referring to any individual image in $\mathbf{X} | i \in \mathbb{Z}_{>0}$. This thesis is constrained to supervised learning where each input vector $\mathbf{x}_i$ has an associated label y_i such that, $(\mathbf{x}_i, y_i) \in \mathbf{X} \times Y$ i.e. the input dataset consists of n number of images with associated class labels $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\}$.

In most cases specifying the function f linearly is improbable using conventional methods due to the high-dimensional interpolated characteristics of the input data space. It is relatively easy to obtain samples of data which can be annotated by humans with corresponding class labels enabling an approximation ($\hat{f}$) of the function f. Adopting the standard probabilistic assumption (Goodfellow, Bengio & Courville, 2016) that there exists an underlying data distribution $\mathcal{D}$ over the collection of labelled input data $\{\mathbf{X}, Y\}|\{\mathbf{x}_i, y_i\} \sim \mathcal{D} \ \forall i \in \mathbb{Z}_{>0}$, the approximated function, $\hat{f}$ would be the learned representation of the mapping $f : \mathbf{X} \rightarrow Y$. Furthermore, $\{\mathbf{x}_i, y_i\}$ are all identical and independently distributed data samples generated from a distribution D.

In an ideal scenario where the input space can be linearly separated, $\hat{f}$ would have learned the underlying representation of the mapping $f : \mathbf{X} \rightarrow Y$. Due to the complexity involved in learning the mapping of high-dimensional interpolated input data, the predicted class label $\hat{y_i}$ could be different to the ground truth class label y_i for an input image $\mathbf{x}_i|\{\mathbf{x}_i, y_i\} \in \{\mathbf{X} \times Y\}$. A loss function $L(\hat{Y}, Y)$ quantifies the average error in classification for the collection of n input samples.

The primary objective of deep learning is to minimize the loss function such that $\hat{f}$ is a close approximation to the underlying representation of the mapping $f : \mathbf{X} \rightarrow Y$. Once a close approximation is achieved, the training data samples can be disregarded and the learned function $\hat{f}$ can be used to predict mapping of new sample data. Note that the search space for f can possibly be only a subset of all possible classes of functions $\mathcal{F}$ that maps $\mathbf{X} \rightarrow Y|f \in \mathcal{F}$. In a very simple scenario, $\mathcal{F}$ could just be $y = mx + c$, an equation for a straight line. The approximation of $\hat{f}$ is over $\mathcal{F}$ and not f, therefore the general average loss function for deep learning can be given as Equation 2.1 (Karpathy, 2016).

$$\hat{f} \approx \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}_i), y_i) \tag{2.1}$$

2.1.1 Difficulty in Functional Approximation

Functional Regularization

Optimizing Equation 2.1 is theoretically non-linear and challenging; consider a simple solution where the function f maps $\mathbf{x}_i$ to its corresponding y_i and zero in all other instances, achieving a minimum regret in terms of the loss function L. While this solution might be acceptable, it is highly ungeneralizable and would produce a high loss for any sample data that is not represented by D. In other words, an additional parameter (regularization) is needed in Equation 2.1 to account for optimizing f regardless of the fact that some functions in $\mathcal{F}$ might not fit the training data $\mathcal{D}$. Regularization can be achieved by introducing a scalar-valued function $R(f)$, which can limit overfitting to the training set, Equation 2.1 can be rewritten using $R(f)$ as Equation 2.2.

$$\hat{f} \approx \underset{f \in \mathcal{F}}{min} \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}_i), y_i) + R(f) \tag{2.2}$$

The Bias-Variance Trade-off

Belkin, Hsu, Ma and Mandal (2019) present the Bias-Variance Trade-off for functional approximation. The authors assert that, functional approximation $\hat{f}$ for a function f mapping an input space $\mathbf{X}$ to an output space Y where $\mathbf{x}_i, y_i \in \mathbf{X}, Y \;\; \& i \in \mathbb{Z}_{>0}$ should be rich enough to express the underlying structural information in the data yet, be simple enough to avoid approximating spurious patterns. In other words, $\hat{f}$ should not underfit or overfit f for a given input $\mathbf{X}$. Further discussion around the Bias-Variance Trade-off from a neural network perspective is presented in Section 2.2.3.

The Role of Noise in Functional Approximation

The complexity of determining accurate approximations of $\hat{f}$ for f is further compounded by the fact that, if there is insufficient information on the structure or topology of $\mathbf{X}$

then, constructing accurate approximations of $\hat{f}$ fails for even binary representations of Y (Shwartz-Ziv & Tishby, 2017). This lack of approximation is due to invariance of information measures to invertible transformations such that, distinguishing low complexity classes from high complexity classes comes at a high cost (Moshkovitz & Tishby, 2017).

A simple solution to the aforementioned problem is to introduce a small amount of noise such that the function f can be approximated using stochastic methods. Discussions around noise and information measures are discussed throughout the thesis and more specifically in Chapters 3-4.

2.1.2 Neural Networks (NNs)

Evaluating $\hat{f}(\mathbf{X})$ outputs a predicted class label Y, which is a vector of all class posterior probabilities for each input $\mathbf{x}_i \in \mathbf{X}$. As f is non-linear due to the high-dimensional input space, closer approximations to f can be achieved by reducing the dimensionality of the input space such that it can be linearly separated. In lack of any a priori knowledge of the structure of $\mathbf{X}$, repeated multiplication of matrices and introducing element-wise non-linearities can help in linear separation (X. Zhang et al., 2019).

In Figure 2.1, for any given d-dimensional input image $\mathbf{x}_i \in \mathbf{X}$, the output $y_i \in Y$ is a vector of class posterior probabilities in the interval [0,1], i.e. $f(\mathbf{x}_i) = \sigma(\mathbf{W}\mathbf{x}_i)$. $\mathbf{W}$ is the weight matrix for the neural connections and σ is an element-wise non-linearity such as a Rectified Linear Unit (ReLU) which is of the form max(0,$\mathbf{x}_i$). Other non-linearities include sigmoid, which is of the form $1/1+e^{-\mathbf{x}_i}$ and tanh of the form, tanh = sinh($\mathbf{X}$) / cosh($\mathbf{X}$) = ($e^{\mathbf{X}}$ - $e^{-\mathbf{X}}$) / ($e^{\mathbf{X}}$ + $e^{-\mathbf{X}}$).

Utilizing additional hidden layers in a NN increases its representational/learning capacity and enables the network to expand on its exploration of the search space thus a narrower convergence in the solution space. A two-hidden layer NN would have an

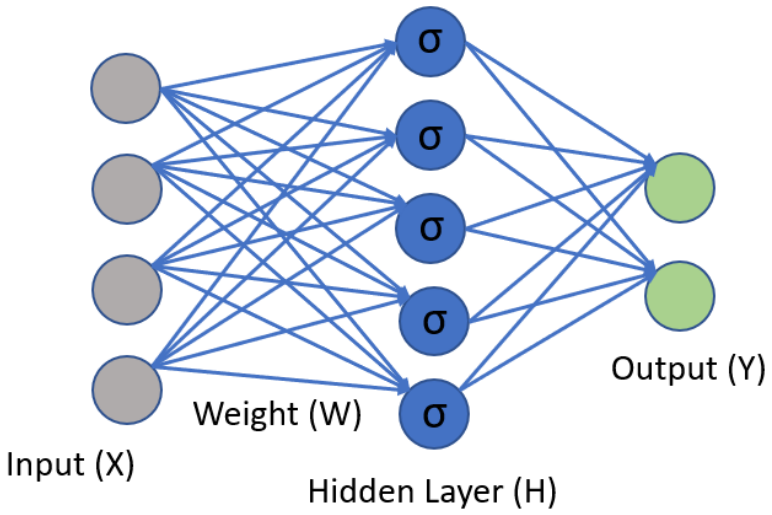

Figure 2.1: Illustration of a single hidden layer Neural Network (NN), adopted from (X. Zhang et al., 2019)

effective functional representation capacity as defined by $f(\mathbf{x}_i) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_i)$ and a three-hidden layer NN $f(\mathbf{x}_i) = \mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_i))|\mathbf{W}_i \in \mathbf{W}$ (layer weights).

2.1.3 Classification

Instead of assigning continuous scalar-values for each input; in a classification task, the values are discretized into a single class from all available classes. As an example, take the binary classification problem of classifying images of dogs and cats. The output of f will be a 2-dimensional vector for which the class posterior probabilities can be computed. The output of f is dependent on the values of K and χ, where K is the number of classes and χ is the number of neurons in a hidden layer of a NN. In other words, $\chi_i(h_i)$ is the number of neurons for a given hidden layer h_i where, i is the index for any hidden layer in the total number of hidden layers H where, $h_{i-H} \in H; \chi_i in \chi$.

A softmax function is commonly used to calculate the required probability distributions (Goodfellow et al., 2016). The softmax function takes a vector input α and outputs a vector of the same size β, where $\beta_i = e^{\alpha_i} / \sum_{j=1}^{K} e^{\alpha_j} | \alpha_i; \beta_i \in \alpha; \beta$. The output

vector β for the input vector α is normalized such that the logits are in the interval of [0,1] and sum to 1. In our binary classification example, say for an input image $\mathbf{x}_i$ the softmax outputs are ([0.8,0.2]) i.e. an 80% confidence that the input image contains a dog. Using a threshold function the softmax output can be reduced down to a one-hot encoded vector of y_i = [1,0].

2.1.4 Categorical Cross-Entropy (CCE) Loss

The most commonly used loss function is the Categorical Cross-Entropy (CCE) loss given in Equation (2.3), which is a measure of difference between the probability distributions of one-hot encoded CNN computed class labels and ground truths. CNN classification uses a softmax function to calculate the required probability distributions (Goodfellow et al., 2016).

$$E(p,q) = \sum_{i=1}^{n} = p(\mathbf{x}_i)\ log\ q(\mathbf{x}_i) \quad Where, \mathbf{x}_i \in \mathbf{X} \tag{2.3}$$

In Equation (2.3), $q(\mathbf{x}_i)$ and $p(\mathbf{x}_i)$ represent the probability distributions of the one-hot encoded CNN predicted class labels and ground truths respectively for an input data vector $\mathbf{x}_i$.

2.1.5 Summary

Supervised deep learning involves approximating the functional representation $\hat{f}$ of an underlying mapping function $f : \mathbf{X} \rightarrow Y$, where X is the input space and Y is the output space for a given collection of n, d-dimensional input vectors where, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. The search space for approximation of $\hat{f}$ is the problem space of $\mathcal{F}$, where each $f \in \mathcal{F} | \mathcal{F} : \mathbf{X} \rightarrow Y$ implying $\hat{f} : \mathbf{X} \rightarrow Y$. A loss function $L(\hat{y}, y)$ quantifies the error in the ground truth class labels Y and predicted class labels $\hat{Y} = \hat{f}(\mathbf{X})$. Closer approximations of f is dependent on the total number of classes K and the number of neurons d' in the

hidden layer h_i. The total number of hidden layers is denoted as H and each specific hidden layer is indexed using h_i i.e. the first hidden layer after the input layer in a NN

2.2 Optimization

The aim of any supervised deep learning task is to optimize a function $\hat{f}$ which is the functional representation obtained by exploring the *search space* for a *problem space* $\mathcal{F}$ that can accurately capture an underlying mapping of a function $f : \mathbf{X} \rightarrow Y$ for a given *input space* $\mathbf{X}$ with *output space* Y and *solution space* $\mathcal{Y}$. Assume that the optimization is of the form $\hat{\theta} = \min_{\theta} g(\theta)$.

2.2.1 Non-Convex Optimization

As discussed in Section 2.1, a linear separation of f is improbable using conventional methods due to the high-dimensional interpolated characteristics of the input data space. Closer approximations to f capturing the learning and classification problems accurately involves imposing structural constraints such as sparsity or low rank or assuming the objective is the optimization of a non-convex function. A convex optimization problem is where both the objective function and the constraint set are convex, if either are non-convex they can be categorized as non-convex. Formal definitions for both can be found in (P. Jain & Kar, 2017).

Simply put, in a non-convex optimization problem, there are multiple local optima (minima for loss minimization functions and maxima for generalization functions) and the optimization of non-convex functions can converge to any local optima. Changing the constraint set or enhancing the learning of the functional representations can yield convergence to a global optima. Image classification is an example of a non-convex optimization problem. Optimizing the non-convex problem can be achieved using a stochastic trial-and-error method checking a range of θ values and selecting the one

that greatly minimizes the function $g(\theta)$. Due to the large search space involved, this method will prove ineffective and impractical.

2.2.2 First Order Derivative Optimization

Constraining the optimization of g to only differentiable functions allows us to compute gradient ∇_θ for g using a well known back-propagation algorithm, presented as Equation 2.4 (Van Ooyen & Nienhuis, 1992). The gradient is a vector constructed using the partial derivatives for the first order approximation of g to judge the direction of slopes along all the dimensions of θ. A gradient search using a Stochastic Gradient Descent (SGD), presented as Equation 2.5 (Bottou, 1991) can be employed to adjust the weight parameters $\mathbf{W}_1, \mathbf{W}_2, \cdots \mathbf{W}_H$, where H is the number of hidden layers (depth) of the neural network using back-propagation to get closer approximations of g. The weight adjustments are calculated for each neuron $\omega_1, \omega_2, \cdots, \omega_\chi | \chi << d \in \mathbb{Z}_{>0}$ and $\omega_i \in \mathbf{W} | i \in [1 - \chi] \& \mathbf{W} \in [1 - H]$, where χ is the number of neurons in a given hidden layer.

Full theoretical proofs for non-convex optimization including first and second order derivative optimization are provided by Soltanolkotabi, Javanmard and Lee (2018).

Back-Propagation (BP)

Back-propagation is the recursive application of the chain-rule from calculus, gradients of θ for the function g ($\nabla_\theta g$) are computed for each instance of $\{\mathbf{x}_i, y_i\} \in \{\mathbf{X}, Y\}$ i.e. the n number of samples in the training dataset using the parameter vector θ. Since the problem at hand is a function approximation in high-dimensional vector spaces, partial derivatives are most appropriate.

Assume that the NN needs to approximate a simple linear function $y = mx + c$, in this instance $\partial y / \partial x$ would yield a method to optimize the global optima; ∂x would

be the input data vector and network parameters, while ∂y would be the total loss. Intermediate functions (local optima) can be calculated using the chain rule, $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}$. Intermediate functions help the NN to reduce the dimensionality for the search space of $\mathbf{x}_i$ by calculating the intermediate matrices which can be multiplied using dot products to yield a final gradient ∂y_i and adjust the parameter vector θ for a given weight parameter vector $\mathbf{W}$ to minimize the error. The simplified weight update rule for a single hidden layer NN using back-propagation is given as Equation 2.4 (Van Ooyen & Nienhuis, 1992).

$$\Delta\omega_i = -\eta\frac{\partial \hat{f}'}{\partial \omega_i} \tag{2.4}$$

Where, $\omega_i \in \mathbf{W}$ is the old weight, $\Delta\omega_i$ is the new weight, η is the learning rate which influences the magnitude of weight updates and $\hat{f}'$ is the error function being optimized.

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) can be used to optimize a NN model for any set of differentiable functions f that maps an input vector $\mathbf{X}$ to a predicted output $\hat{Y}$. SGD can be considered as an inner-loop for optimization whereas optimizing other network parameters, referred to as HyperParamters (HPs) such as learning rate, depth of the NN, input data vector size used for training and so on, can be thought of as outer-loop for optimization. The permutation and combination of HPs order in the millions and optimization of these HPs are critical in establishing model performance (Bebis & Georgiopoulos, 1994). The equation for NN parameter weight updates is given as Equation 2.5 (Bottou, 1991).

$$\omega_{t+1} = \omega_t - \eta_t \nabla_\omega J(\mathcal{Z}, \omega_t) \tag{2.5}$$

Where, $J(\mathcal{Z}, \omega_t)$ is the cost function being optimized with $\mathcal{Z}$ computed using some

fixed function for a given $\{\mathbf{x}_i, y_i\}$ pair, say $\mathcal{Z} = x_i \bullet y_i$. ∇_ω is the vector of weights for the NN, ω_{t+1} is the updated weight matrix for the old weight matrix $\omega_t \in W$ at a given instance of time t.

2.2.3 L1 and L2 Regularization for Neural Networks

The intuition behind regularization is that of Ockham's razor to promote simpler functional representations by penalizing overly complex functions during model training. Unlike empirical risk minimization where loss minimization is the only consideration, regularization minimizes structural risk by considering both model complexity and loss optimization. In other words, simpler and prominent functions which contribute to loss minimization are preferred and selected (Bilgic et al., 2014). Model complexity can be considered in two ways, as a function of the total number of non-zero feature weights (L_1) or as a function of all the feature weights in a model (L_2). The most commonly used regularization method for computer vision tasks is L_2 regularization. Model complexity for a L_2 regularization is computed using Equation (2.6). The L_2 regularization term can be defined as the sum of squares of all the feature weights (Cortes, Mohri & Rostamizadeh, 2012).

$$||\omega||^2 = \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_\chi^2 \tag{2.6}$$

In Equation (2.6), the absolute magnitude of the feature weights ω quantifies model complexity. Feature weights approaching zero have little significance in determining model complexity in contrast to large outlier weights which have a more pronounced significance. The size of the feature weight matrix $H \times K \times \chi$ representing the number of trainable model parameters contribute significantly towards determining model complexity.

The Bias-Variance Trade-off revisited from a neural network perspective

As previously discussed in Section 2.1.1, functional approximation should neither overfit or underfit to the input data. Neural Network (NN) models are typically trained to interpolate the input data. Thus in theory, the models would severely overfit and as such should fail when unseen data (i.e. generalization) is presented (Belkin et al., 2019). However, contrary to theory, neural network models in practice provide impressive classification accuracy.

Achieving a good balance between empirical/training risk and true/test risk is dependent on model learning capacity. Usually NN models are over-parameterized to capture rich feature information and then subsequent techniques such as normalization and regularization of weight matrices are conducted to achieve high accuracy. The approach of utilizing severely over-parameterized models might work well in certain instances but, comes with several risks and sub-optimal performance. Mitigating severe over-parameterization is explored in depth in Chapter 5.

2.2.4 Batch Normalization (BN)

Batch normalization was introduced to reduce internal covariate shift (Ioffe & Szegedy, 2015). As discussed in Section 2.2.3, over-contribution of large outlier weights reduces stability of NN models and increases model learning saturation. BN normalizes model weight distributions across a hidden layer for a given mini-batch input, consisting of multiple input data vectors rather than a single input data vector. The mini-batch size is a HP consisting of a subset of the total n number of input training images. Equation 2.7 is used to implement BN for hidden layers, which can be understood as, $\hat{\mathbf{x}} = Norm(\mathbf{x}_i, \mathbf{X})$ where, $\mathbf{x}_i$ is a single input data vector for the mini-batch selected from $\mathbf{X}|\mathbf{x}_i \in \mathbf{X}$.

$$\hat{\mathbf{x}}^k = \frac{\mathbf{x}^k - \hat{f}'[\mathbf{x}^k]}{\sqrt{Var[\mathbf{x}^k]}} \qquad Where,\ \hat{f}'[\mathbf{x}^k] = \frac{1}{n}\Sigma_{i=1}^{n}\mathbf{x}_i. \tag{2.7}$$

Equation 2.7 applies BN for a given hidden layer with a d-dimensional input $\mathbf{x}^d = \langle x^1, \cdots, x^d \rangle \in \mathbb{R}$, each scalar feature is normalized by setting the mean to zero and variance to one for a given input $\mathbf{x}_i$. Recursive application of Equation 2.7 for all inputs of the mini-batch selected from $\mathbf{X}$ where $\mathbf{x}_i \in \mathbf{X}$ offers reduced overfitting, increased learning rates and limited regularization.

Overfitting occurs when the network parameters are precisely tuned to the unique variances of the input training data that the model fails to generalize for new unseen data (Hawkins, 2004), essentially memorizing the input training data (C. Zhang, Bengio, Hardt, Recht & Vinyals, 2016). BN requires large mini-batch sizes of 64 or 128 to be effective, constraining a full exploration of the search space due to memory and computational constraints for a NN model. Furthermore, using BN does not eliminate or account for a NN model's susceptibility to noise introduced during feature computations or determination of a class label.

2.2.5 Summary

In this section, we explored the supervised image classification problem in the context of non-convex optimization problems. Due to the high-dimensional interpolated characteristic nature of the input space for image data, structural constraints such as sparsity need to be accounted in order to achieve closer functional approximations of the underlying mapping function f. The vastness of the search space relative to the input space might result in a convergence to a suboptimal local minima/maxima. Utilizing traditional optimization methods such as linear programming or derivative-free optimizations are impractical due to the higher-order search space involved. Non-traditional techniques such as using first order derivatives to compute gradients enables the problem to be reformulated into a hill-climbing, more specifically gradient descent problem.

Exploiting the non-linear characteristics of NNs, the underlying mapping function

f can be approximated by minimizing the error of a cost function using algorithms such Stochastic Gradient Descent (SGD) with Back-Propagation (BP) to adjust the NN parameter space to achieve closer approximations to f. As the problem is non-convex in nature, the NN might optimize the cost function to any of the numerous local optima leading to overfitting to the training data causing performance degradation on new unseen data sampled from the same distribution $\mathcal{D}$. Regularization and Normalization methods can help alleviate the problem of overfitting and suboptimal convergence to a local optima but cannot eliminate the issue.

2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are the state-of-the-art for complex computer vision tasks such as image classification (Szegedy et al., 2015) and localization (Krizhevsky, Sutskever & Hinton, 2012). CNNs are primarily used to learn functional representations from data with underlying interpolated spatial structures using a input data vector $\mathbf{x}_i$ also known as a tensor. The CIFAR-10 dataset (Krizhevsky & Hinton, 2009) consists of a $32 \times 32 \times 3$ (RGB) tensor color images with three color channels, Red (R), Green (G) and Blue (B) color channels sampled from ten natural image classes such as cars, cats, dogs and so on. Whereas, handwritten black and white images in the MNIST dataset consist a $28 \times 28 \times 1$ (grayscale) tensor belonging to the ten number (0-9) classes. Intuitively, classification of natural images requires the exploration of a wider search space relative to handwritten character recognition. Implementing a fully-connected NN as illustrated in Figure 2.1 even with multiple hidden layers for such high-dimensional data will invariably yield suboptimal classification performance.

The goal of any CNN as illustrated in Figure 2.2 is to reduce the higher d-dimensional input vector $\mathbf{x}_i$ into a lower-dimensional feature vector, say $\phi(\mathbf{x}_i) = \langle \phi_j(\mathbf{x}^1), \cdots, \phi_{\chi'}(\mathbf{x}^d) \rangle$, where $i, j, d, \chi' \in \mathbb{Z}_{>0} | \mathbf{x}_i^j \in \mathbf{X}$ can be linearly separated using the CNN feature weight

vector $\mathbf{W} = \langle \omega^{j}, \cdots \omega^{\chi'} \rangle$ (Mallat, 2016). In other words, a CNN weight vector $\mathbf{W}$ is constructed using χ' convolutional kernels/filters/channels each with a weight $\omega_{j-\chi'}$ in order to further construct a lower-dimensional feature map vector $\varphi(\mathbf{x}_i)|\varphi(\mathbf{x}_i) \in \varphi(\mathbf{X})$ can be linearly separated using $\mathbf{W}$ for an input data vector $\mathbf{X}$ formed from n number of training images.

A linear separation of $f(\mathbf{X})$ can be accomplished by using $\phi(\mathbf{x}_i) = \Sigma_{j=1}^{\chi'} \varphi_j(\mathbf{x}) \in \mathbb{R}$, where the computed feature map collection $\mathcal{D}$, such that, $|\mathcal{D}| = \chi'$ is much larger than the d-dimensional input vector $\mathbf{x}_i$ where, $|\mathbf{x}_i| = d$ i.e. $1 \leq d \ll \chi'$. In other words, the underlying mapping function can be linearly separated if the feature collection is sufficiently reduced in dimensions and a rich feature collection $\mathcal{D}$ is constructed using the χ' convolutional kernels/filters/channels such that the n number of images sampled from the distribution of $\mathcal{D}$ correctly map the input space $\mathbf{X}$ to the output space Y.

Assuming a constant $f(\mathbf{X})$, evaluation of $\hat{f}(\mathbf{X})$ is given by Equation 2.8,

$$\hat{f}(\mathbf{X}) = \sum_{i=1}^{n} \langle \phi(\mathbf{x}_i), \mathbf{W} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{\chi'} \langle \varphi(\mathbf{x}_i), \omega_j \rangle \;\; i, j \in \mathbb{Z}_{>0}. \tag{2.8}$$

In Figure 2.2, the convolutional layers $\mathbf{C}_{1-H'} \in \mathcal{C}$ generating lower-dimensional feature maps φ is the essence of a CNN i.e. the number of convolutional layers H' is a subset of the total number of hidden layers H which produces $\mathbf{C}_i|\mathbf{C}_i \in \mathcal{C}$ feature maps from $\chi'_{i-H''}(\mathbf{C}_{i-H'})|\chi'_{i-H''} \in \chi, \mathbf{C}_i \in \mathcal{C}$ convolutional units/kernels/channels. In other words, the first convolutional layer $\mathbf{C}_1$ with χ'_1 number of convolutional kernels produces D_1 collection of feature maps such that, $\mathbf{C}_{1-H'} \in \mathcal{C}$, where H' is the total number of convolutional layers with $\chi'_{i-H''}(\mathbf{C}_{i-H'})$ number of convolutional kernels producing $D_{1-\chi'_{i-H''}}|D_{1-\chi'_{i-H''}} \in \mathcal{D}, \chi'_{i-H''} \in \chi$, where H'' is the number of convolutional kernels for a given convolutional layer.

The convolutional layer takes a tensor input and produces a lower dimensional tensor output by convolving the input with a collection of much smaller kernels/filters/channels.

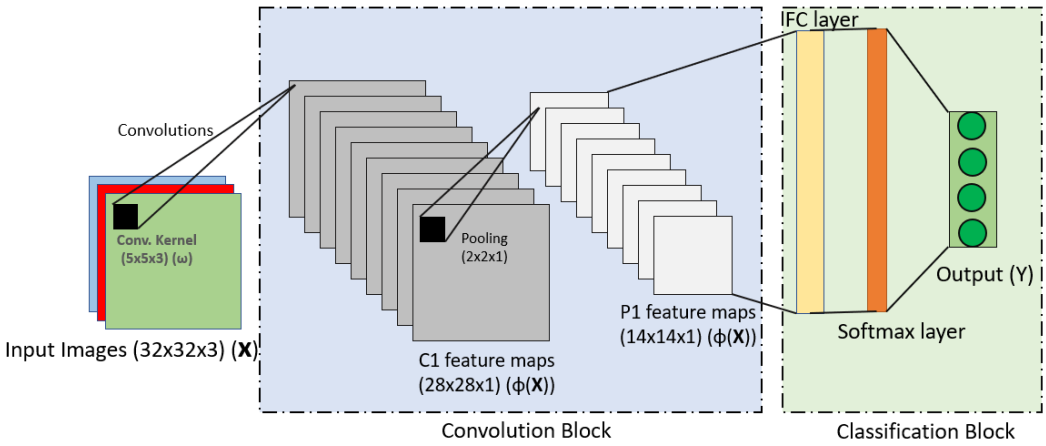

Figure 2.2: Illustration of a (5×5×3) convolving kernel/filter/channel over an input data vector $\mathbf{x}_i$ (32×32×3) and stride 1 with no padding and a (2×2×1) pooling kernel/filter/channel, yielding a weight vector $\mathbf{W}$ and a class output y_i, adopted from (LeCun et al., 2010).

Consider the 5×5×3 convolving kernel/filter/channel, $\omega_{i,j}|\omega_{i,j} \in \mathbf{W}$ where $\omega_{1,1}$ is the weight for (i.e. the weight of the convolving kernel) which has a parameter space of 75 (5*5*3) representing its learning capacity. ω slides with the specified stride size in pixels across the whole input tensor and computes a dot product at each position producing an activation map characterizing the features of the input tensor $\mathbf{x}_i$ yielding a $\mathbf{c}_1$ feature map, subsequent convolutional layers will $\mathbf{c}_2, \mathbf{c}_3, \cdots, \mathbf{c}_{\chi'}$ feature maps for a network depth of H and convolutional depth of H'.

The pooling layers, $p_{1-(H'-1)} \in \mathcal{P}$ where p_i is the number of pooling layers for a total of $H' - 1$ convolutional layers. Pooling layers simply spatially reduce the size of feature maps to increase computational efficiency. Pooling layers have fixed downsampling transformations (no trainable parameters, commonly a MAX operation) in the figure using a 2×2 pooling kernel/filter/channel, a downsampling ratio of 2 is achieved. Optimizing/training the weight vector $\mathbf{W} = \{\omega_1, \omega_2, \cdots, \omega_{\chi'}\}$ using back-propagation for all χ' convolutional kernels/filters/channels facilitates localized spatial features to be learned from the input tensor. Finally, a Fully-Connected (FC) NN with

single or multiple hidden layers as illustrated in Figure 2.1 is utilized in conjunction with a softmax function to calculate the class posterior probabilities.

Theoretically, deeper CNNs have an increased capacity to compute a much closer approximation of $\varphi(\mathbf{X})$ compared to shallower CNNs with no other variances in HP configurations. The drawbacks of using very deep CNNs is discussed in later chapters of this thesis. The enumeration of the feature map collection D exponentially grows for a given function $f(\mathbf{X})$. A richer and lower-dimensional feature map collection D can also be achieved using a shallower yet broader model since they are functionally equivalent to a deeper yet narrower model in terms of generating a larger feature map collection D which is explored in later chapters (Chapters 4 - 5).

Advantages of CNNs

CNNs are primarily feature extractors, unlike traditional Deep NNs (DNNs) which specialize in function mapping as explained in Section 2.1. This key distinction allows CNNs to outperform other types of DNNs on high-order non-linear spatially interpolated data. This is because, DNNs are highly susceptible to subtle spatial variations in the input data leading to poor non-linear maps (Bebis & Georgiopoulos, 1994). In other words, the resiliency of CNNs arises due to their decreased sensitivity to noise, feature information is preferred over simple direct mapping between the two spaces and feature abstraction can be increased through localized spatial detection and optimization (LeCun, Huang & Bottou, 2004). Abstraction in CNNs can be forced since convolutional layers have sparse structural connectivity and employ a shared weight architecture. Increasing network depth subsequently increases feature extraction and inturn results in specialized feature maps optimized for the given input data vector (Krizhevsky & Hinton, 2010).

These specialized feature maps are primarily characterized by a limited degree of feature variance consisting of a small set of features contributing significantly to

classification performance neglecting any subtle changes in the input data (Krizhevsky & Hinton, 2009). Unlike DNNs which need to account for a vast range of variations for accurate function mapping (Hartigan & Wong, 1979), the process of feature extraction eases these constraints.

Drawbacks of CNNs

CNNs are computationally expensive for less demanding tasks like linear regression or clustering (Ardakani et al., 2018) in these instances, DNNs are much more efficient (LeCun et al., 2015; D. J. Becker et al., 1995). Furthermore, in problems where little domain knowledge is required using deep CNNs unnecessarily increase the computational requirements and exposes the model to significant training afflictions like overfitting (Hawkins, 2004). Due to the large number of HPs involved in CNNs training deep CNNs is both time consuming and difficult (Simonyan & Zisserman, 2014; Glorot & Bengio, 2010).

2.3.1 Convolutional Neural Network Architectures

VGGNet

The VGGNet architecture developed by the University of Oxford's Visual Geometry Group (VGG) (Simonyan & Zisserman, 2014) is similar to a traditional CNN illustrated in Figure 2.2 but with multiple stacked convolutional layers. VGGNet primarily focuses on investigating the effects of network depth on classification accuracy with the standard error minimization function computed using the combination of feature and weight vectors $\langle \phi(\mathbf{X}), \mathbf{W} \rangle$. The VGG-16 model has fourteen convolutional layers, with intermediate pooling layers to reduce computational complexity, and two final fully connected classification layers.

The stacked consecutive convolutional layers in the VGGNet architecture are designed to enable feature hierarchy and increase the effective receptive field in a convolutional block without significant trade-offs in computational performance. The major limitation of the VGGNet architecture is that it suffers from training afflictions such as exploding/vanishing gradients (Caruana, Lawrence & Giles, 2001) during back-propagation where the differentiable parameters do not propagate back to the initial layers. Another challenge of VGGNets which affect information propagation is its limited architectural and spatial constraints which become significant and affect classification performance, especially in complex datasets, where strict feature hierarchy might impede convergence.

Residual Network (ResNet)

The Residual Network (ResNet) architecture was first proposed by (K. He et al., 2016) utilizing skip connections between convolutional layers as a way to solve the degradation/saturation problem where the back-propagation mechanism becomes ineffective for greater network depths and strict architectural constraints, a problem that is persistent in any deep NN. The ResNet architecture is an arrangement of residual learning blocks, where multiple convolutional layers are stacked to enhance feature extraction effectiveness. Although stacking layers and adopting skip connections or shortcut paths is not new and has been researched since the introduction of Multi-Layer Perceptrons (MLPs) (Ripley, 2007), initial implementations offered little improvements and suffer from the problem of vanishing/exploding gradients during back-propagation as discussed and witnessed in the VGGNet architecture. ResNet models obey the $\hat{f}(\mathbf{X}) = \langle \phi(\mathbf{x}_i + \mathbf{x}), \omega_i \rangle$, where $i \in \mathbb{Z}_{>0}$ function approximation for a single input data vector, where the identity mappings are of the form $f(\mathbf{x}_i) = \mathbf{x}_i$ (K. He et al., 2016). The general form for ResNets is presented as Equation 2.9.

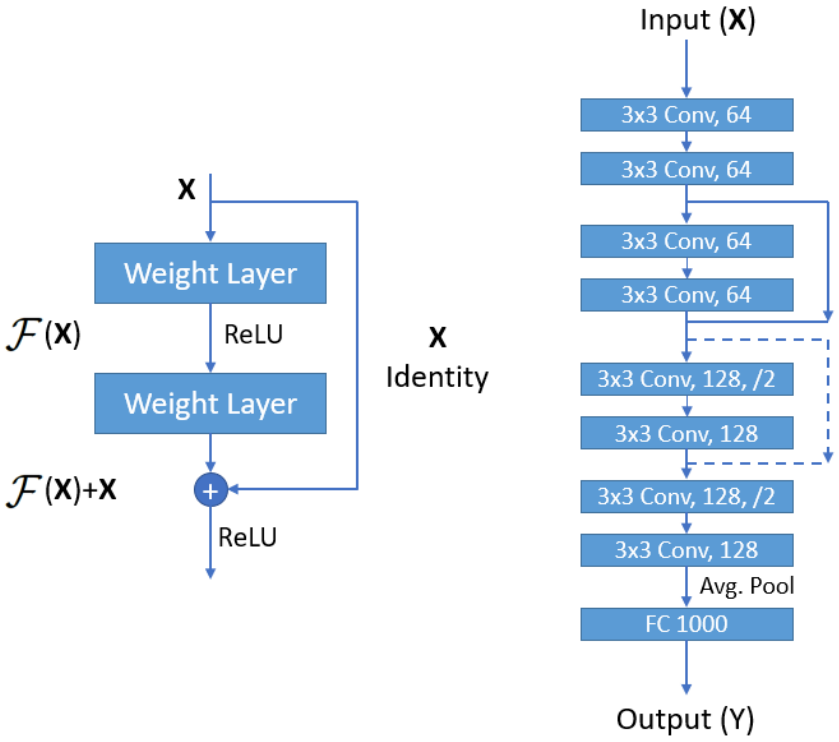

Figure 2.3: Example of a ResNet architecture. **Left:** A residual block. **Right:** A Residual network with 8 convolutional layers. The dashed lines represent an increase in dimensions, while the pooling operations are indicated by (/2), adopted from (K. He et al., 2016).

$$y_l = h(\mathbf{x}_i) + \mathcal{F}(\mathbf{x}_i, \mathbf{W}_l) \quad \& \quad f(y_l) = \mathbf{x}_{l+1} \qquad l \in H' \in \mathbb{Z}_{>0} \tag{2.9}$$

Aggregated Residual Transformations (ResNeXt)

A neuron can be thought of as an aggregation of signal transformations along all the kernel/filter/channel inputs ζ given as $\sum_{i=1}^{\zeta} \omega_i \mathbf{x}_i$, where $i \in \mathbb{Z}_1^+$. The working principle of aggregated residual transformations called ResNeXt proposed by (Xie et al., 2017) is to replace the computation of ω_i with the transformation of a lower-dimensional projection of $\mathbf{x}_i$ as $\tau(\mathbf{x}_i)$. ResNeXt models can be represented using the $\hat{f}(\mathbf{X}) = \langle \sum_{i=1}^{\zeta} \phi(\mathbf{X}), \mathbf{W} \rangle$ error minimization function, where ζ (cardinality) is the set size of aggregated transformations. Cardinality is an important HyperParameter affecting model capacity similar to network depth. The error minimization function for ResNext given in Equation 2.10.

$$\hat{f}(\mathbf{X}) = \langle \sum_{i=1}^{\zeta} \tau(\mathbf{X}_n^{\zeta}), \mathbf{W} \rangle \quad n \in \mathbb{Z}_{>0}, \zeta \in \mathbb{R} \tag{2.10}$$

where ζ (cardinality) is the set size of aggregated transformations. ResNeXt in practice, implements a split-transform-merge paradigm where a pointwise grouped convolutional layer splits the input into sets of feature maps and performs convolution operations (transform step) on these sets independently with the outputs being merged or concatenated depthwise with a final bottleneck convolutional layer (1×1) producing the output.

Densely Connected Convolutional Network (DenseNet)

Densely Connected Convolutional Networks (DenseNets or DNs) proposed by (Huang et al., 2017) and illustrated in Figure 2.4 were envisioned from a simple concept that the input of final feature vector with a layer depth H' should include the concatenation of all

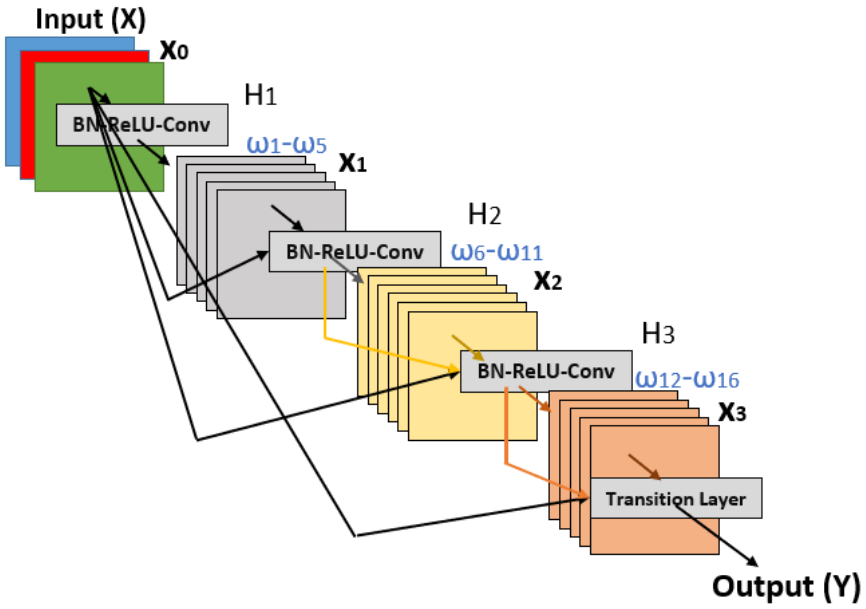

Figure 2.4: Example of a DenseNet architecture, adopted from (Huang et al., 2017).

preceding feature maps produced from H'-1 layers rather than just the feature maps from the H'-th layer. The distinction of DN from ResNet and ResNext CNN architectures is that the feature maps generated from the first convolutional layers contribute equally to the creation of the final feature vector ϕ, instead of the first layers being abstracted into deeper layer feature maps. Although this distinction limits the depth of DN models due to practical considerations like memory constraints, DN models outperform ResNets on several benchmarking datasets, discussed in later chapters of this thesis. DN models obey the $\hat{f}(\mathbf{x}_i) = \langle \phi(\mathbf{x}_i), \mathbf{W}_i \rangle = \langle \varphi([x_0, x_1, ...x_l]), \mathbf{W}_1 \rangle$ error minimization function for a single input data vector, a more general form is presented as Equation 2.11. DNs also requires the computation of a lower dimensional vector representation $\phi(\mathbf{X})$ from the input data $\mathbf{X}$. Growth rate is an important HP in DNs, which influences the contribution of an individual convolutional layer to the final feature vector $\phi(\mathbf{X})$.

$$\hat{f}(\mathbf{X}) = \langle \phi([\mathbf{X}_i^1, \cdots, \mathbf{X}_n^{l-1}]), \mathbf{W} \rangle \quad i, l \in \mathbb{Z}_{>0} \tag{2.11}$$

2.3.2 Summary

In this section, the working principle of CNNs were investigated where the creation of lower-dimensional feature maps is the primary distinction from traditional DNNs. The convolution operation in CNNs can resolve spatial variances in the input data vector more effectively however, a CNNs ability to create an adequate feature vector φ relies upon the fine-tuning of HPs such as stride size, kernel/filter/channel size, number of convolutional kernels/filters/channels along with similar HPs as traditional HPs such as network depth and batch size. A major factor explaining the state-of-the-art classification performance of CNNs in image classification problems is due to convolution operation being able to accurately reduce a high-order non-linear spatially interpolated data into lower-dimensional feature maps while preserving spatial topological integrity.

Consistent with traditional DNNs a deep CNN with no skip connections suffer from the same training afflictions like exploding/vanishing gradients during back-propagation where the differentiable parameters do not propagate back to the initial layers causing suboptimal convergences. The distinction of residual network CNN architectures using skip connections compared to a simple stacked VGGNet architecture helps alleviate the problem of exploding/vanishing gradients but ensure that the feature maps generated from the initial convolutional layers in a CNN are abstracted into later convolutional feature maps.

The aggregated residual transformation architecture known as ResNeXt aimed to preserve initial feature map information through the use of a split-transform-merge paradigm on the signal transformations along the kernel/filter/channel inputs. A stricter constraint was imposed by the DenseNet CNN architecture where the feature maps are concatenated before transitioning to the next convolutional layer.

Chapter 3

Examining Feature Extraction in CNNs

In the previous chapter (Chapter 2), in Section 2.3, we explored the working principle of CNNs and the convolution operation which creates richer internal feature representations compared to a traditional DNN. A simple CNN is illustrated in Figure 2.2. Section 2.3.1 explored how stacking multiple consecutive convolutional layers within a single block and stacking these convolution blocks increase the feature representational capacities of CNNs. In this chapter we quantitatively examine the effectiveness of consecutive layer stacking in both the convolution and classification blocks. We analyze the effectiveness of varied neural breadth configurations specifically in the classification block of simple CNN models. In the next chapter (Chapter 4), we investigate the nature of semantic feature extraction and abstraction for more complex residual architectures such as ResNet, ResNeXt and DenseNets, explained previously in Section 2.3. Furthermore, we investigate the nature of convolutional kernel saturation and its correlation to information overflow and underflow.

This chapter is published in the 46th Annual Conference of the IEEE Industrial Electronics Society, IECON-2020; © 2020, IEEE.

3.1 Introduction

As stated earlier, CNNs are the state-of-the-art for image classification tasks verified on standard benchmarking image datasets such as MNIST (LeCun, Bottou, Bengio & Haffner, 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009). Achieving these state-of-the-art performance is predicated upon complex fine-tuning of the HyperParameters (HPs), such as network depth and breadth in both the classification and convolution blocks of a CNN with improper HP configurations leading to suboptimal classification performance (Coates, Ng & Lee, 2011). In this chapter, we quantitatively measure, examine and present evidence to support the use of Signal-to-Noise Ratio (SNR) and Maximum Entropy (ME) measures in understanding information propagation within CNNs.

Our approach is based on the understanding that Maximum Entropy (ME) measures can be used to accurately quantify the total amount of signal information in a given series of data (Gull & Skilling, 1984). The degree of corruption of the signal i.e. the usable component can be estimated by calculating the signal-to-noise ratio (SNR) (Gonzalez & Woods, 2007) which is commonly used in digital image processing to enhance and restore degraded images (Krbcova & Kukal, 2017; Petrovici et al., 2018). Through our experiments and analyses we show that the use of these measures to tailor CNN model HPs, specifically network depth, breadth and configuration yields statistically significant and improved classification performance in a simple CNN model.

3.2 Background

3.2.1 Maximum Entropy of Image Data

Entropy measures are widely used for digital image enhancement such as de-noising and image restoration/reconstruction using de-convolutions (Petrovici et al., 2018; Gull

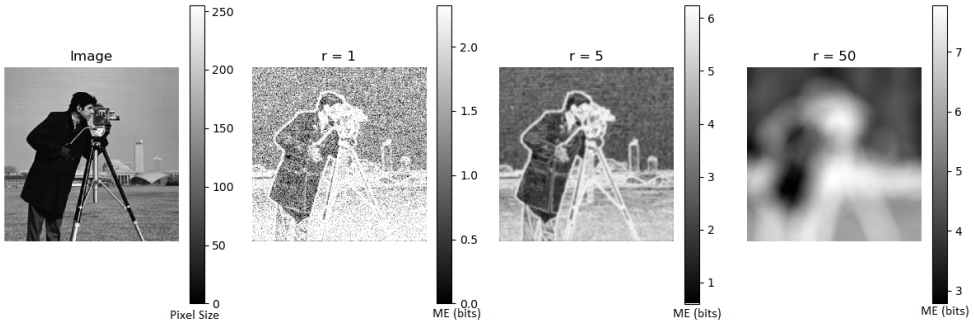

Figure 3.1: Difference in ME measurement for changes in the radius r used to measure the complexity contained in a local neighborhood.

& Skilling, 1984; Krbcova & Kukal, 2017). A method to approximate ME for digital images is through the use of distributed normalized histograms (Gonzalez & Woods, 2007; A. K. Jain, 1989). The open-source SciKit-image processing library written in Python can be used to calculate the ME measures for images (Virtanen et al., 2020).

Entropy in images is related to the complexity contained in a given neighborhood, computed by using a circular disk with a radius of r. The disk is used to measure minute variations in local grayscale level distribution. The maximum entropy for an image depends on the number of gray levels, an 8 bit image has 256 gray levels (0-255) which has a theoretical maximum entropy of $log_2(2^8) = 8$ bits per pixel. Changing the value of r can invariably produce higher or lower ME measure as illustrated in Figure 3.1. Similarly higher or lower ME values will be obtained while measuring convolutional kernel weights. A decrease in ME divergence can be observed in Figure 3.1 for r values of 5 and 50 relative to r values of 1 and 5. A significant difference in spatial/semantic information in the images can be seen with greater r values, which suggests a loss in precision during approximation.

ME measures for color images require the computation on each of the three color channels, Red (R), Green (G) and Blue (B) i.e. RGB separately and averaging the results. The averaged ME measures for images in the *MNIST* and *CIFAR-10* datasets are *1.04* and *2.2* bits per pixel and *2.41* and *5.28* respectively for r values of 1 and 5 respectively.

The amount of time taken to calculate the ME measures is insignificant as the ME calculation script can be executed in parallel on the CPU, while CNN model training occurs on the GPU. Solutions other than ME for image reproduction/reconstruction from noisy or incomplete measurements such as, the use of non-linear variations on fourier transformations fail when convolutional kernels are incorporated (Donoho et al., 1990), which is implemented for CNN optimization in later chapters of this thesis. Furthermore, ME reconstruction has been shown to provide superior noise suppression while mostly preserving de-emphasized structural noise near the baseline (relative to high signal information) (Donoho et al., 1990).

3.2.2 Signal and Noise Ratio of Image Data

Accurate quantifiable estimation of image quality regardless of variances in lighting conditions, depth of field and noise to name a few properties play an important role in processing digital images. While numerous methods exist to calculate digital image quality such as Mean Square Error (MSE), Root Mean Square Error (RMSE), Signal-to-Noise (SNR) and Peak Signal-to-Noise (PSNR), there are no perceivable distinct advantages between these methods for applications incorporating the Human Visual System (HVS), the biological inspiration for CNNs (Wang & Bovik, 2002).

Due to the stochastic distribution of image data in the MNIST and CIFAR-10 datasets, PSNR, MSE or RMSE values will yield imprecise measurements. Therefore, the conventional SNR calculations are employed to measure image quality. According to authors in (Gonzalez & Woods, 2007), SNR is a relative measure of the intensities of the usable signal to the environmental noise.

To be specific, SNR with respect to feature information represents the probability that unique and effective features can be extracted relative to noise. In other words, a higher SNR ensures that there is a greater chance to extract effective features which

should help mitigate overfitting to the input dataset.

In digital images, SNR is defined as the ratio of the average signal intensity to the standard deviation in the noise (Welvaert & Rosseel, 2013), given by equation 3.1.

$$SNR = \mu(\overline{S})/\sigma_{noise} \tag{3.1}$$

Where, SNR is the signal-to-noise ratio (unit-less), $\mu(\overline{S})$ is the mean of signal data and σ_{noise} is the standard deviation of the signal data with respect to the random noise.

The equation for calculating the mean of signal data given in equation 3.2,

$$\mu(\overline{S}) = \sum_{i=1}^{n} S/n \quad \forall S \in (0-255). \tag{3.2}$$

Where, $\mu(\overline{S})$ is the mean of signal data, when the pixel values for S is in between 0-255 for MNIST and 0-255 for red, green and blue color channels for CIFAR-10 and n is the total number of pixels.

The equation for calculating standard deviation of signal data with respect to the noise is given by,

$$\sigma_{noise} = \sqrt{(1/n)(\sum_{i=1}^{n}(x_i - \mu(\overline{S}))^2)} \quad \forall S \in (0-255). \tag{3.3}$$

Where, σ_{noise} is the standard deviation of the data, i.e. the signal data with respect to the noise, $\mu(\overline{S})$ is the mean of signal data calculated using equation 3.2, n is the total number of pixels in the data, $\mathbf{x}_i$ is the value of i^{th} pixel in the image.

Using Equation 3.1, we can calculate SNR for the MNIST and CIFAR-10 datasets with the mean of signal data calculated using Equation 3.2 and standard deviation computed using Equation 3.3.

As an example, lets assume a simple 3×3 grayscale image consists of an array of

pixel values,

$$\begin{bmatrix} 255 & 128 & 238 \\ 0 & 116 & 145 \\ 98 & 128 & 56 \end{bmatrix}$$

The above 2d array is converted to a 1d array using row-wise serialization as [255,128,238,0,116,145,98,128,56]. Inbuilt functions in the Python 3.7 environment computes a mean of 129.33 and a standard deviation of 75.36.

Using Equation 3.2, the mean can be computed as the sum of all elements in the 1d array divided by 9 (i.e. number of pixels). Using Equation 3.3, the standard deviation can be calculated as the square root of the average of squared deviations from the mean, explained as follows.

We compute the squared deviations from the mean as, $(255 - 129.33)^2$ + $(128 - 129.33)^2$ + $(238 - 129.33)^2$ + $(0 - 129.33)^2$ + $(116 - 129.33)^2$ + $(145 - 129.33)^2$ + $(98 - 129.33)^2$ + $(128 - 129.33)^2$ + $(56 - 129.33)^2$ = 15792.95 + 1.77 + 11809.17 + 16726.25 + 177.69 + 245.55 + 981.57 + 1.77 + 5377.29 = 51114.01.

Calculate the average of squared deviations from the mean as, 51114.01/9 or 51114.01 * (1/9) = 5679.28

Finally, the standard deviation is the square root of average of squared deviations from the mean, i.e. $\sqrt{5679.28}$ = 75.36.

SNR can then be computed assuming the standard deviation is greater than 0 as SNR = mean/std = 1.7162.

A key point to note is that the SNR measures are averaged across all of the images in the dataset to calculate an average SNR measure for the entire dataset. Furthermore, color images have 3d arrays (the third dimension being the color channels) which get converted to a 1d array exactly as discussed in the above example i.e. row-wise serialization for the first color channel followed by the second and third.

Therefore, the SNR values for MNIST and CIFAR-10 datasets are *0.44* and *2.40*

respectively.

3.3 Relation Between ME, SNR and CNNs

As explored in Section 2.3, the first convolution layer is only able to extract a limited quantity of signal information dependent on the breadth χ'' of the layer and every subsequent convolution layer gradually builds hierarchical abstractions over these initial feature maps and extracted signal information. Ideally, with perfect information extraction the input image can be reconstructed using the signal information propagated through the layers of a CNN in this instance the classification performance would be very high. However in practice, due to random noise and other non-ideal characteristics, the classification performance of CNNs can be substantially affected.

The feature extraction capabilities of CNNs is dependent on the amount of usable information extracted in conjunction with the quality and complexity of the extracted information. Using SNR and ME measures as a proxy to measure information propagation in CNNs provides valuable insights into the effectiveness of the feature extraction process. High SNR and ME measures indicate a greater degree of usable and total amount of signal information respectively and suggests the feature extraction process is effectively being captured by the CNN.

Similarly, images that have high ME scores contain larger amounts of information, thus requiring a broader network for sufficient initial layer information extraction, whereas images with high SNR measures indicate that the usable signal information is not greatly corrupted by noise and therefore facilitates the use of deeper networks.

In instances where images have low SNR and ME measures like those images present in the MNIST dataset, any attempts made to recover the original signal information using inverse filtering and other such methods produce outputs of unacceptable quality. This is because, according to authors in (Pierce, 1980), noise and signal are interpolated,

signifying that artificially amplifying a single characteristic such as the signal in the data introduces distortions and errors which leads to uncertainties. Therefore, adopting deep network architectures on such datasets is not advisable. In Section 3.4 we put forth two arguments that tend to explain this non-ideal performance of CNNs using ME and SNR and present our experimental findings in Section 3.4.4.

3.4 Understanding Information Extraction Capabilities of CNNs

3.4.1 Information Overflow

Information overflow is the phenomenon when there is a discrepancy between the amount or quality of information extracted by the CNN to the inherent information in the input data. Overflow occurs when the first convolutional layer has saturated and extracted its maximum, but is still lower than the information present in the input image (i.e. ME and SNR measures after the first convolutional layer are lower than that of the input image). This phenomenon is usually persistent when the breadth of the first convolutional layer is insufficient or if there are deficient number of convolutional layers to completely abstract the signal information after the last convolutional layer.

As an example, according to authors in (Krizhevsky et al., 2012), for high classification performance, a relatively deeper network is required when classifying the CIFAR-10 dataset compared to the MNIST dataset. Assessing the properties of the MNIST dataset indicates lower SNR and ME values when compared to CIFAR-10, thus it is intuitive that the more complex CIFAR-10 dataset would require a deeper and broader network compared to the MNIST dataset.

If the same narrow and shallow configuration (breadth and depth) was used for both datasets, it would lead to convolutional saturation causing information overflow in

CIFAR-10. However the inverse also holds when a deeper and broader network is used for the MNIST dataset where an alternate phenomenon of information underflow would occur, explained in Section 3.4.2.

3.4.2 Information Underflow

Information underflow is the inverse of overflow where the network attempts to extract and abstract more information than is present in the input images. Information underflow is relatively less severe in terms of affecting classification accuracies. Information underflow can be characterized by overfitting (Caruana et al., 2001), while there are several methods proposed by authors in (Hawkins, 2004; Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014) and to mitigate this problem by using dropout, stochastic gradient descent and hyper-parameter tuning, it persists to a certain degree in all NNs. We also propose a few techniques based on this understanding of information underflow and overflow to mitigate the problem of overfitting in Section 3.5.

3.4.3 Experimentation

The tests were conducted with an objective to understand the effect that SNR and ME measures have on numerous neural configurations with respect to information underflow and overflow. The experimental results are presented in Section 3.4.4 were all conducted using a TFLearn front-end and a Tensorflow back-end written in python on a single 2080ti and Tesla P100 GPU with 12GB and 16GB of VRAM, generously provided by InfuseAI Limited and New Zealand eScience Infrastructure respectively. The datasets used are detailed comprehensively in 3.4.3.

Similar to the VGGNet model discussed in 2.3.1, a simpler model using the principle of stacked convolutional layers was implemented. The variations on the VGGNet model included smaller convolutional kernel sizes, shorter stride sizes, the use of a dropout

layer and a shallower depth. A dropout layer was included since it has shown to significantly reduce overfitting (Srivastava et al., 2014) for image classification tasks. No preprocessing of the data was conducted and the training-validation split was established to be 50,000-10,000 images for MNIST and 40,000-10,000 images for CIFAR-10 datasets. Experiments were performed for CNN architectures having a limited amount of convolutional layers in-line with the initial AlexNet CNN model (Krizhevsky et al., 2012) which has only five convolutional layers (Krizhevsky & Hinton, 2009).

Datasets

The MNIST dataset (LeCun et al., 1998), consists of ten black and white handwritten numerical classes (0-9) of 28×28 dimensional. The images present in CIFAR-10 dataset (Krizhevsky & Hinton, 2009) consist of ten natural image classes of 32×32 dimensional color (RGB) images, differentiated by the type of object in the images, for example cars, cats, dogs and so on. The pixel intensity values for both the datasets range from 0-255 i.e. 0 is a black pixel and 255 is white in the instance of MNIST and Red, green or Blue for CIFAR-10.

3.4.4 Results

All experiments were repeated three times and the test-set classification accuracy was averaged across the three experimental runs. The mean results are presented in Table 3.1.

MNIST Dataset		
Configuration	**Epochs**	**Acc. in %**
32 Convolutional kernels and 784 neuron FC layer	10, 500, 1,000, 1,500, 2,000	93.3, 97.0, 97.0, 97.1, 97.2
32-64 Convolutional kernels and 784 neuron FC layer	10, 500, 1,000, 1,500, 2,000	94.5, 96, 98.5, 97.7, 96.5
32 Convolutional kernels and 784-392 neuron FC layers	10, 500, 1,000, 1,500, 2,000	94.2, 96.6, 96.3, 96.89, 97.3
32-64 Convolutional kernels and 784-392 neuron FC layers	10, 500, 1,000, 1,500, 2,000	95.4, 97.4, 98.2, 98.4, 98.9
32 Convolutional kernels and 784-512 neuron FC layers	10, 500, 1,000, 1,500, 2,000	93.5, 97.5, 97.39, 97.2, 98.1
32-64 Convolutional kernels and 784-512 neuron FC layers	10, 500, 1,000, 1,500, 2,000	93.89, 97.5, 97.5, 98.1, 97.9
32 Convolutional kernels and 784-784 neuron FC layers	10, 500, 1,000, 1,500, 2,000	95.19, 96.6, 96.7, 97.5, 97.39
32-64 Convolutional kernels (2 layers) and 784-784 neuron FC layers	10, 500, 1,000, 1,500, 2,000	94.19, 96.0, 98.0, 98.4, 97.6
CIFAR-10 Dataset		
32 Convolutional kernels and 3072 neuron FC layer	10, 500, 1,000, 1,500, 2,000	21.4, 10.9, 10.2, 10.1, 10.0
32-64 Convolutional kernels and 3072-3072 neuron FC layers	10, 500, 1,000, 1,500, 2,000	43.5, 52.4, 57.12, 55.19, 53.35
32-64 Convolutional kernels and 3072-1536 neuron FC layers	10, 500, 1,000, 1,500, 2,000	45.5, 56.64, 59.43, 61.30, 60.82
32-64-128 Convolutional kernels and 3072-1024-512 neuron FC layers	10, 500, 1,000, 1,500, 2,000	43.5, 51.0, 55.2, 58.5, 61.5
32-64-128 Convolutional kernels and 3072-1536-768 neuron FC layers	10, 500, 1,000, 1,500, 2,000	60.1, 67.2, 67.45, 68.58, 68.49
32-64-128 Convolutional kernels and 3072-3072-3072 neuron FC layers	10, 500, 1,000, 1,500, 2,000	44.1, 53.8, 55.2, 60.0, 53.4

Table 3.1: Table of results comparing the average test-set classification performance for the MNIST and CIFAR-10 datasets on varied neural configurations

3.5 Discussion

The results from Section 3.4.4 suggest that certain neural configurations significantly affect image classification performance. To ensure reproducibility and draw meaningful insights which could not be attributed to random variances, statistical analyses of the experimental data were conducted using a two-tailed paired t-test. p-values below 0.05 were used to highlight statistical significance and two-tailed t-test were employed since no prior knowledge or estimates on the direction of significance were be able to determined with the null hypothesis being, no statistical significance exists between the tests.

We acknowledge that the sample size for statistical testing is low. Due to the immense computational power required for computer vision tasks along with the fact that typically the variance between multiple neural network runs on the same dataset is insignificant, extensive statistical testing is often not possible in the field of computer vision and pattern recognition. Therefore, the rest of the thesis also includes statistical testing for a small sample size.

On the MNIST dataset, increasing depths for the convolutional block while keeping the Fully Connected (FC) layer/s constant at one layer with a breadth of one neuron for every pixel i.e. 784 (28×28) produced no observable difference with a p-value of 0.558. Tests where the convolutional layer was kept constant at one layer with a breadth of 32 kernels and varying the FC depth of the classification block also yielded no observable differences with p-values of 0.0816, 0.2065, 0.9046 in the order presented in Table 3.1. The only statistical difference was observed when the breadth, depth of the convolutional layers were increased and the breadth, depth of the FC layer was decreased by a factor of two obtained a p-value of 0.0018 and an accuracy of 98.9%.

Clear statistical differences were observed for variances in both the convolutional and classification blocks. Between depths of two, three convolutional layers and two,

three FC layers yielded a p-value of 0.0382. Keeping the convolutional and FC depth constant at two but varying the breadth of the FC layers between 3072-3072 and 3072-1536 produced a p-value of 0.0142. The same patterns emerge for network depths of three and varied FC configurations with p-values of 0.0027 and 0.0006 when using FC layers and a reduction factor of two in the order of results presented in Table 3.1, the highest accuracy obtained was 68.49%. There were no other observable statistical differences.

Based on these observations, we recommend adopting a broader convolutional and FC layers with halving the number of units for each subsequent layer/s added until the phenomenon of underflow is observed. the depth of the network is restricted to the initial breadth and thus might become a limiting factor for more complex datasets due to the associated memory and computational constraints.

3.5.1 Limitations

The key limitation of the study is that the relationship between ME, SNR and information propagation of CNNs along with the phenomena of information overflow and underflow were studied only on simple non-residual architectures where skip connections are not implemented. Further, the ME and SNR measures for the multi-class image datasets were computed for image individually and then averaged across the entire dataset. The averaging of SNR and ME measures prompt further investigation into if there exists a correlation of individual classes contributing to the phenomenon of information underflow or overflow. These limitations pose an intriguing research area for possible exploration in future publications.

3.6 Conclusion

We can infer from the results that our hypothesis that information underflow and overflow characteristics are influenced by the ME and SNR measures of the input data vectors. Classification performance of CNNs are regulated by these phenomena and during instances where information underflow is predominant, variations in topological configurations play an inconsequential role since the information propagation is unimpeded. Topological variations play a prominent role only when information overflow is more pronounced where statistically significant differences in classification performance can be observed.

Furthermore, the best classification accuracies were obtained when the neurons in the FC layers were halved for additional layer. We can postulate that in order to enhance classification performance, forced abstractions might play an important role as abstractions are encoded and decoded along the full depth of the CNN model (Hinton & Salakhutdinov, 2006). Furthermore, due to the topological and informational sparsity, theoretical performance gains could be achieved by augmenting the datasets to increase the feature extraction capability of the convolutional block in a CNN. We explore this concept along with studying convolutional kernel saturation in the next chapter (Chapter 4).

In conclusion, experimental evidence was presented to support the claim that CNNs have limited informational feature extraction and abstraction capabilities. These limitations were theorized to occur when either phenomena of information overflow or underflow were predominant. ME and SNR measures were established as accurate heuristics to quantify the magnitude and influence of either phenomena on information propagation in CNNs.

The understanding and insights gained from this chapter around information propagation in CNNs lead us to examine ways of improving information propagation. In the next

chapter (Chapter 4), we investigate and propose a novel data augmentation technique to improve classification performance by mitigating convolutional kernel saturation.

Chapter 4

Examining and Mitigating Kernel Saturation in CNNs

In the prior chapter (Chapter 3), we examined the feature extraction process of CNNs and discovered that images that have low ME and SNR values lead to a phenomenon of information underflow. The opposite is true for input images having high SNR and ME scores, causing information overflow. In this chapter we explore neural saturation in DNNs, more specifically convolutional kernel saturation in CNNs. We hypothesize greater semantic feature information can be extracted when the input data vector is augmented/supplemented with a bitwise *NOT* operation on the input dataset, producing an exact negative representation of the input vectors. These negative images have the same structural information as the standard images but differ significantly in their data representations. Our assumption is that the distinct data representation of the negative images decreases the probability of kernel saturation and thus increases the effectiveness of feature extractions in CNNs.

Furthermore, in the last chapter (Chapter 3), the primary limitation was that the study was conducted on simple non-residual architectures where skip connections were not implemented. In this chapter we bridge this gap by studying convolutional kernel

saturation and the effect of negative image augmentation on classification performance for a residual CNN architecture such as the ResNet-50 model explained previously in Section 2.3.1.

This chapter is published in the 46th Annual Conference of the IEEE Industrial Electronics Society, IECON-2020; © 2020, IEEE.

4.1 Introduction

The working principle of CNNs was discussed previously in Section 2.3 and illustrated in Figure 2.2. Some of the main drawbacks in training deep CNN models such as ResNet-50 with forty-eight convolutional layers is limited feature extraction capabilities (Mallat, 2016) for such a deep network. A possible explanation for this lack of feature extraction capability is attributable to the effects of neural saturation (van Hemmen & Zagrebnov, 1987). The effect of saturation on classification performance was alluded to in the previous chapter (Chapter 3), where there was a marked decrease in classification performance when the neurons were saturated.

In this chapter, we limit the scope of the study to examine neural saturation in the convolutional block primarily for two reasons, enhanced feature extraction in the convolutional block will invariably increase classification performance, and there is extensive research conducted for DNNs used in the classification block (van Hemmen & Zagrebnov, 1987; Cogswell, Ahmed, Girshick, Zitnick & Batra, 2015).

Neural saturation predominantly presents itself during the back-propagation step, when a non-linear differentiable activation function ρ is applied to an input d-dimensional vector $\mathbf{x}_i = \langle x^1, x^2 \cdots x^d \rangle$ and is said to be fully saturated when the output $y_i = \rho(\sum_{i=1}^{n} \omega_i \mathbf{x}_i + b_i)$, acquires values close to the minimum of a bounded subset of ρ, where b_i is the bias, ω_i is the weight of the computed convolutional kernels/filters/channels, such that, $\mathbf{x}_i \in \mathbf{X}$ and $\omega_i \in \mathbf{W}$.

If a convolutional kernel is fully saturated, when the gradient calculations ∇ (discussed in Section 2.2.2) for the output y_i relative to the input $\mathbf{x}_i$ i.e. $\partial y_i / \partial \mathbf{x}_i$ approaches the minimum (near zero) of ρ for a non-linear activation function. In other words, a convolutional kernel is saturated when the output y_i approaches non-differentiability at a point where the computed future weight updates are negligible causing no apparent change affecting model convergence.

Our hypothesis is that supplementing the training data with similar semantic and structural information but significantly different data representations will lead to a more accurate feature map generation. Due to the contrasting data representations of the standard and negative images we ascertain that convolution kernel saturation will be partially offset. Furthermore, the logical bit-wise *NOT* operation used to create negative images can be performed in real-time on the CPU while model training occurs simultaneously on a GPU, reducing computational overheads.

We evaluate our research hypothesis using the ResNet-50 model, initially proposed by (K. He et al., 2016) on two well known benchmarking datasets, CIFAR-10 (Krizhevsky & Hinton, 2009) and STL-10 (Coates et al., 2011). The ResNet architecture is chosen as it introduces stacked convolutional layers and skip connections which ensures feature maps can always resolve into identity transformations limiting the extent of saturation and promoting only the most important feature information contributing to classification performance is retained.

The CIFAR-10 and STL-10 datasets are chosen since they exhibit similar spatial information but differ in their image resolutions thus enabling a better understanding of the saturation phenomenon for varied image resolutions. We also employ two quantitative measures, Maximum Entropy (ME) and Signal to Noise Ratio (SNR) to measure the informational content contained in the input datasets, discussed previously in Sections 3.2.1 and Section 3.2.2.

4.2 Utilizing ME and SNR measures in understanding weight updates

The accuracy of any Neural Network (NN) depends on generating adequate internal feature representations from the input data. In CNNs, the feature representations are separated into N feature maps generated from the input image of pixel size $R \times R$ using χ' convolutional kernels from H' convolutional layers with weight matrices $\omega_{i-\chi'} \in \mathbf{W}$. The feature maps are hierarchical, meaning the feature maps generated at a given layer $h_i | h_i \in H'$ are computed from the data propagated from the preceding layer $h_i - 1$.

The three-dimensional input color images (2D image of size $R \times R$ with 3 color channels) are dimensionally reduced down to $\phi(\chi')$ 2D feature maps of size $S \times S$, such that $R \geq S$. Dimensionality reduction of a higher-order non-linear complex function into a collection of lower-order linearly separable feature maps is not always accurate and may exclude critical information. This limitation can be mitigated by introducing a non-linear activation function ρ (Arora, Basu, Mianjy & Mukherjee, 2016) to retain structural information and preserve only the most important features contributing to linear separation i.e. classification performance.

Assuming the input image is fed at the input layer denoted by pixels $R_{0,0} - R_{32,32}$ for CIFAR-10 images and $R_{0,0} - R_{96,96}$ for STL-10 images with a color channel depth of 3. The individual pixels can be accessed using the indices $R^0_{i,j}$, where the 0 denotes the input layer. The first convolutional layer output can be denoted as y^1 for the n number of input images. Similar indexing for the h_i^{th} layer can be denoted as y^{h_i} for a convolutional depth of H' layers. The back-propagation error can be calculated using the H'^{th} layer gradient $\partial y^{H'} / \partial R^{H'-1}_{0,0}$, where the input to the final convolutional layer are the feature maps generated from the $H' - 1^{th}$ convolutional layer.

The loss function L can be denoted as $\partial L / \partial y_0^{H'}$, applying the chain rule to the error function for the total number χ' convolutional kernels/filters/channels, we get

$\partial L/\partial y^{H'} = \sum_i^{\chi'}(\partial R_i^{H'}/\partial y_{H'-1})(\partial y^{H'}/\partial R_i^{H'-1})$ (W. Luo, Li, Urtasun & Zemel, 2016). Weight updates are performed using the backpropagation of errors using a partial derivative of the neural output computed using the gradient descent approach for error minimization of a loss function L. The loss function is calculated using the difference between the initial random kernel weights $\mathbf{W}$ and the output y_i for a given input $\mathbf{x}_i$, i.e. $L(\mathbf{W}, y_i(\mathbf{x}_i)) = (\mathbf{W} - y_i)^2$ for a Mean Squared Error function. Classification performance is predicated upon future weight updates, which are calculated using the gradients for all χ' convolutional kernels/filters/channels. Therefore, if some or most of the χ' convolutional kernels become saturated, the learning process is restricted and future weight updates increasingly deviate from the optimal causing convergence issues.

ME and SNR scores previously explored in Sections 3.2.1 and 3.2.2 accurately measure and quantify the information contained in the input data which aids in the qualitative analysis of the weight update mechanism. Larger ME/SNR values indicate greater usable signal information in the data, suggesting the signal can be abstracted to a higher degree using deeper networks without significant corruption. Furthermore, larger ME/SNR scores indicate greater information can be extracted without causing unexpected events such as overflow or underflow, discussed earlier in Sections 3.4.1 and 3.4.2 to occur.

Mathematically, if CNNs are sensitive to kernel saturation on standard datasets, real-time data augmentations such as rotating or cropping, which inevitably produces images with the same information characteristics will not significantly enhance the feature extraction process. It is intuitive to suggest that augmenting the input dataset with similar spatial and structural characteristics, quantified through ME measures but with high SNR values should yield increases in classification performance. Other methods to amplify the SNR characteristics will yield outputs of unacceptable quality (Pierce, 1980), which would become problematic if a conventional NN was implemented. As CNNs primarily focus on feature extraction, augmenting input data with negative images

should not adversely affect classification performance, verified through experimentation on the MNIST dataset.

4.3 Experimental Design

The experimentation revolved around establishing if different ME and SNR measures contributed to convolutional kernel saturation and its effect on model convergence. Three standard benchmarking datasets were used, MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky & Hinton, 2009) and STL-10 (Coates et al., 2011). The research hypothesis is that supplementing the training dataset with negative images increases classification accuracies by mitigating convolutional kernel saturation.

The hypothesis was examined by training ResNet-50 models with different variations of the input datasets, while keeping all other HPs such as learning rate and batch size constant with no data excluded or pre-processing steps applied to images in the datasets for three evaluation runs of 500 epoch instances. Learning rate was set to 0.001 using an adaptive optimizer (ADAM) and a batch size of 128 based on the example script provided by Keras (The software framework used).

4.3.1 Datasets

The MNIST and CIFAR-10 datasets are the standard benchmarking datasets used previously and discussed in Section 3.4.3. The STL-10 dataset includes 500 training and 800 test natural color images split into much of the same classes of natural images as the CIFAR-10 datasets but in a higher 96×96 pixel resolution, derived from the ImageNet dataset (Russakovsky et al., 2015). The standardized testing protocol for the STL-10 dataset is not adopted in this chapter, as the scope of this thesis is limited to supervised learning tasks in the domain of computer vision. To obtain the negative images of all the samples in the standard datasets, a logical bit-wise NOT operation on

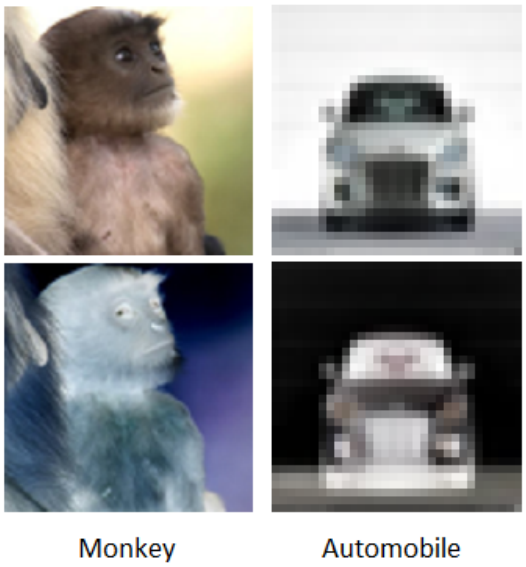

Figure 4.1: Illustration of random negative sample images for two classes in the STL-10 and CIFAR-10 datasets

every pixel in the images and separately on the three color channels, Red (R), Green (G) and Blue (B) was conducted using the OpenCV python library. An illustration of the negative images obtained is presented in Figure 4.1.

4.3.2 Experimental Setup

All experimentation was performed using a single NVIDIA 2080ti and Tesla P100 GPU with 12GB and 16GB of VRAM generously provided by InfuseAI Limited and the New-Zealand e-Science Infrastructure (NeSI) respectively. The models were trained from scratch and saved after 500 epochs and the saved models were tested using the standard test-set, the process was repeated three times and averaged to produce the final results provided in Table 4.1. The training-validation allocation for the models was kept constant at 80%-20% for all the datasets and a batch-size of 128 was employed based on the original ResNet paper (K. He et al., 2016).

The learning rate of 0.001 and the decision to implement an adaptive optimizer (ADAM) was derived from the standard test script published by Keras (the software

framework utilized for experimentation). No modifications were made to the network architecture, no real-time data augmentation methods or pre-processing of the images were implemented to ensure reproducibility of the results.

4.4 Results

The mean results from the experimentation are presented in Table 4.1. Three experimental runs were deemed sufficient as any significant differences could only realistically arise if the augmented datasets affect model convergence. The activation maps after the last convolutional layer for a test image of a cat in the CIFAR-10 dataset is visualized in Figure 4.3 and the convolutional kernel weight maps are illustrated in Figure 4.2.

4.4.1 Statistical Analysis

To determine if the raw data was normally distributed, the Shapiro-Wilk test for normality was employed with a p-value threshold of 0.05. The data was established to be normally distributed with all p-values greater than 0.05. To remove any incorrect interpretations of the data which might be due to random chance, we select the parametric paired t-test for statistical analysis. A paired two-tailed t-test is most appropriate since no assumptions on the direction of improvement are made.

Analysis was performed with the independent variable fixed as the input training dataset i.e. standard, augmented with negative images or negative images only and classification accuracy as the dependent variable. Significance was established at the standard p-value cut-off level of 0.05. The default null hypothesis is ascertained that no observable differences in accuracy can be established when the training dataset is augmented with negative images.

Both the natural image datasets (CIFAR-10 and STL-10) on the ResNet-50 showed statistical significance with p-values of 0.0013 and 0.0158 with means and variances of

Table 4.1: Summary table of results without pre-processing or real-time data augmentations

CNN Model	Dataset	Accuracy
Standard datasets		
ResNet-50	MNIST	**99.41** %
ResNet-50	CIFAR-10	80.65 %
ResNet-50	STL-10	56.08 %
Datasets augmented with negative images		
ResNet-50	MNIST	99.35 %
ResNet-50	CIFAR-10	**83.81** %
ResNet-50	STL-10	**63.06** %
Datasets with only negative images		
ResNet-50	MNIST	33.95 %
ResNet-50	CIFAR-10	43.73 %
ResNet-50	STL-10	27.96 %

The standard test-set was used for measuring performance

(80.65, 83.81), (56.08, 63.06) and (0.0121, 0.0157), (3.5087, 0.1905) respectively for the standard and negative augmented datasets. Experiments on the MNIST dataset yielded no statistical significance with a p-value of 0.0762 along with means of (99.41, 99.35) and (0.0021, 0.0002) for the standard and negative augmented datasets respectively.

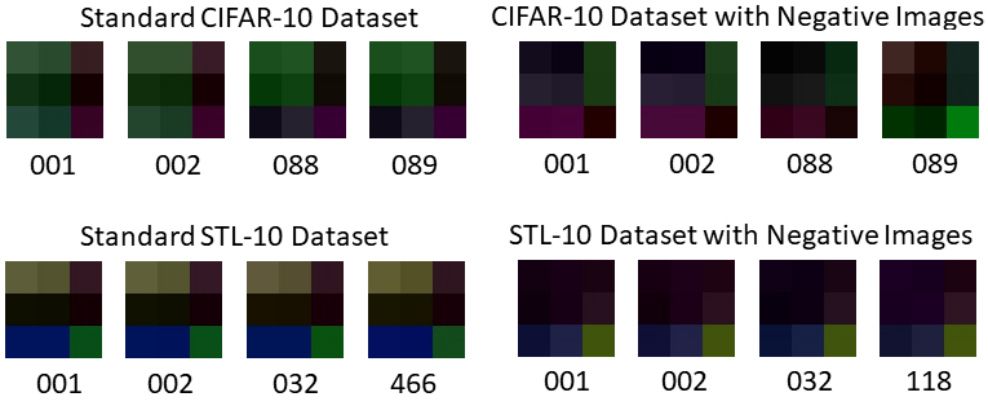

Figure 4.2: Visualized convolutional kernel weights with epoch instances for two experimental datasets extracted from a trained ResNet-50 architecture

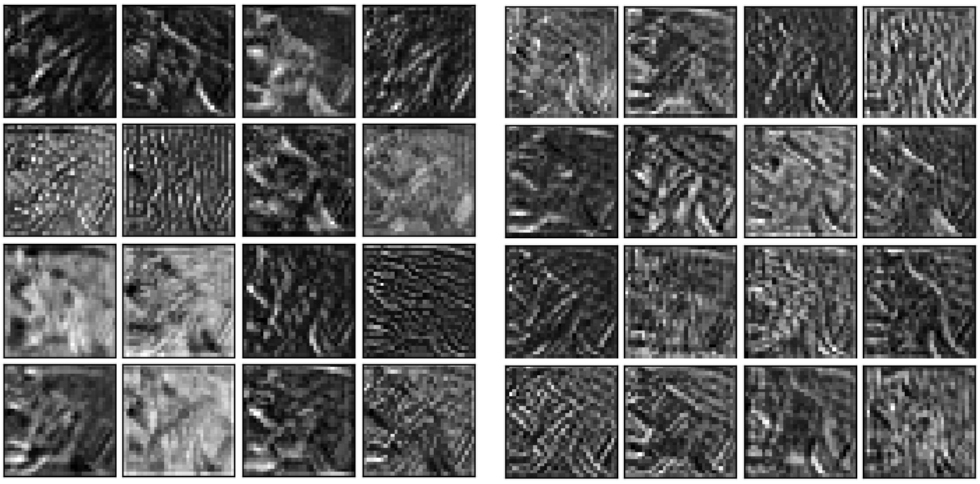

Figure 4.3: Visualized activation maps after the final convolutional layer for a trained ResNet-50 architecture (89 epochs)

4.5 Discussion

As there were no significant differences in the classification performance on the MNIST dataset, we will explore only the CIFAR-10 and STL-10 datasets for further discussion. Through experimentation on the MNIST dataset it can be established that there are no adverse impacts from negative image augmentation. The worst case scenario is that of the MNIST dataset where negative image augmentation offers no improvements on classification performance.

A visual representation of the convolutional kernel weights for the first layer of the ResNet-50 CNN architecture is presented in Figure 4.2 for the standard experimental datasets and the negative image augmented datasets. The visualizations are generated across multiple epoch instances where the model convergence improved from previous instances. Figure 4.2 illustrates the kernel weight adjustments for the CIFAR-10 dataset are more varied and spread out across a larger set of epochs, but this alone cannot verify our hypothesis that augmenting training datasets with negative images reduces convolutional kernel saturation. Therefore, visualization of the activation maps for a

random test image of a cat in the CIFAR-10 dataset is presented in Figure 4.3. We can clearly establish that much of the structural information is preserved relative to the unaugmented dataset. Furthermore, the feature extraction is smoother and more controlled for the datasets supplemented with negative images compared to the standard datasets.

The negative images are quite indistinguishable in terms of their structural and semantic information but only vary in data representations. The ME (r=image size) and SNR measures for the negative only CIFAR-10 and STL-10 datasets are 3.452 and 3.02 respectively. The standard CIFAR-10 and STL-10 datasets have ME and SNR measures of 3.139 and 0.44 respectively. Weight changes are highly tuned towards converging on a single collection of images (standard or negative) with others being neglected as they offer little significance in the back-propagation error. In other words, optimizing the loss function L to account for either the standard or negative only image datasets would yield high errors when evaluating classification performance on the other set, as evidence from experimentation using only negative images.

The lack of significantly increases in the ME values for the augmented datasets can be attributed to the relatively similar amounts of semantic information in the datasets. However, SNR measures for the augmented datasets show significant differences. This difference in SNR measures for the standard and augmented datasets indicate the true signal variance σ_S which forces convolutional kernel weight updates closer to the mean ignoring pseudo-random variances. The average ME measures for the augmented MNIST, CIFAR-10 and STL-10 datasets are 3.295, 6.850 and 6.907 respectively with corresponding SNR measures of 1.729, 2.562 and 2.323 respectively.

Assume that the elements of the convolutional kernel weight matrix $\mathbf{W} = \omega_{i-N}$ for N number of feature maps can represent the feature information in only a 2-dimensional space, then the loss function $L(\mathbf{W}, y_i(\mathbf{x}_i))$ can be trivially optimized for linear separation in a 3-dimensional space. If the loss function remains unchanged and

the weight matrix saturates, the back-propagation error calculated using the H'^{th} layer gradient $\partial y^{H'}/\partial R^{H'-1}$ will decrease the loss function L as $\partial L/\partial y^{H'}$ to ever-smaller values for a similar collection of input data (negative only or standard datasets) thus the loss function will also saturate.

Now assume the loss function for a weight matrix for negative augmented dataset, denoted as $\mathbf{x}_i^{-1}$ is $L(\mathbf{W}, y_i(\mathbf{x}_i^{-1}))$. The H'^{th} layer gradient for this new dataset would have similar semantic information (indicated by the corresponding maximum entropy measure) but would require different feature representations. Therefore, the back-propagation error would be equivalent to maximizing the previous loss function. In other words $L(\mathbf{W}, y_i(\mathbf{x}_i^{-1}))$ = 1 / $L(\mathbf{W}, y_i(\mathbf{x}_i))$. The effect of feeding in standard and negative images would thus be an alternating Loss function between the maximization and minimization of $L(\mathbf{W}, y_i(\mathbf{x}_i))$, making neural saturation less likely.

4.6 Conclusion

In this chapter, we validated our hypothesis that augmenting standard datasets with negative images can significantly increase classification performance for natural image datasets such as CIFAR-10 and STL-10 by 3.16% and 6.98% respectively. Exploration into convolutional kernel saturation yielded the same results as neural saturation in conventional DNNs that convolutional kernel saturation lead to premature convergence to local optima.

Similar to other approaches used to enhance learning algorithms, augmenting datasets with negative images does not guarantee model convergence or improve classification accuracies in all instances, as witnessed through experimentation on the MNIST dataset. While the proposed negative image data augmentation technique has a solid mathematical basis for being implemented, it does not guarantee mitigation of convolutional kernel saturation. Further experimentation using large scale datasets such as

ImageNet or MS COCO is reserved as future work. Augmenting datasets with negative images however does have a major drawback of increasing the time required for model training.

In the next chapter (Chapter 5), we look into increasing model training effectiveness by exploiting the sparsity of feature maps and investigate forced abstraction through compression of residual CNN architectures such as ResNet, ResNeXt and DenseNet (discussed previously in Section 2.3.1). As most CNN models are heavily over-parameterized, increasing model training efficiency should subsequently decrease training time.

Chapter 5

Forced Feature Compression With an Entropy-Based Heuristic

In prior chapters (Chapter 3 and 4), we looked at the properties of feature map generation and propagation in the context of information underflow and overflow. Different neural configurations were examined to study their effects on the classification performance of simple CNN models. A few discoveries made from Chapter 3 included identifying the topological and informational sparsity in CNNs. In that chapter (Chapter 3), the primary limitation was that the study was conducted on simple non-residual architectures where skip connections were not implemented. In this chapter we bridge this gap by conducting our study on residual CNN architectures such as ResNet, ResNeXt and DenseNet explained previously in Section 2.3.1.

In the previous chapter (Chapter 4), we examined convolutional kernel saturation and investigated the effects of augmenting standard datasets with negative images to enhance the feature extraction process. Although the experimental results were promising, a major drawback of the study is the direct consequence of expanding the number of input data vectors, subsequently increasing model training time. The results from previous chapters suggested significant performance gains could be achieved by

compressing the topology or limiting the number of trainable parameters in a CNN. Due to the topological and informational sparsity, forcing abstractions in a CNN model should not have a significant negative impact on classification performance, since abstractions are encoded and decoded along the full depth of the CNN model.

In this chapter, we explore the topological and informational sparsity in CNNs along with the effects of forced abstractions and over-parameterization of CNN models. We go on to propose an entropy-based heuristic to determine if model training time can be drastically reduced by limiting the number of convolutional layers/blocks and forcing feature compression, abstraction without adversely affecting classification performance.

This chapter is published in the Pattern Recognition Journal, Volume 119, November 2021, 108057; © 2021, Elsevier Ltd.

5.1 Introduction

A key challenge in designing CNN models is estimating their appropriate size (depth & breadth) since these parameters are critical in establishing a CNN's representational capacity (Krizhevsky et al., 2012). Initially, designing a CNN model seems trivial as there exists mathematical proof, that any decision boundary can be approximated with a single sufficiently broad hidden layer (Cybenko, 1989). Training a CNN model with a single broad layer is difficult, introducing afflictions like overfitting (Caruana et al., 2001) or underfitting, increased susceptibility to spatial variances in the input data (Wiatowski & Bölcskei, 2018) and ineffective feature extractions (Mallat, 2016).

Training deeper CNN models with stacked hidden layers can help mitigate training afflictions and improve model performance since deeper layers learn more complex feature representations. In the worst-case deeper layers can resolve into identity transformations (K. He et al., 2016) without incurring any performance penalties. Although training very deep CNNs with up to a thousand layers can be achieved, utilizing current

computational hardware, practical limitations such as time and cost for training such very deep CNN models become prohibitively expensive relative to shallower models.

Furthermore, training very deep, yet narrow CNN models present similar training afflictions when compared to a shallow, yet very broad CNN model (Schmidhuber, 2015). Training inefficiencies also become especially apparent when empirically shallower models learn the same functional representations and characteristics as deeper models (Ba & Caruana, 2014). Diminishing returns for the ResNet architecture with an exponential increase in layer depth results in marginal gains of accuracy as indicated in (H. Zhao, Shi, Qi, Wang & Jia, 2017) where a nominal increase in classification accuracy of 1.1% is achieved from an additional 117 convolutional layers.

While deeper CNNs are extensively employed for computer vision problems like image classification, efficient CNN architectures optimizing CNN depth (Z. Wu, Shen & Van Den Hengel, 2019), breadth (H. Zhao et al., 2017) or both (Tan & Le, 2019) are gaining prominence, with some architectures achieving good results even with limited computing infrastructures (X. Zhang, Zhou, Lin & Sun, 2018). New and emerging research trends are focussing on compression and pruning of very deep CNNs to reduce the associated computational overheads arising in training excessively deep models (Canziani, Paszke & Culurciello, 2016).

Diminishing model performance with exponential increases in the total number of model parameters can be witnessed in all CNN architectures employing skip connections as these shortcut paths essentially produce an ensemble of shallower networks (Veit, Wilber & Belongie, 2016). Therefore, experimental data suggests there might be an upper bound to model depth beyond which there is an insignificant contribution of feature abstraction from the additional layers and could even be detrimental to model performance as these additional layers might induce overfitting. The absence of a general framework to effectively determine a CNN model's size stems from an incomplete understanding of the underlying mechanisms of action.

The lack of a thorough understanding of the internal workings of CNNs have led to conjecture and opinionated postulations of researchers in justifying architecture selections. The rapid progress in the domain of computer vision has also created hurdles for performance evaluations of broader and deeper residual networks. A conclusive determination of optimal CNN HyperParameters (HPs) cannot be ascertained due to the inherent immense complexity and variability involved in computer vision tasks. HP optimization is currently dominated by practitioners knowledge on the subject matter, the computer vision task at hand and available computational resources. Scientific evaluations of model performance with regards to network depth suggests, diminishing returns in model performance for excessive network depths (Z. Wu et al., 2019) and as such, more investigation is needed to regulate CNN model depth.

Contemporary compression and pruning techniques sacrifice classification accuracy for decreasing model training times. Forcing feature abstraction and compression by constraining model depth to the entropic data distribution of the input dataset should prove be a targeted solution since critical feature information is retained compared to stochastic methods of pruning or compression.

In this chapter, we highlight CNN model training inefficiencies in deep CNNs and propose an Entropy-Based Convolutional Layer Estimation (EBCLE) heuristic to eliminate residual depth redundancies improving feature compression. Adequate feature compression enhances hierarchical feature abstraction and reduce model training time. The proposed EBCLE heuristic provides an upper bound value for model depth in CNN architectures based on the a priori knowledge of the entropic data distribution of the input dataset.

A heuristic is justified since it is well understood that optimality in terms of hidden layers cannot be accurately determined for CNN models (K. He et al., 2016). Furthermore, using entropy-based approaches for effective feature extraction is well grounded in literature (Zheng & Wang, 2018). However, the problem with using entropy measures

is that, there are numerous entropy measurements for digital data and it is imperative that a suitable entropy measure is utilized.

Shannon's Entropy (SE) (Shannon, 2001) is a measure used primarily in digital communication to improve the latency between information transmission through compression. We hypothesize that, feature compressibility and abstraction in CNNs can only ever meet but not exceed the SE measure, since it is the theoretic limit of digital data compressibility. Thus, a function of SE is justified for estimating the upper bound of convolutional depth in CNNs as these layers are principally involved in feature extraction/information processing.

As CNNs disregard the spatial orientation of the features in an image (Sabour, Frosst & Hinton, 2017), utilizing SE measures for information measurement is warranted since SE measures are independent of spatial variances in the data. The inherent problem in limiting network depth of CNNs is that, it invariably restricts the information extraction capability i.e. decreases learning capacity of the network since representational power of a CNN is proportional to its size (i.e. depth $\times$ breadth).

CNN models using simplistic datasets (MNIST) do not suffer significantly from a decrease in learning capacity due to severe model over-parameterization but, a more pronounced effect can be witnessed for complex natural image datasets such as CIFAR-100 or ImageNet due to their associated feature complexities. In order to alleviate the decrease in learning capacity from constraining network depth, a subsequent increase in convolutional breadth is necessary as discussed in Section 5.2. In our experimentation, shallower yet broader CNN models are shown to maintain or even outperform baseline test-set classification accuracies for all the five benchmarking datasets (MNIST, CIFAR-10, CIFAR-100, STL-10 and ImageNet32), while model training time decreased by 45.22% on average across three different CNN architectures (ResNet (K. He et al., 2016), DenseNet (Huang et al., 2017) and ResNeXt (Xie et al., 2017)). Furthermore, our proposed EBCLE heuristic outperforms dynamically scaling approaches utilizing

depth and breadth co-efficients (Tan & Le, 2019).

The contributions of this chapter are as follows:

- We propose an accurate heuristic to determine an upper bound to convolutional depth using Shannon's entropy measure for forced feature compression and abstraction.
- We provide empirical evidence to demonstrate that EBCLE-based shallow neural networks can learn similar high-level feature maps compared to deeper models, as presented in Figure 5.3.
- Our proposed entropy-based heuristic reduces CNN model training times by 24.99%-78.59% across three different CNN architectures and five benchmarking datasets without compromising model performance.
- We show that competitive results can be achieved using shallow yet broader CNN models relative to baseline models.
- Our experiments empirically validate and support the findings presented in (Z. Wu et al., 2019) that, deep CNN models behave as a collection of ensemble networks and the conclusions found in (Zagoruyko & Komodakis, 2016) that, wider yet shallower CNN models can learn the same functional representations as deeper yet narrower CNN models with a reduction in the associated trade-off of relative model training time.

The impact of contributions made in this chapter are apparent, EBCLE-based CNN models substantially reduce CNN model training time, democratizing research in the domain of computer vision to researchers or practitioners with limited compute capabilities. Accelerated research outputs with the opportunity to test hypotheses rapidly can be achieved through our proposed EBCLE-based heuristic. Furthermore,

researchers or practitioners can analytically justify their HyperParameter (HP) choices rather than arbitrarily selecting HP configurations. In general, all Deep Neural Networks (DNNs) exhibiting asymmetries in generalization ΔG and complexity ΔC (discussed in Section 5.2.2) should greatly benefit from feature compression to reduce model training time, regardless of the associated task such as image classification or segmentation.

5.2 Background

Consider the task of classifying high-dimensional interpolated data such as images, which can be represented by an underlying function say, $f(\mathbf{X})$ from a collection of n number of d-dimensional input image vectors, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ where, $\mathbf{x}_i = \langle x^1, \cdots, x^d \rangle | i, d \in \mathbb{Z}_{>0}; f(\mathbf{X}) \in \mathbb{R}$. All the images in $\mathbf{X}$ have an associated class label denoted as, $Y = \{y_1, y_2, \cdots, y_n\}$ such that, $(\mathbf{x}_i, y_i) \in \mathbf{X} \times Y$. The goal of image classification is to learn the underlying functional representation of $\mathbf{X}$ using the n number of input images such that, all images in $\mathbf{X}$ can be linearly separated. Traditional statistical techniques will fail in some classification tasks if the data is represented in high-dimensional vector space and as such alternate methods are needed for accurate classification.

A typical deep CNN model comprises of a convolutional block containing stacked convolutional layers, followed by a pooling layer which is followed by a classification block consisting of multiple fully connected layers with an ultimate softmax activated classification layer. The softmax layer converts all real vector values into class posterior probabilities which sum to 1. Convolutional Neural Networks (CNNs) approximate the underlying functional representation of $\mathbf{X}$ denoted as $\hat{f}(\mathbf{X})$ by projecting the higher-dimensional input vectors of $\mathbf{X}$ into a lower-dimensional vector space say, $\phi(\mathbf{X}) = \{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \cdots, \phi(\mathbf{x}_n)\}$. A simple regression vector ϑ at the final classification layer (predominantly softmax activated) can be utilized for linear separation of images in $\mathbf{X}$. The lower-dimensional feature map vector φ produced from ϕ are critical for classifying

higher-dimensional interpolated data (Krizhevsky et al., 2012). Functions f and $\hat{f}$ outputs continuous scalar values in the set of rational numbers $\mathbb{R}$ i.e. class posterior probability values provided each input vector in $\mathbf{X}$ has an associated single class label in Y.

As output values of $\hat{f}(\mathbf{X})$ are continuous scalar values, they need to be discretized into a single class from all available classes i.e. the number of classes K for the input dataset. Discretization can be achieved using a softmax layer which takes an input vector α and outputs a vector of same size β, where $\beta_i = e^{\alpha_i} / \sum_{\iota=1}^{K} e^{\alpha_\iota} | \alpha_i; \beta_i \in \alpha; \beta$. The output vector β for the input vector α is normalized such that the logits are in the interval of [0,1] and sum to 1. Consider a binary classification example, lets say for an input image $\mathbf{x}_i$ the softmax outputs are ([0.8,0.2]) i.e. an 80% confidence that the input image belongs to class one and a 20% confidence that the image belongs to class two. A threshold function can encode the logits into a one-hot vector such that, y_i = [1,0]. Model performance is dependent on an accurate dimensional reduction of the input $\mathbf{X}$ denoted as $\phi(\mathbf{X})$ as this is challenging for traditional statistical or function mapping techniques, alternatives need to be explored.

Computing $\phi(\mathbf{X})$ could be achieved through feature extraction rather than mapping the underlying function of $\mathbf{X}$. CNNs are state-of-the-art for feature extraction as they utilize convolutional kernels/channels/units/filters which are scanned across the input image in $\mathbf{X}$ producing feature maps. The number of convolutional kernels/channels/units/filters can be denoted as $\chi' | \chi' \in \mathbb{Z}_{>0}$. The χ' number of convolutional kernels have an associated weight vector $\mathbf{W} = \{\omega_1, \cdots \omega_{\chi'} | \mathbf{W} \in \mathbb{R}\}$ extract lower-dimensional feature information for the input vectors in $\mathbf{X}$ to produce a feature map vector $\phi(\mathbf{X}) = \langle \sum_{i=1}^{n} \phi_j(\mathbf{x}_i), \cdots, \phi_{\chi'}(\mathbf{x}_i) \rangle$, illustrated in Figure 5.2. Closer approximations to $f(\mathbf{X})$ can be accomplished by adjusting the weight vector $\mathbf{W}$ of the χ' number of conv. kernels until $\hat{f}(\mathbf{X})$ can approximate a one-dimensional projection of $f(\mathbf{X})$. A full-linear separation of $f(\mathbf{X})$ can be achieved for the regression vector ϑ given

an optimal weight selection for $\mathbf{W}$ where, $|\phi| = d'$ is much larger than the d-dimensional input vector $|\mathbf{X}| = d$ i.e. $1 \leq d \ll d'$.

Assuming a constant $f(\mathbf{X})$ for $\mathbf{X}$, computing $\hat{f}(\mathbf{X})$ is given by Equation 5.1 (Mallat, 2016),

$$\hat{f}(\mathbf{X}) = \langle \phi(\mathbf{X}), \mathbf{W} \rangle = \sum_{j=1}^{\chi'} \phi(\mathbf{X}), \omega_j. \tag{5.1}$$

$\mathbf{W}$ is the weight vector for which the regression vector ϑ is optimized utilizing the n number of training images in $\mathbf{X}$ and $\phi(\mathbf{X})$ is the feature map vector computed using the feature vector $\phi(\mathbf{X})$ for an input vector $\mathbf{x}_i \in \mathbf{X}$. Theoretically, deeper CNNs have an increased capacity to compute a much closer approximation to $f(\mathbf{X})$ compared to shallower CNNs, since deeper networks can abstract more complex feature maps. This is the reason why the enumeration of the feature map vector $\phi(X)$ exponentially grows for a given underlying function $f(\mathbf{X})$ for a given dataset. A larger feature map vector $\phi(X)$ can also be achieved using a shallower yet broader model since they are functionally equivalent to a deeper yet narrower model in terms of generating feature maps shown, discussed later in this chapter and illustrated in Figure 5.3.

It is worth emphasizing that the dimensionality of feature map vector $|\phi| = d'$ can only be reduced by increasing the convolutional depth, validating the arguments made for using ever deeper CNN models (K. He et al., 2016). An oversight to this argument is that, reductions in dimensionality increasingly deviate the computed $\hat{f}(\mathbf{X})$ from the ideal functional representation since activation functions apply approximations at each layer and residual models behave as an ensemble of smaller networks (Z. Wu et al., 2019) thus increasing redundant feature extractions.

In Section 5.3, we mathematically deduce that increasing convolutional depth beyond Shannon's entropy measure adds to redundancies, which is consistent with the findings presented in (Z. Wu et al., 2019). Furthermore, in Equation 5.1, the problems

of using deeper networks become especially apparent as χ' plays a parallel role to $\phi(\mathbf{X})$, which conforms to the conclusions presented in the previous work of (Z. Wu et al., 2019) and is the reason why wider residual networks (Zagoruyko & Komodakis, 2016) perform better than a thousand layer deep network. The dimension of the feature map vector, $|\phi(\mathbf{X})| = d'$ is often why CNN models overfit or underfit to the input data.

5.2.1 Convolutional Neural Network Architectures

Residual Network (ResNet)

Residual Network (ResNet) (K. He et al., 2016) using skip connections was proposed to solve the degradation problem existing in deep CNNs. The ResNet architecture is a deployment of residual learning-oriented blocks. The goal of using these blocks is to integrate nonlinear convolutions into residual operations, which greatly reduce the difficulty of linear separation in pattern classifications.

Research into shortcut paths to address the problems of vanishing/exploding gradients has been undertaken since the days of Multi-Layer Perceptrons (MLPs) but, offered little improvements. Variations on the shortcut connection paths have since been used in state-of-the-art CNN models (Szegedy et al., 2015) such as ResNets with skip connections between hidden layers to allow for retention of the initial features. ResNet follows the function of error minimization for the input training data $\mathbf{X}$ given in Equation 5.2

$$\hat{f}(\mathbf{X}) =< \phi(\mathbf{X}) + \mathbf{X}, \mathbf{W} > \tag{5.2}$$

Densely Connected Convolutional Network (DenseNet)

Densely connected convolutional network (DenseNet) (Huang et al., 2017), was conceived from a simple idea that the output of any hidden layer $h_i|h_i \in H; i \geq 2$ where, H is the depth, should include the concatenation of all preceding feature maps produced

from h_i-1 layers. DenseNet also obeys the error minimization function given in Equation 5.3, which requires the computation of a lower dimensional feature vector from the input data **X**.

$$\hat{f}(\mathbf{X}) = \langle \phi([\mathbf{x}_i(1), \cdots, \mathbf{x}_n(h_i - 1)]), \mathbf{W} \rangle \tag{5.3}$$

Growth rate is a critical parameter in DenseNet, viewed as the total amount of feature information contributed by an individual layer to the entire network. DenseNets with similar capacity can have varying classification performance by adjusting the growth rate HyperParameter.

Aggregated Residual Transformations (ResNeXt)

A neuron can be thought of as an aggregation of signal transformations from all the input data. The principle of ResNeXt (Xie et al., 2017) is to replace the computation of $\varphi(\mathbf{X})$ with the transformation of input **X** as $\tau(\mathbf{X})$. ResNeXt, i.e., aggregated residual transformations, can be represented by the error minimization function given in Equation 5.4.

$$\hat{f}(\mathbf{X}) = \langle \sum_{i=1}^{L} \tau(\mathbf{X}_n^L), \mathbf{W} \rangle, \quad n, L \in \mathbb{Z}_{>0} \tag{5.4}$$

Cardinality L is the fixed size of aggregated transformations. Cardinality is an important HyperParameter affecting model capacity similar to network depth.

5.2.2 CNN Optimization

Parameter compression and pruning

Compressing CNN models either through spatial or channel decomposition (Denton, Zaremba, Bruna, LeCun & Fergus, 2014) is extensively adopted in practice to increase

training efficiency by removing depth redundancies. While channel (Y. He, Zhang & Sun, 2017) and spatial (J.-H. Luo & Wu, 2017) pruning show significant reduction in model training time, they inevitably offer lower classification performance compared to a deeper un-pruned CNN models. In Section 5.2.2, the ineffectiveness of spatial compression is discussed. In Section 5.3, the variances in convolutional outputs from channel pruning are highlighted. CNN parameter pruning is also challenged, as broader models with greater numbers of trainable parameters outperform narrower yet deeper models with lower numbers of trainable parameters in terms of training time as they can be more efficiently computed in parallel.

Efficient Convolutional Neural Network (EfficientNet)

The premise behind EfficientNet is that, CNN models are developed with a fixed resource budget and are then scaled up to improve model performance. A uniform compound co-efficient is introduced as an alternative to single-dimension scaling (Tan & Le, 2019). The authors argue that compound scaling is warranted for increasing the receptive field important to capture fine-grained patterns in large images. In this chapter, in Section 5.5, we present empirical evidence to support targeted depth scaling (utilizing Shannon's entropy measure) and manual width scaling constrained only by computational resource budgets offers similar or even enhanced model performance compared to uniform scaling approaches while significantly decreasing training time.

Information theory and Entropy

Information theory has wide-ranging applications in interdisciplinary domains such as communication systems and complexity theory. Information theory is a derivative of probability theory where the probability measures of particular events are used to determine the complexity of information contained in events (Reza, 1994). The equation to determine the total amount of information contained in an input X for a given event

E is presented in Equation 5.5.

$$I(\mathbf{X}) = ln(1/p_E) = -ln(p_E) \tag{5.5}$$

Where, $I(\mathbf{X})$ is the total amount of information contained in the event for the input dimensional vector $\mathbf{X}$. The number of states or independent symbols that a single element for a single instance of X, denoted as $\mathbf{x}_i^j$, where $i, j \in \mathbb{Z}_{>0}$ can exist in is denoted by a for an event $\mathcal{E}$ with the natural log $ln(\cdot)$ representing the probability p. As a natural log is used, the unit of measurement is NATural units (nats). A natural log with base $e = 2.7182...$ is appropriate in this context as selecting any other logarithm base would restrict the true measurement of representational power. According to (Shannon, 2001), the total amount of information contained in any given data is expressed through its entropy (E), which can be calculated using Equation 5.6.

$$E(\mathbf{X}) = -\sum_{1}^{n}\sum_{i=0}^{a} p_{\mathcal{E}}\, ln\, p_{\mathcal{E}} = \sum_{1}^{n}\sum_{i=0}^{a} p_{\mathcal{E}}\, I(\mathbf{x}_i), \quad n, a \in \mathbb{Z}_{>0} \tag{5.6}$$

In Equation 5.6, E is the Shannon's Entropy (SE) measure in NATural units (nats), $p_{\mathcal{E}}$ the probability of choice for a distinct independent symbols and n is the number of training data/images. Equation 5.6 also implies that the entropy measure is dependent on the total amount of information in an event and the probability of its stochastic source. In other words, if new events yield no new information, the entropy would be zero. In digital images, the a value for a grayscale image would be 256 for 8-bit images, i.e. $2^8 = 256$ or 0 to 255 distinct gray values. $p_{\mathcal{E}}$ is the probability of a pixel possessing the gray value a.

As determining probability and relative probability measures for digital images is impractical due to the high-dimensional interpolated nature of images, histograms or frequency of pixel intensities are computed instead to calculate close approximations to actual probabilities utilizing the open-source scikit-image library written in python. The

same process can be applied for color images but probability measures are computed for every color channel i.e. Red(R), Green(G) and Blue(B) color channels.

It is worth highlighting that images with different spatial configurations have the same entropy measures. The loss of accountability in measuring spatial configurations is a drawback of CNNs in general (Sabour et al., 2017). SE calculations also disregard spatial variations during measurement and as such, the SE measure is a perfect metric for quantifying the amount of information $I(\mathbf{X})$ in an image.

Using the skikit-image library, we calculate the entropy measures for all the training images present in the MNIST, CIFAR-10/100, STL-10 and ImageNet32 datasets, described further in Section 5.4.1. The SE measures of all the training images are then averaged (as the CNN should be able to generalize between all classes of images) across the entire training set and rounded to two digits. The averaged entropy measures are **MNIST: 2.14, CIFAR-10 and STL-10: 5.03, CIFAR-100: 4.97** and **ImageNet32: 4.97**. As most of the natural image datasets contain images from much of the same classes, it is not surprising that they have similar entropy measures.

5.3 Entropy-Based Layer Estimation

There are multiple methods proposed to estimate layer/neural configurations of networks as discussed in Sections 5.3 and 5.3. In this chapter, we primarily focus on feature extraction, abstraction and compression to determine the upper and lower bounds for the input vector and a heuristic upper bound to estimate the number of hidden layers required in a CNN. As computation of $\phi(\mathbf{X})$ is predicated upon the information extracted within the hidden layers of a CNN, discussions around information theory and its principles are warranted and most appropriate.

Mutual Information Neural Estimation (MINE)

Authors in (Belghazi et al., 2018) empirically demonstrate that Shannon's entropy-based measures to determine mutual information of images $(\mathbf{x}_i, y_i) \in \mathbf{X} \times Y$ decreases the uncertainty in approximating the underlying function $f(\mathbf{X})$ given the computation of conditional entropy. The equation to determine mutual information between two vectors $\mathbf{X}$ and Z is given in Equation 5.7,

$$I(\mathbf{X}; Z) := E(\mathbf{X}) - E(\mathbf{X}|Z) \tag{5.7}$$

Where, E is Shannon's entropy measure and $E(\mathbf{X}|Z)$ is the conditional entropy of $\mathbf{X}$ given Z. Theoretic proofs of MINE exhibit strong consistency for multi-variate information estimation while capturing non-linear dependencies. Furthermore, MINE has been empirically validated to outperform non-parametric estimation in (Kraskov, Stögbauer & Grassberger, 2004). MINE performs well for adversarial networks and proves tractable for applications utilizing the **principle of Information Bottleneck (IB)** but, no evidence is presented in terms of its applications in Deep Neural Networks (DNNs). Furthermore, MINE is used as an objective function in adversarial setting to maximize $I(\mathbf{X}; Z)$. IB has shown to approximate optimal representations of $\mathbf{X}$ with respect to Y in a discrete setting and with the addition of a small noise in a stochastic setting for both adversarial networks and DNNs (Shwartz-Ziv & Tishby, 2017). Therefore, MINE's application is limited for DNNs but offers strong empirical evidence that SE can be utilized as a quantitative metric for information compression in neural networks outperforming other estimation methods.

Mutual Information of Layers in Deep Neural Networks

According to the authors in (Shwartz-Ziv & Tishby, 2017), the commonly used Stochastic Gradient Descent (SGD) optimizer in DNNs behaves in two different and

distinct phases, Empirical erroR Minimization (ERM) and representation compression, with the phases characterized by variations in the gradients Signal to Noise (SNR) ratios of individual layers. The ERM phase results in a rapid increase of the mutual information $I(\mathbf{X};Y)$ with respect to the class label Y and the compression phase (the majority of model training is utilized in this phase) is marked by a slow compression of the feature representation of $\mathbf{X}$. Furthermore, the authors in (Shwartz-Ziv & Tishby, 2017) empirically demonstrate that the optimized layers approach the optimal IB bound which plays a pivotal role for computational and accuracy trade-offs.

In a multi-class classification problem a single-objective optimization T_ϵ of the hidden network layers (H') between $1 \leq \epsilon \leq H'$ is dependent on a multi-objective optimization of $I_{\mathbf{X}} = I(\mathbf{X};T_\epsilon)$ and $I_Y = I(T_\epsilon;Y)$. The ERM phase of model training minimizes the cross-entropy loss characterized by I_Y while the compression phase optimizes $I_{\mathbf{X}}$ which can be represented as $I_{\mathbf{X}} = E(\mathbf{X}) - E(X|T_\epsilon)$, If the input entropy $E(\mathbf{X})$ is invariant, optimizing $E(X|T_\epsilon)$ is sufficient, also known as stochastic relaxation. As CNNs are fundamentally differentiated by their convolutional operations to extract feature representations from input $\mathbf{X}$, authors in (Shwartz-Ziv & Tishby, 2017) assert the **entropy growth ΔE for convolutions is logarithmic** in the number of time steps i.e. $\Delta E \; \alpha \; log(\mathcal{D}_t)$ where, $\mathcal{D}$ is the underlying data distribution from which the independent input samples $\mathbf{X}$ are obtained.

There is an exponential decrease in model training time with reduced network depths due to stochastic relaxation. In other words, the IB bound is greatly responsible in optimizing $I(\mathbf{X};T_\epsilon)$ and since shallower networks have fewer number of hidden layers the representation of $I(\mathbf{X})$ is subsequently constrained and thus will train faster given identical computational resources relative to deeper networks. The decrease in representational capacity and methods of mitigating representational losses for shallower networks are explored in Section 5.3.1.

A search on the information plane (illustrated as Figure 5.1) i.e. $I_{\mathbf{X}}$ and I_Y could

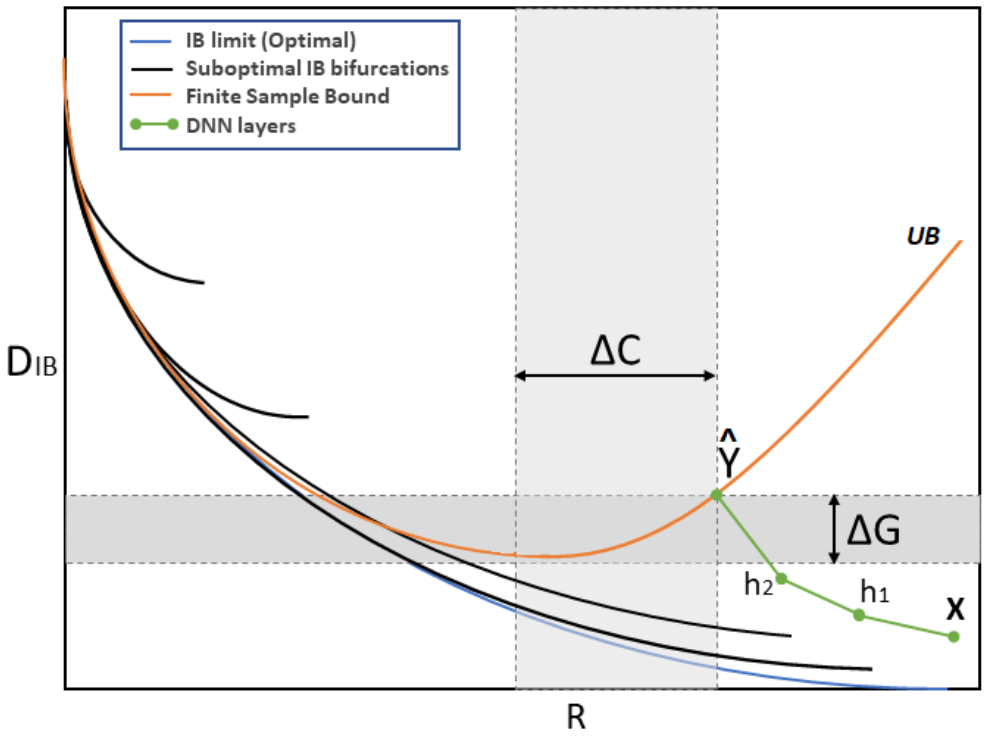

Figure 5.1: Information plane with a hypothesized layer path in a DNN for finite set of samples in **X**. ΔC is the complexity gap and ΔG is the generalization gap, D_{IB} is the optimal achievable IB limit for samples in **X**, $R = I(X;\hat{X})$ and UB is the upper bound on the out-of-sample IB distortion. Figure reproduced from (Tishby & Zaslavsky, 2015)

yield an upper bound for convolutional layer estimation but, this method involves a pre-training step which is both computationally and time sensitive. In other words, a trial-and-error approach could be adopted to get an ideal curve for $\hat{Y}$ to minimize ΔC and ΔG by evaluating R and varying D_{IB} but, this is time consuming and requires additional computational resources.

We propose a logical upper bound and heuristic convolutional depth in Section 5.3.3 using only the a priori knowledge of the SE measure of **X** i.e. $E(\mathbf{X})$ without pre-training.

5.3.1 Entropy and convolutional depth

As discussed in Section 5.3, authors in (Shwartz-Ziv & Tishby, 2017) assert that exponential decreases in model training times are achieved with a reduction in network depth since the majority of model training time is dedicated to feature compression. However, as discussed in Section 5.2.2, compression or pruning inevitably results in adverse model performance due to the associated loss in model learning capacity. The characteristic nature of deep Convolutional Neural Networks (CNNs) using skip connections (such as the ResNet, DenseNet and ResNeXt architectures discussed in Section 5.2.1) resolve into an ensemble of shallower networks (Veit et al., 2016) suggesting limiting convolutional depth to enhance feature compression could potentially decrease training time without a significant impact on model performance.

Limiting convolutional depth will invariably constrain the formation of ensembles of shallower networks and a corresponding expansion of the convolutional breadth (i.e. the number of convolutional kernels/channels/filters/units in a hidden layer) should counteract the problem of decreased model learning capacity. While limiting convolutional depth is in stark contrast to the work done by authors in (J.-H. Luo & Wu, 2017) proposing convolutional channel pruning, there is empirical evidence (Ba & Caruana, 2014) supporting the fact that shallower networks can learn similar complex feature representations as deep networks, primarily because majority of model training is dedicated to feature compression (Shwartz-Ziv & Tishby, 2017) during which redundant information is compressed and only the most important features improving model performance are retained.

An ideal determination of network depth is impractical due to the fact that residual connections propagate information non-sequentially between layers and they can always learn identity transformations allowing for training of very deep CNNs with up to and beyond a thousand layers (K. He et al., 2016). Heuristic optimization of the network

depth is desirable since lowering architectural complexity decreases the generalization gap but increases the informational complexity gap, as illustrated in Figure 5.1 i.e. allows for a broader representation of $\mathbf{X}$ in the information plane, $I_{\mathbf{X}}$ and I_Y. Ideally, the level of abstraction within a CNN should be equal to the informational complexity of the input dataset. Although this would be ideal, there are no methods of estimating when this level of abstraction is achieved.

The abstraction capability of CNNs is reliant on the representational power of the model, which refers to the ability of the network to accurately extract and represent information in feature maps. Increasing the representational capacity in CNNs acts as a compensation mechanism for the loss of spatial information during the abstraction process. The representational capacity of CNN models discussed in Section 5.2.1 is increased with each additional convolutional layer. Note that however, although each additional convolutional layer increases the representational capacity of a CNN, these additional layers might be performing identity transformations which do not contribute in enhancing model performance. Furthermore, the compression phase of model training for deep CNN models require an exponential increase in computational resources and training time.

Input compression bound

Authors in (Shwartz-Ziv & Tishby, 2017) proposed a new input compression bound presented as Equation 5.9, to replace the generalization bounds defined by classic learning theory presented as Equation 5.8.

$$\epsilon^2 < \frac{log|\mathcal{H}_\epsilon| + log1/\delta}{2n} \tag{5.8}$$

Where, ϵ is the difference in errors between training ΔC and generalization ΔG as illustrated in Figure 5.1. $\mathcal{H}_\epsilon$ is the ϵ-cover for a depth hypothesis assuming the

size $|\mathcal{H}_\epsilon| \sim (1/\epsilon)^d$. d, the dimensionality of n number of input samples in $\mathbf{X}$. δ, the confidence interval of $\hat{Y}$ is between [0,1].

$$|\mathcal{H}_\epsilon| \sim 2^{|\mathbf{X}|} \rightarrow 2^{T_\epsilon} \tag{5.9}$$

Where, the size of input vector $\mathbf{X}$ is $E(\mathbf{X})$ given $\mathbf{X}$ is large. T_ϵ is the single-objective optimization as an ϵ-partition of the input vector $\mathbf{X}$ of size $2^{E(\mathbf{X}|T_\epsilon)}$, assuming 2^{T_ϵ} is the cardinality for a depth hypothesis $H_\epsilon|T_\epsilon \in 1 \leq \epsilon \leq H'$. Furthermore, $|T_\epsilon| \sim \frac{2^{E(\mathbf{X})}}{2^{E(\mathbf{X}|T_\epsilon)}} = 2^{I(T_\epsilon;\mathbf{X})}$, discussed in Section 5.3. Simplifying Equation 5.9, the input compression bound can be presented as Equation 5.10,

$$\epsilon^2 < \frac{2^{I(T_\epsilon;\mathbf{X})+log1/\delta}}{2n} \tag{5.10}$$

Assuming an absolute confidence and a finite number of samples images (in-bound disregarding out-of-bound distortions), the compression bound in Equation 5.9 is dependent on $I(T_{\epsilon;\mathbf{X}})$, therefore maximizing $I(T_{\epsilon;\mathbf{X}})$ should be sufficient for enhanced feature compression.

Stochastic relaxation: As discussed in Section 5.3, Layer compression can be computed as $\Delta E_i = I(\mathbf{X};T_i) - I(\mathbf{X};T_{i-1})$ for a given hidden layer $h_i \in H'$. Implying an exponential decrease in training time for decrements in the number of hidden layers. Our hypothesis is that redundant information in the input vector $\mathbf{X}$ can be compressed as X_E for which the resultant feature maps generated by the hidden layers of a CNN cannot exceed Shannon's Entropy measure $E(\mathbf{X})$. In other words, limiting convolutional depth (H') with a corresponding increase in convolutional breadth (χ') for a CNN should exponentially decrease training time without compromising model performance. The progressive increase in spatial convolutions is exponential for $H' \times \chi'$ and is in the order of $2^{H'\chi'}$ (Mallat, 2016) i.e. $2^{H'\chi'} \leq I(\mathbf{X}_E) \leq E(\mathbf{X})$, the CNN compressed feature vector cannot exceed the theoretical limit compression of the input vector. Furthermore, the

representational capacity of a CNN is proportional to its size (depth × breadth) and its size is 2^{T_ϵ} (from Equation 5.9) i.e. $I(\mathbf{X}_E; H') \leq 2^{E(\mathbf{X})}$, the information contained in a H' deep CNN is distributed among all of the convolutional kernels which is its representational capacity obtained after the compression phase of model training.

5.3.2 Upper bound of convolutional depth

Determining an adequate convolutional depth for which the model provides sufficient dimensionality reductions without introducing inefficiencies or redundancies is a challenging problem. Assume the d-dimensional input vector $\mathbf{X}$ has no redundancies i.e. stochastic noise, in this instance there are no practical ways to apply stochastic relaxation without compromising model performance. This can be considered the lower-limit for hidden layer compression where a convolutional depth estimation is impossible since additional layers will increase model performance significantly.

As most information captured in the real-world has some redundancy, the n samples in input $\mathbf{X}$ can be compressed up to the theoretical limit i.e. Shannon's Entropy (SE) measure E (computed using Equation 5.6). Lets denote the compressed input vector as $\mathbf{X}_E | \mathbf{X}_E \leq \mathbf{X}$. *Increasing the number of input samples will in effect reduce the suboptimal IB bifurcations as illustrated in Figure 5.1.* As discussed in Section 5.3, in instances of stochastic relaxation, optimizing $I_{\mathbf{X}}$, specifically $E(\mathbf{X}|T_\epsilon)$ is adequate for exponential decreases in model training times. We know that entropy growth ΔE is logarithmic, particularly ln (From Equations 5.5 and 5.6) and $\Delta E_i = I(\mathbf{X}; T_i) - I(\mathbf{X}; T_{i-1})$ (From Section 5.3.1).

The final convolutional layer output for a CNN model of depth H' is dependent on the previous layer $H' - 1$ and the output for layer $H' - 1$ is determined by the output from its previous $H' - 2$ layer and so on until the first input layer. Therefore, entropy growth ΔE can be rewritten for the entire convolutional depth of a CNN as Equation

5.11,

$$\Delta E = ln(\sum_{i=2}^{H'} I(\mathbf{X}; h_i) - I(\mathbf{X}; h_{i-1})) \tag{5.11}$$

A unique characteristic of information propagation in the hidden layers is that $I(\mathbf{X}; h_i) \leq I(\mathbf{X}; h_{i-1}) \leq I(\mathbf{X}_E) \leq I(\mathbf{X})$. In other words, any information lost in the initial layer/s cannot be recovered in deeper layers (Shamir, Sabato & Tishby, 2010). Furthermore, for any $i \geq j$, $I(Y; \mathbf{X}) \geq I(Y; \mathbf{X}_E) \geq I(Y; h_j) \geq I(Y; h_i) \geq I(Y; \hat{Y})$ holds true. $I(Y; \hat{Y})$ quantifies the predictive features in $\mathbf{X}$ for Y, determining $I(\mathbf{X}; H')$ i.e. the final convolutional layer should yield an upper bound for depth estimation.

The feature map outputs of any convolutional layer is governed by the non-linear activation function $\rho(\cdot)$, most commonly the Rectified Linear Unit (ReLU), $\rho(Z) = max(0, Z)$ for some vector input Z (Arora et al., 2016). The activation function essentially bottlenecks information propagation within the hidden layers, such that, $E(\mathbf{X}) \geq E(\mathbf{X}_E)$ and $I(\hat{Y}; \mathbf{X}_E) \leq \rho(I(\mathbf{X}_E))$. The final layer output for a convolutional depth H' requires as an input the compressed vector $\mathbf{X}_E$ (because only the first convolutional layer can accept the uncompressed input vector $\mathbf{X}$, all other layers have the feature map output from the first layer as an input) and is constrained by the activation function i.e. $\rho(I(\mathbf{X}_E; H'))$.

Equation 5.11 can be rewritten and reduced as Equation 5.12,

$$\Delta E = ln(\rho(I(\mathbf{X}_E; H'))) \tag{5.12}$$

$I(\mathbf{X}_E) = 2^{H'\chi'}$ and $I(\mathbf{X}_E; H') \leq 2^{E(\mathbf{X})}$ (from Section 5.3.1 applying stochastic relaxation). The activation function ensures to maximize $I(\mathbf{X}_E; H')$ and equality is achieved if and only if $\hat{\mathbf{X}} = \mathbf{X}$. Therefore, the relationship is invariant for $I(\mathbf{X}_E; H')$ i.e. $\rho(I(\mathbf{X}_E; H')) \leq 2^{E(\mathbf{X})}$. In other words, the final compressed feature vector cannot exceed the theoretical compressibility of the input vector $\mathbf{X}$ and equality is achieved in

the best case when $\hat{\mathbf{X}} = \mathbf{X}$. Therefore, Equation 5.12 can be rewritten as an upper bound of convolutional depth (which is the Entropy-Based Convolutional Layer Estimation or EBCLE equation) as Equation 5.13

$$\Delta E = ln(2^{E(\mathbf{X})}) \tag{5.13}$$

Where, $\Delta E \in \mathbb{R}_{1}^{\dagger}$ is the objective function for maximizing feature compression within the hidden layers of a CNN, given the Shannon's Entropy measure $E(\mathbf{X})$ for an input dataset $\mathbf{X}$.The complexity of $\mathbf{X}$ determined by its entropy measure E, indicates higher degree of data complexity requires CNN models with corresponding complexity for dimensionality reduction and linear separation.

Note that since the *EBCLE* heuristic for feature compression belongs to $\mathbb{R}_{1}^{\dagger}$, upper and lower bound values are mandatory. It is safe to assume that the upper bound should be used, as using the lower bound might lead to premature feature complexity growth. Utilizing Equation 5.13, we can determine the upper bound heuristic for the selected input datasets **CIFAR-10/100, STL-10 and ImageNet32** as **4** and the **MNIST** dataset as **2**.

5.3.3 Using EBCLE for CNN Architectures

The EBCLE heuristic or ΔE offers a way to maximize feature compression by utilizing minimal number of hidden convolutional layers and as such this upper bound measure behaves differently for various static CNN architectures due to their architectural constraints. All the CNN architectures employed in this chapter have residual learning blocks with stacked convolutional layers, these stacked convolutional layers should not degrade model performance since they can always perform identity transformations (K. He et al., 2016). Since stacked convolutions reduce dimensions exponentially, shortcut paths are introduced after each learning block to ensure model performance

(feature learning capacity) is not impeded. Therefore, an architectural design lower-bound is placed on model depth.

The design limitations proposed by the authors are, ResNet v1: Depth = $N \times 6 + 2$ (K. He et al., 2016); DenseNet: Depth = $N \times 3 + 4$ (Huang et al., 2017) and ResNeXt: Depth = $N \times 9 + 2$ (Xie et al., 2017). $N|N_{>1+}$ is the EBCLE value derived earlier in Section 5.3.2, i.e. 4 for CIFAR-10/100, STL-10, ImageNet32 and 2 for MNIST. As such, the lower bound for model depth that can be employed for these architectures are, ResNet v1: 8, DenseNet: 7 and ResNeXt: 11.

5.4 Experimental Design

To validate EBCLE as a heuristic, we employ a quantitative approach using five well-known benchmarking datasets, MNIST (LeCun et al., 1998), CIFAR-10/100 (Krizhevsky & Hinton, 2009), STL-10 (Coates et al., 2011) and ImageNet32 (Chrabaszcz, Loshchilov & Hutter, 2017). The selected comparison criteria are, test-set classification accuracy and the model training time. We test the efficacy of EBCLE against deeper ResNet-50, ResNeXt-56 and DenseNet-28 models (deeper, broader models could not be evaluated due to memory constraints), while keeping other HPs such as learning rate and batch size constant with no data excluded or pre-processing steps applied to images in the datasets for three independent evaluation runs. Learning rate and batch size were selected based on configurations by the original authors of the proposed architectural models.

The current consensus is that, using a trial and error methodology to vary HP configurations and using expert domain knowledge, fine-tuning of deep CNN models yield enhanced model performance (K. He et al., 2016). Our primary objective for this study is to investigate the relative classification performance of deeper yet narrower and shallower yet broader CNN models with an emphasis placed on training time.

5.4.1 Datasets

The MNIST dataset includes 28×28 pixel resolution black and white handwritten digits. MNIST consists of 60,000 training and 10,000 test images split equally into ten classes for each numeric digit. The CIFAR-10/100 datasets includes 50,000 training and 10,000 testing natural color images with a 32×32 pixel resolution, split equally into ten/hundred classes for CIFAR-10/100 respectively, which include pictures of airplanes, automobiles, birds and other such natural image classes. The STL-10 dataset includes 500 training and 800 test natural color images split into much of the same classes of natural images but in a higher 96×96 pixel resolution, derived from the ImageNet dataset (Russakovsky et al., 2015). The ImageNet32 dataset is a downsampled (32×32 pixel resolution) version of the original ImageNet dataset (Russakovsky et al., 2015), consisting of a thousand natural image classes.

5.4.2 Experimental Setup

The first set of experiments presented in Table 5.1 were conducted using a single Nvidia 2080ti GPU with an AMD Threadripper 1920x CPU and 32GB of RAM, generously provided by InfuseAI Limited (New Zealand). The second set of experiments presented in Table 5.2 and evaluation of the transfer learning performance were conducted using a single 3070 GPU with a AMD Ryzen R5 2600 CPU with 64GB of RAM, yet again generously provided by InfuseAI Limited (New Zealand). The training-validation split for every model was kept constant at 80%-20% across all datasets. There were no modifications made to the CNN architectures and to ensure reproducibility, no image augmentation techniques were used. HP configuration included using the Adam optimizer with a batch size of 128 and a constant learning rate of 0.001 for 100 epochs. Where possible, official Github repositories were cloned for the four CNN architectures built on the target software platform (Keras with a tensorflow backend) utilized in this

chapter.

All the models were trained from scratch on the specified hardware utilizing the same source code and libraries. Only the model depth (H') and breadth (χ'), presented in Table 5.1 had to be modified for baseline comparisons against EBCLE models. In other words, the baseline ResNet-50 model had 50 hidden layers (H') with 16 convolutional units for the first hidden layer (χ') whereas, EBCLE-models had 26 hidden layers with 24 convolutional units.

5.5 Our Results

The first set of experiments in this chapter is to examine model performance for static CNN architectures with an emphasis on training time with respect to EBCLE-based models, presented as Table 5.1.

The second set of experiments focused around examining model performance for dynamically scaling CNN architectures such as EfficientNet (EN). The HP configuration used was similar to the settings proposed in (Tan et al., 2019); an RMSprop optimizer with the default learning rate of 0.001 and momentum of 0.9 for 100 epochs **with no weight decays or custom layers/objects used to ensure reproducibility**. Furthermore, the image resolution for CIFAR-10 is 32×32 but, authors in (Tan & Le, 2019) trained models on the ImageNet dataset with 224×224 resolution images, therefore model performance will deteriorate significantly. All models were trained from scratch on the specified hardware.

Finally, we examine the **transfer learning** performance of EBCLE models relative to baseline. The objective is to investigate if limiting depth omits important feature information from being retained that might be pertinent for model performance. The results for transfer learning for STL-10 and CIFAR-10 datasets for the ResNet models were on average **15.75% and 16.93% for the ResNet-50 and ResNet-EBCLE models**

CNN Model	H'	χ'	Params.	**Acc. (%)**	**Time (h:m:s)**	**REL. Δ (%)**
MNIST dataset						
ResNet-50	50	16	760,266	**99.44**	0:43:31	-65.68
ResNet-EBCLE	14	24	400,474	99.42	0:14:56	
DenseNet-40	40	12	1,058,866	**99.60**	1:03:46	-78.59
DenseNet-EBCLE	10	20	210,050	99.54	0:13:39	
ResNeXt-56	56	16	11,003,712	99.15	3:48:01	-63.12
ResNeXt-EBCLE	20	16	3,893,056	**99.21**	1:24:05	
CIFAR-10 dataset						
ResNet-50	50	16	765,098	79.85	0:38:33	-36.27
ResNet-EBCLE	26	24	830,698	**80.85**	0:24:34	
DenseNet-40	40	12	1,059,298	**87.22**	1:05:38	-62.09
DenseNet-EBCLE	16	20	692,690	86.35	0:24:53	
ResNeXt-56	56	16	11,004,864	87.08	3:38:47	-30.08
ResNeXt-EBCLE	38	16	7,449,536	**87.29**	2:32:59	
STL-10 dataset						
ResNet-50	50	16	765,386	56.56	0:16:35	-32.73
ResNet-EBCLE	26	24	838,378	**60.17**	0:12:30	
DenseNet-40	40	12	1,059,298	74.45	1:05:33	-59.50
DenseNet-EBCLE	16	20	692,690	**77.44**	0:26:33	
ResNeXt-56	56	16	11,004,864	58.25	3:25:21	-24.99
ResNeXt-EBCLE	38	16	7,449,536	**60.65**	2:34:09	
CIFAR-100 dataset						
ResNet-50	50	16	766,116	40.09	0:38:15	-34.99
ResNet-EBCLE	26	24	839,428	**48.57**	0:24:52	
DenseNet-40	40	12	1,100,428	**59.27**	1:07:17	-62.30
DenseNet-EBCLE	16	20	725,180	58.43	0:25:22	
ResNeXt-56	56	16	11,097,024	58.89	3:43:03	-29.57
ResNeXt-EBCLE	38	16	7,541,696	**60.45**	2:37:06	
ImageNet32 dataset						
ResNet-50	50	16	824,616	33.52	14:34:16	-35.67
ResNet-EBCLE	26	24	926,728	**33.57**	9:22:23	
DenseNet-40	40	12	1,511,728	**36.72**	25:43:49	-33.00
DenseNet-EBCLE	16	40	1,050,080	36.39	17:14:25	
ResNeXt-56	56	16	12,018,624	**36.32**	85:20:45	-43.21
ResNeXt-EBCLE	38	16	8,463,296	35.77	59:35:47	

Table 5.1: Table of results comparing different CNN models on various benchmarking datasets.

CNN Model	H'	χ'	Params.	Acc. (%)	Time (h:m:s)
ResNet-50	50	16	765,098	79.85	0:38:33
ResNet-EBCLE	26	24	830,698	**80.85**	0:24:34
EfficientNet-B0	1.0*	1.0*	4,020,358	71.29	0:25:31
EfficientNet-B1	1.1*	1.0*	6,525,994	75.78	1:09:05
EfficientNet-B2	1.2*	1.1*	7,715,084	75.83	1:10:25

Table 5.2: Summary table of results highlighting the relative efficacy of the ResNet models trained adopting the EBCLE heuristic and a dynamic compound scaling approach on the CIFAR-10 benchmarking dataset. EfficientNet-B3-B7 could not be evaluated due to the required memory constraints.
* H' = depth and χ' = breadth co-efficients for EfficientNet models.

respectively. These specific datasets were selected because of their similar constituent class information. All images in the CIFAR-10 dataset were upsampled to normalize pixel resolutions for equalization with the STL-10 dataset. Other models could not be evaluated due to the lack of video memory on the new commissioned hardware.

5.5.1 Statistical testing

First, the Shapiro-Wilk test for normality was used to establish if the collected raw data were normally distributed. The data were normally distributed with all p-values less than 0.05, Table 5.3 present the mean results over three experimental runs. As the distribution of data is normal, we select the parametric one-tailed paired t-test for statistical testing of the data. A one-tailed paired t-test is the most applicable since we want to question if there was an observable difference in accuracy and training time for EBCLE models on the same CNN architectures relative to deeper models. In other words, is there a statistical difference in the classification accuracies and training costs when EBCLE models are used instead of the standard CNN models.

Tests were performed with the independent variable as the CNN depth and classification accuracy as the dependent variable. The interpretation was done at the standard significance p-value threshold of 0.05 for a one-tailed test, with the assumption that

deeper models should provide higher accuracies when compared to shallower EBCLE models. The default null hypothesis is that no observable differences are present.

5.6 Our Analysis

In this chapter, conventional wisdom advocating the use of deeper CNN models (Schmidhuber, 2015) for enhancing classification accuracy has been challenged, with empirical data supporting the validity and efficacy of our proposed novel EBCLE heuristic to significantly reduce model training time without compromising model performance. Examining the input test images in Figures 5.2 and 5.3, the EBCLE models exhibit identical high-level abstractions after the last convolutional layer compared to a deeper ResNet-50 model.

The SE values, measured after the first and last convolutional layer of the ResNet-EBCLE and ResNet-50 models, as visualized in Figures 5.2 and 5.3 are 5.2735 and 5.5625 for the first convolutional layer and 5.3668 and 6.0959 for the last convolutional layer respectively. The difference is more pronounced for the MNIST dataset where the SE measures of the activation maps for the EBCLE and ResNet-50 models after the first convolutional layer are 4.9176 and 4.2341 and after the last convolutional layer are 1.9172 and 2.3010 respectively. The lower SE values in the EBCLE model indicate a higher degree of feature compression in the ResNet-EBCLE model compared to the standard ResNet-50 model with similar higher dimensional feature maps using only half as many convolutional layers thereby maintaining or even outperforming deeper networks.

In a few instances (CIFAR-100 and STL-10), the broader EBCLE models outperformed deeper models by a statistically significant margin, implying that the performance improvements in both accuracies and training times of the EBLCE models are not random. Furthermore, in all instances, the average EBCLE model training time and cost

CNN Models	p-value	Mean	Variance	(p < 0.05)?
MNIST dataset				
ResNet-EBCLE	0.38	99.44	0.0086	No
ResNet-50		99.42	0.0003	
DenseNet-EBCLE	0.14	99.60	0.0064	No
DenseNet-40		99.54	0.0020	
ResNeXt-EBCLE	0.42	99.15	0.1159	No
ResNext-56		99.21	0.0741	
CIFAR-10 dataset				
ResNet-50	0.29	80.85	5.3479	No
ResNet-EBCLE		79.85	0.4723	
DenseNet-40	0.08	86.35	0.1657	No
DenseNet-EBCLE		87.22	0.1922	
ResNeXt-56	0.32	87.29	0.8660	No
ResNext-EBCLE		87.08	0.4510	
STL-10 dataset				
ResNet-50	0.03	56.56	0.3796	**Yes**
ResNet-EBCLE		**60.17**	2.1576	
DenseNet-40	0.04	74.45	0.1657	**Yes**
DenseNet-EBCLE		**77.44**	0.1922	
ResNeXt-56	0.08	58.25	0.6175	No
ResNext-EBCLE		60.65	1.7851	
CIFAR-100 dataset				
ResNet-50	0.002	40.09	0.0409	**Yes**
ResNet-EBCLE		**48.57**	1.2637	
DenseNet-40	0.02	59.27	0.01163	**Yes**
DenseNet-EBCLE		**62.40**	0.8481	
ResNext-56	0.081	58.89	0.0134	No
ResNeXt-EBCLE		60.45	1.8388	
ImageNet-32 dataset				
ResNet-50	0.44	33.52	0.3141	No
ResNet-EBCLE		33.57	0.0007	
DenseNet-40	0.21	36.72	0.1876	No
DenseNet-EBCLE		36.43	0.0030	
ResNext-56	0.16	36.32	0.3685	No
ResNeXt-EBCLE		35.77	0.4832	

Table 5.3: Table of paired one tailed t-test results to validating the EBCLE heuristic.

reduction was **45.22%**. The reason is due to the efficient minimization in the trade-off between complexity and information gaps for EBCLE-based models. To discriminate between images related to ships and cars, simple edge detectors that can abstract salient features such as wheels, bow and stern are sufficient to achieve a high classification accuracy. Overly complex abstractions start to increase the information gap while minimizing the complexity gap causing overfitting and detrimental classification performance.

Utilizing an EBCLE model ensures sufficient dimensionality reduction has occurred before the final classification layer allowing greater fine-grained features to be learned. However, optimality in depth for any CNN model cannot be accurately determined, as asserted by authors in (K. He et al., 2016). The proposed EBCLE, at the very least, offers a mathematically sound way to justify HP choices and optimizations affecting classification performance while mitigating untrained features, a characteristic of deep models (Z. Wu et al., 2019).

A significant contribution of the EBCLE heuristic is the reduction in model training time while maintaining or outperforming baseline classification performance, inline with wider residual network architectures (Zagoruyko & Komodakis, 2016). Other compression, quantization or pruning methods discussed in Section 5.2.2 are accompanied by a statistically significant decrease in classification performance and thus are not studied extensively in this chapter.

5.6.1 Exponential increase in trainable parameters leads to marginal gains in performance

The number of trainable parameters increases exponentially for additional convolutional layers, as there is a $2^{\chi'}$ increment in convolutional kernels/units in the model to compensate for the reduction of model capacity (Mallat, 2016). As gradients are in

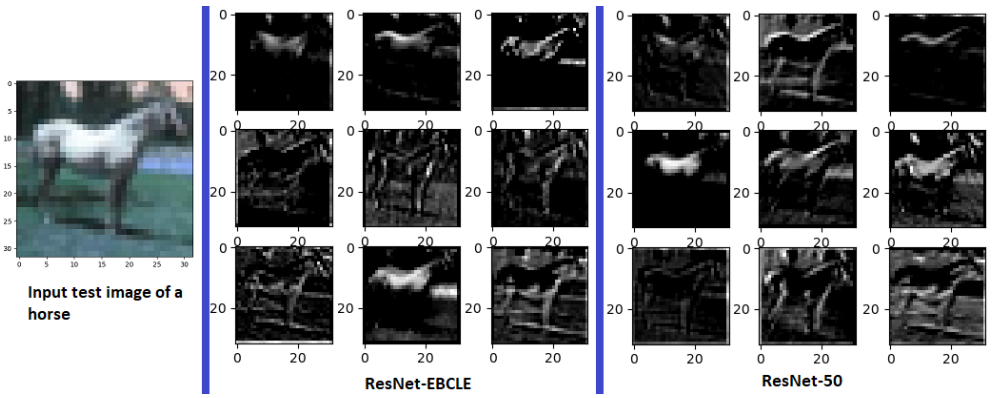

Figure 5.2: Feature/Activation maps visualized after the first convolutional layer for a test image of a horse in the CIFAR-10 dataset, EBCLE depth = 26

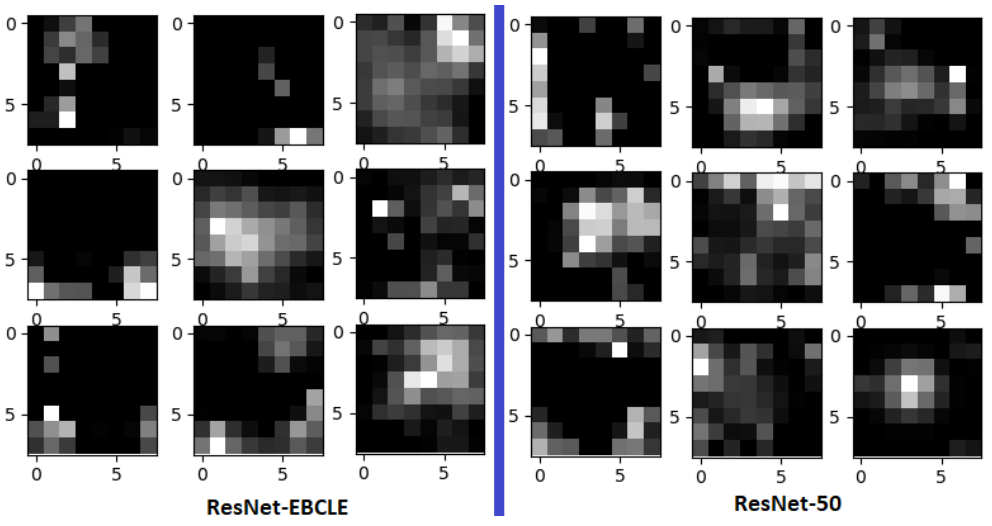

Figure 5.3: Feature/Activation maps visualized after the last convolutional layer for a test image of a horse in the CIFAR-10 dataset, EBCLE depth = 26

the direction of the steepest descent in back-propagation (Veit et al., 2016), utilizing unnecessarily deep networks will lead to untrained features (Z. Wu et al., 2019). The EBCLE heuristic presented in Section 5.13 provides an adequate depth estimation using Shannon's entropy for measuring the theoretical limit for feature compression by the convolutional layers after which feature representations resolve into identity transformations which are ineffective in enhancing model performance. Further credence for shallower yet wider models is provided by the data presented in (H. Zhao et al., 2017), where a classification improvement of 1.1% was achieved from an 117 additional depth increase.

Training CNN models by varying the depths and widths on the same CIFAR-10 dataset while keeping all other HyperParameters constant resulted in the EBCLE model outperforming deeper models. It is noteworthy to mention that additional increases in the initial convolutional width (χ') caused overfitting at extremely large values (256) and resulted in decreased classification performance. Moderate values of χ' (128) produced the best classification accuracy (with an increment of 0.75%) but resulted in an exponential increase in the number of trainable parameters. In other words, models with a χ' value of 128 had 23,493,130 number of trainable parameters compared to a narrower model with a χ value of 24 with only 830,698 i.e. a 96.46% decrease in the number of trainable parameters resulted in only a 0.75% decrease in classification performance. This decrease in the number of parameters suggests excessive model width increases do not offer huge improvements in classification performance similar to very deep models.

Counter-intuitive decrease in model training time

A marked increase in the time required to compute gradients for some EBCLE models due to increase in the total number of parameters can be witnessed relative to baseline models. However, since most of the model training time is consumed during the feature

compression phase (discussed in Section 5.3.1), EBCLE-based models are inherently restricted in terms of their feature compressibility and as such a corresponding decrease in the solution space with yields the observed training time reductions. The increased breadth of the EBCLE-models enables optimal utilization of the computing hardware due to enhanced data loading and parallel processing, relative to deeper networks which experience frequent information processing bottlenecks.

Results presented in Tables 5.1 and 5.2 indicate that EBCLE-based models show slight to significant increases in model performance even with decrease in model parameters due to more effective feature extraction, abstraction and compression relative to baseline models. As discussed earlier in Section 5.6.1, exponential increases in model parameters lead to only marginal gains in performance. As such, more effective training regimes provide significant performance gains compared to simply increasing model sizes. This is due to the tendency of deeper layers to resolve into identity transformations.

5.6.2 Limitations

A key limitation for employing EBCLE is that, the heuristic is limited in applications where the entropic variance of constituent classes in the input dataset is high, as there are no practical ways to determine relative effective variances for individual classes to optimize feature compression. In other words, if some of the constituent classes in the input dataset have high entropy and others have low entropy, EBCLE would not be applicable since mean entropic measures are utilized in this chapter.

Another limitation for a comprehensive evaluation of the EBCLE heuristic presented in this chapter is that, although the principles of optimizing feature compression should hold true for different application domains or tasks such as audio classification or segmentation; empirical evidence is critical in drawing any meaningful conclusions and

as such EBCLE remains confined to CNN image classification in this article.

5.7 Conclusion and Future Work

To overcome the problems posed by severe over-parameterization concerning model training time and architecture selection, we proposed an entropy-based heuristic that imposes feature abstraction and compression restricting over-parameterization with regards to convolutional depth in CNNs. The proposed heuristic employs a priori knowledge of data distribution for the input dataset to simplify and accelerate CNN model training. Using the EBCLE heuristic, we provided empirical evidence utilizing several well-known benchmarking datasets and CNN architectures against established baselines to validate the efficacy of EBCLE-based models with respect to training time and classification accuracy. Results for the EBCLE heuristic adopting a shallow yet broad CNN model indicate a 24.99% - 78.59% reduction in model training time compared to deeper yet narrower CNN models for the same HyperParameter (HP) configurations without significant performance degradation.

The results presented in this chapter support the independent findings obtained in (Z. Wu et al., 2019), where the authors assert that wider, yet shallower models outperform deeper, yet narrower CNN models. Furthermore, the authors in (J. Zhao, Liang, Dong, Tang & Liu, 2020) establish both theoretically and empirically that entropy-based heuristics can simplify and accelerate inner and outer loop calculations for feature selection. Additional validation for our EBCLE heuristic regrading forced feature abstraction and compression can be corroborated by the findings presented in (H. Zhao et al., 2017), where the authors establish experimentally, that shallower CNN models can learn the same functional representations as deeper networks.

Our proposed EBCLE heuristic offers a simplified approach to select CNN architectures and accelerate model training by utilizing the a priori entropy of input the dataset.

Additionally, our EBCLE heuristic is architecturally agnostic facilitating application in multiple domains. The empirical data presented in this chapter allude to the same phenomenon of over-parameterization for convolutional widths χ', suggesting further gains could be achieved with regards to decreasing model training times. This is an important area for exploration and future publications. Empirical validation for our proposed EBCLE heuristic is conducted on five benchmarking datasets (ImageNet32, CIFAR-10/100, STL-10, MNIST) and three network architectures (DenseNet, ResNet, ResNeXt) along with a dynamically scaling network architecture (EfficientNet).

Wider but shallower residual networks have shown to outperform narrow yet deeper networks (Zagoruyko & Komodakis, 2016), corroborating the findings presented in this chapter. Furthermore, very deep CNN architectures resolve into a collection of independent feature extractors making the process of feature extraction redundant since skip connections facilitate only the most important features to be captured (Z. Wu et al., 2019). The EBCLE heuristic could be employed to introduce forced feature abstraction and compression enhancing the efficiency of model training. Empirical evidence supports the fact that shallow residual networks can learn the same functional representations as deeper networks (H. Zhao et al., 2017), providing further independent validation that the EBCLE heuristic could help optimize model training, in terms of addressing severe over-parameterization with regards to training time and simplified CNN model selection.

Finally, the theory behind EBCLE for CNN architectures supports the fact that the same principles governing feature compression should apply to other deep learning tasks such as segmentation or regression but, empirical evidence is essential to draw any meaningful conclusions and as such it is reserved as future work.

Through this chapter, we have gained a more comprehensive understanding of the feature extraction process in CNNs and investigated the effects of over-parameterization on model training times. In the next chapter (Chapter 6), we explore a novel method of

automatically tuning the model learning rate HyperParameter to further reduce model training times by increasing the rate of model convergence.

Chapter 6

Entropy-Based Inner-Loop Optimization

In the previous chapter (Chapter 5), we verified that over-parameterized CNN models could be compressed without significant adverse impacts on classification performance, due to their informational and topological sparsity. An ongoing problem in CNN training is the vast number of HyperParameters(HPs) that need to be optimized which can greatly affect model classification performance.

In this chapter, we focus on inner-loop fine-tuning of the Learning Rate (LR) HP. Optimizing the LR will accelerate the rate of model convergence and decrease model training time. We propose the use of Maximum Entropy (ME) measures as a monitoring function to assess variational drifts in the CNN feature extraction properties, thus reducing training complexity by offering an entropy-based optimization algorithm.

6.1 Introduction

Machine learning algorithms have numerous HyperParameters (HPs) that are critical for model training influencing the effectiveness of model learning. HPs need to be adjusted

either manually or automatically to ensure model convergence for high classification accuracies. Manual tuning of HPs is time-consuming and often requires expert domain knowledge, whereas automatic tuning might lead to sub-optimal HP configurations. HP tuning unnecessarily increases the complexity of model training and is often considered as a nuisance (Snoek, Larochelle & Adams, 2012). Most machine learning algorithms aim to update a weight matrix ω representing the trainable parameters of a Neural Network (NN) such that an objective function $f(\omega)$ is optimized for a given task. The weight updates $\Delta\omega$ are performed after each iteration based on a simple rule, $\omega_{t+1} = \omega_t + \Delta\omega_t$, where t is the iteration index (Robbins & Monro, 1951). Calculating $\Delta\omega_t$ using Stochastic Gradient Descent (SGD) is given in Equation 6.1.

$$\Delta\omega_t = -\eta\nabla_\omega f[\omega_t, \mathbf{x}_t] \tag{6.1}$$

Where, η is the learning rate which dictates the magnitude of weight updates and ∇_ω is the gradient of the weight matrix optimized for the objective function at iteration t using the input data vector $\mathbf{x}_t$. The SGD algorithm is computationally efficient since the weight updates are calculated using only the first-order partial derivatives, which is computationally equivalent to a function evaluation. Adjusting HPs often requires a time-consuming process of experimentation using trial and error, manual fine-tuning or random searches of the HP space (Bergstra & Bengio, 2012), which are computationally expensive and imprecise. The sensitivity of optimizers to learning rates is the key determining factor for network convergence. It is well established that excessively high learning rates cause model divergence, whereas lower learning rates induce unusually slower rates of convergence (Bengio, 2012). Adding to the selection of an optimizer, the determination of an optimal learning rate η is essential for faster and more accurate convergences in NNs.

In this chapter, we propose a **M**aximum **E**ntropy-Based **L**earning Ra**T**e **E**nhance**R**

(MELTER). MELTER is a real-time adaptive learning rate scheduler to improve the SGD optimizer utilizing Maximum Entropy (ME) measures of the input training data in conjunction with the ME measures of the convolutional kernels during model training, detailed further in Section 6.2.5. We limit the scope of this research to computer vision and Convolutional Neural Network (CNN) model training as they are state of the art for image recognition/classification (Simonyan & Zisserman, 2014) and stochastic optimizers, specifically Residual Networks and SGD, as they are most commonly used by computer vision practitioners. The proposed MELTER algorithm uses a threshold to reset the learning rates periodically for enhanced exploration of the domain/solution space using the input data vector, significantly extending the work done in (Loshchilov & Hutter, 2016).

Empirical validation of MELTER on well known benchmarking datasets such as MNIST, CIFAR-10/100 and STL-10 establishes superior performance compared to ResNet models trained using manually-tuned learning rates and other existing tuning-free optimization methods. The proposed MELTER solution as Algorithm 6.3.1, is noise averse in terms of image distortions, requires no pre-training or manual fine-tuning and is applicable across multiple datasets. We observe an improvement of **2.67%** and **2.06%** for the ResNet-20 and ResNet-56 CNN models respectively compared against a piecewise constant training method (Rolinek & Martius, 2018). On the MNIST dataset MELTER achieved state-of-the-art accuracy of **99.75%** using the ResNet-32 model without real-time data augmentation, outperforming all other training regimes compared in this chapter (Smith & Topin, 2019; Smith, 2017; Snoek et al., 2012; Vaswani et al., 2019; Rolinek & Martius, 2018). On the CIFAR-100 and STL-10 datasets, MELTER outperformed manually tuned models by **5.58%** and **2.77%** respectively.

6.2 Background and Related Work

Calculating optimal Learning Rates (LR) through the use of a Hessian matrix of second-order partial derivatives (Yu, Efe & Kaynak, 2002) guarantees model convergence, but the required computational resources for calculating such matrices prove impractical. Due to these immense computational limitations for calculating the entire Hessian matrices, proposals for optimization methods using only the first-order information for approximations are widely adopted (S. Becker, Le Cun et al., 1988; Duchi, Hazan & Singer, 2011). Sub-optimal LRs either induce overfitting when the rate is small or conversely promote model divergence when the LR is high. Conducting grid searches (Smith, 2018) or line searches (Vaswani et al., 2019) prior to LR initialization can help alleviate the problem of sub-optimal LR calculations, but these methods are time-consuming and precision cannot be guaranteed.

6.2.1 Cyclical Learning Rate Optimization

A Cyclical Learning Rate (CLR) solution (Smith, 2017) using bounded minimum and maximums with a stepsize to form cycles of linearly increasing and decreasing LRs up to the bounded values, assists in broader solution space explorations aiding in model convergence to the global optimum. The maximum bound value could be determined by performing a range test by pre-training a model with an initial small LR and linearly increasing it to establish the maximum LR realized through model divergence. Similarly, the minimum bound can be established by adopting a trial and error methodology using a factor of 3 to 4 less than the maximum LR and analyzing model convergence. Our proposed MELTER algorithm incorporates similar bounded minimum and maximum values for LR, but eliminates the need for trial and error methods, pre-training or range tests by utilizing the a priori knowledge of the entropical data distributions for the input dataset.

6.2.2 Super-Convergence and 1Cycle Optimization

Super-Convergence is a phenomenon witnessed in CIFAR-10/100 datasets by (Smith & Topin, 2019) for deep ResNets where the test loss and accuracy remain relatively constant for large LRs. The phenomenon was used to propose a 1Cycle policy by slightly modifying the cyclical learning rate scheduling using different HPs. The authors claim of achieving superior performance with orders of magnitude fewer iterations of model training seem promising; however, the reproducibility of these methods is not consistent or independently validated. The principle of Super-Convergence also contradicts the assertion made by (Hoffer, Hubara & Soudry, 2017) that training models for longer improves generalization. Furthermore, the effect of Super-Convergence was only witnessed in specific datasets such as CIFAR-10/100. Our proposed MELTER algorithm does not have such specific requirements and can be adopted for a wider range of datasets.

6.2.3 Linearized Loss-Based Optimal Learning Rate Optimization

Linearized Loss-based optimaL Learning rate (L^4) (Rolinek & Martius, 2018) optimizes the step-size during model training to achieve a minimum attainable loss. The results presented achieve high accuracies in relatively short periods of time during model training. However, achieving these high accuracies is predicated upon the introduction and optimization of several additional HPs, requiring further manual or automatic fine-tuning. Our proposed MELTER algorithm achieves superior performance over L^4 in the same number of iterations/epochs on the same datasets without the introduction of additional HPs avoiding requirements for further fine-tuning or optimizations.

6.2.4 Energy-Based Optimization

An energy-based solution for learning rate scheduling (Battiti, 1989) starts with an initial learning rate and employs a specified energy function $\mathcal{E}(\omega)$ for monitoring weight updates. The learning rates are varied proportionally by an arbitrary factor inverse to the functional evaluation result of $\mathcal{E}(\omega)$. Searches are made on the negative gradient to find a step that decreases the energy function, and a step is guaranteed if learning rates are allowed to approach zero. While this method accelerates weight updates, the correctness of weight updates cannot be guaranteed if the input is noisy. Correctness cannot be guaranteed because noisy inputs result in non-steady decreases of the energy function, leading to sub-optimal convergences to local optima, causing performance degradation. In our proposed MELTER algorithm, a monitoring function based on ME measures is used instead which makes the approach robust and resilient to noise, since calculation of ME is independent of noise and relies extensively on the amount of disorder of the data. Table 6.1 summarizes existing optimization methods with MELTER.

Method	Noise Averse	Datasets	Pre-training	Manual fine-tuning
Grid (Smith, 2018) & Line searches (Vaswani et al., 2019)	No	Multiple	Yes	No
CLR (Smith, 2017)	No	Multiple	No	Yes
Super-Convergence (Smith & Topin, 2019)	No	CIFAR-10/100	Yes	No
1Cycle (Smith & Topin, 2019)	No	Multiple	Yes	No
L^4 (Rolinek & Martius, 2018)	No	Multiple	No	No
Energy (Battiti, 1989)	Yes	Multiple	No	Yes
MELTER	Yes	Multiple	No	No

Table 6.1: Summary table comparing Existing LR optimization methods with MELTER

6.2.5 Convolution Kernel Analysis and Maximum Entropy (ME)

Data transformation is of vital importance in the domain of machine learning, the aim of which is to reduce higher dimensional data into lower dimensional linearly separable representations of the data. In CNN models, feature extraction and data transformation are performed through convolutional kernels, where linear input vectors are reduced to lower-dimensional linearly separable weight matrices through non-linear activation functions.

Kernel analysis is widely used in different application domains (Jenssen, 2009; Collins & Duffy, 2002) but currently, it is not utilized for real-time CNN model optimization. As feature extraction in any CNN is imperfect, the convolution kernel weights are updated based on noisy or inadequate feature information from the input vectors. A solution to counteract the noisy or insufficient feature information data is to utilize ME measures, which are used widely for image reconstruction from noisy or deficient data.

In this chapter, we calculate the ME measures of convolutional kernels after every epoch during model training and employ the entropic difference between the training data and convolutional kernels to dynamically adjust learning rates. The use of entropic differences as monitoring functions has precedent in correcting instrumentation drifts and re-calibration of poorly known parameters (Skilling & Bryan, 1984) in different application domains. The use of ME for kernel analysis is further justified since ME is the only consistent way of selecting a single discrete data point from a given set of data which best fits the data, proven axiomatically in (Shore & Johnson, 1980; Johnson & Shore, 1983).

Theoretical calculations of ME using the equations proposed by Hartley (Hartley, 1928) is impractical to calculate in CNNs for image classification since the relative probabilities of image pixels in a given neighborhood causes state-space problems for

current computing hardware. To counteract this problem, an open-source scikit-image processing library written in python could be used to calculate the ME measures for color and grayscale images employing disk of radius r scanned across the image which returns the color level histograms for the given bins. Frequency measures could then be calculated to determine relative probabilities.

The ME measures for color images require independent ME calculations for each of the Red (R), Green (G) and Blue (B) color channels which are then averaged to get the final ME measure for a single image. The mean ME measure for the entire training set is computed using individual ME scores for the training images which is then rounded to two digits to secure the final reported ME measure for the dataset. The mean ME measures for *MNIST, CIFAR-10, CIFAR-100* and *STL-10* datasets are *1.04, 2.2, 2.18* and *2.15* bits per pixel respectively. The amount of time taken to calculate the ME measures is insignificant, since the ME calculation script can be executed in parallel on the CPU, while CNN model training occurs on the GPU.

6.3 Maximum Entropy-Based Learning RaTe EnhanceR (MELTER)

In Equation 6.1, assume that the gradients for ω are computed using a function $g(\mathbf{x}_i)$ for an input vector $\mathbf{x}_i$. Therefore, Equation 6.1 can now be reduced as Equation 6.2,

$$\Delta\omega_{i_t} = -\eta g(\mathbf{x}_i) \tag{6.2}$$

Now, assume an exaggerated scenario that the learning rate η keeps decreasing every epoch and that an epoch has only a set of ten mini-batch vector inputs. An online deep learning algorithm which has no a priori knowledge of the input feature vector space can determine a finite number of features, $f_1(\theta), f_2(\theta), ..., f_n(\theta)$. The goal of

a deep learning algorithm is to perform weight updates to adjust the available model parameters such that θ can ideally linearly separate the input vector space.

Learning rate η is paramount in accurate separation of the linear space since incorrect updates can fail to separate all of the feature vectors and might cause premature convergence. Assume that η is initialized with a high value which is able to classify only half of the features. Progressing ten epochs (based on the exaggerated scenario), without modifying η, the direction and magnitude of the gradients and weight updates for $\Delta\omega_{i_t}$ can be disregarded as the model would diverge. Therefore, we can imply that at this point in time, the CNN has ceased learning either due to insignificant weight updates or convergence to a local optima. A decreasing learning rate/decay over time can mitigate premature model convergence and, given an infinite model training time, a true convergence using stochastic optimizers like SGD and ADAM (L. Luo, Xiong, Liu & Sun, 2019; Kingma & Ba, 2014) could achieve a general convergence. Problems arise when model learning is constrained to a local minima or saddle points with insignificant weight updates. In these circumstances a decreasing learning rate is detrimental to performance and could implicitly lead to premature model convergence.

Premature convergence can be mitigated but not completely eliminated using resets and threshold values. ResNets ensure that if model learning is constrained to a local optima, a higher learning rate for the next epoch would force significant gradient and weight updates imposing a search for other minima in the input vector space. During this step, however model accuracy will fall significantly but could recover quickly since the learning rates keep decreasing linearly. Thresholds ensure that the model does not fall into saddle points prematurely, allowing for persistent learning. Dynamic LR schedulers discussed in Section 6.2 can mitigate the problem of premature convergences by increasing and decreasing the learning rates automatically but are limited as they cannot accurately account for noisy or insufficient data in the input data or feature extractions requiring time-consuming manual optimizations or pre-training.

Using Maximum Entropy (ME) (as discussed in Section 6.2.5) for re-calibrating any variational drifts caused by structural noise during model training in the next epoch, utilizing the entropic difference Δ for the convolutional kernels and input image dataset allows a robust monitoring function to automatically update learning rates after each epoch, which theoretically can search for the best fit for the given data, proven axiomatically in (Shore & Johnson, 1980).

Assume that the ME of the input dataset at any given instance of time τ is $\delta_{dataset}$ and the kernel entropy is δ_{kernel}. Δ is calculated as $\Delta = |\delta_{dataset} - \delta_{kernel}|$ which can now be employed to monitor variational drifts during model learning. If Δ increases, it would suggest that the model might be stuck in a local minima since the number of micro-states γ that the kernel weights can exist in has reduced. A decrease in Δ would imply that the model is converging to the global minima.

To clarify further, the number of micro-states of a randomly initialized convolutional kernel can exist in is γ. To accurately reproduce the same amount of information as the input image using the kernel would require the ME of all the γ bits be equal to $\delta_{dataset}$ in other words, $\delta_{kernel} = \text{ME}(\gamma) \geq \delta_{dataset}$. During model training, some of the micro-states γ exists in need to be dimensionally reduced for a linear separation of the input space, but, $\delta_{kernel} = \text{ME}(\gamma) \geq \delta_{dataset}$. Therefore, monitoring Δ during model training would provide a more accurate and robust method of dynamically optimizing LRs, since LR invariably affects weight updates. Adjusting LRs significantly within the pre-defined bounds computed through a priori data distribution of the input dataset enables a faster and comprehensive exploration of the solution space. As discussed in Section 6.2, the upper and lower bounds are critical in ensuring efficient model convergence; surprisingly, the MELTER algorithm performs competitively even with unoptimized bounds for different datasets.

6.3.1 Algorithm

Algorithm 1 presents the pseudo-code for our proposed dynamic LR scheduler, MELTER. The primary objective is to update the global learning rate variable η_{MELTER} discussed in Equation 6.2 for an SGD optimizer. The MELTER algorithm has four inputs, total number of training epochs κ_{total}, the input dataset $\mathbf{X}$ as a set of images with vectors $\mathbf{x}_i \in \mathbf{X}$, the lower bound β and reset value ξ. Lower bound is determined using the ME of $\mathbf{X}$ i.e. SGD performs well for LRs less than 1, therefore the lower bound β is set as $0.5e^{-ceil(ME(X))}$ and the reset value ξ as $1e^{-1}$. The reset value could be adjusted to any arbitrary value which would increase the search space but does not considerably affect model performance. Tests for MELTER adopting reset values of 1 showed similar results.

Threshold resets using an epoch count variable κ and bounded values allow for a comprehensive search space exploration. While the MELTER algorithm utilizes threshold resets as a multiplicative factor, adopting a logarithmic function provided similar results but negatively affected rates of convergence. The algorithm outputs learning rates as η_{MELTER} for the next training epoch i.e. $\kappa_{current}$ + 1 using the current epoch's model parameters $\kappa_{current}$.

The ME calculation function $E(y_i)$ is used to calculate the ME measures for the convolution kernels where $y_i = \langle y_1, y_2, ..., y_n \rangle$ as discussed in Section 6.2.5. The function $\omega(t)$ returns the convolutional kernel weight matrix for an input data vector $\mathbf{x}_i = \langle \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \rangle$ at an instance of time τ. The algorithm sets variables ω with the kernel weights and μ represents the ME measure for the set of finite χ' convolutional kernel weight vectors computed as $E(\omega(\tau))$ for an instance of time τ such that $\omega = \langle \omega_1, \omega_2, ..., \omega_m \rangle$. The algorithm is called every epoch up to the total number of epochs κ_{total} while each step count is changed using the variable κ.

Algorithm 1 Pseudo-Code for the proposed **M**aximum **E**ntropy-Based **L**earning Ra**T**e **E**nhance**R** (MELTER)

Input: κ_{total}, $\mathbf{X}$, ξ, β
Output: Learning Rate, η_{MELTER}
Initialize: $\Lambda \leftarrow E(\mathbf{x}_i)$, $\kappa_{current} \leftarrow 1$, $\kappa \leftarrow 1$
while $\kappa_{current} \neq \kappa_{total}$ **do**
  $\omega \leftarrow \omega(\tau)$
  $\mu \leftarrow E(\omega)$
  $\Delta \leftarrow (|\Lambda - \mu|)$
  $\eta_{MELTER} \leftarrow (\eta_{MELTER} \times (1/\Delta))/\kappa$

  if $\eta_{MELTER} < \beta$ **then**
    $\eta_{MELTER} \leftarrow \xi$
    $\kappa \leftarrow 0$
  end if
  $\kappa \leftarrow \kappa + 1$
  $\kappa_{current} \leftarrow \kappa_{current} + 1$
  return η_{MELTER}
end while

6.3.2 Contribution

Others (X. Wu, Ward & Bottou, 2018) assert that, in a stochastic setting, such as CNN model training for image classification, there are no clear best choices where convergence can be guaranteed in practice. Computer vision practitioners often test multiple different learning rate schedules/schedulers using a trial-and-error approach for different tasks, which is time-consuming and requires expert domain knowledge. The MELTER solution proposed as Algorithm 1 in Section 6.3.1 addresses this limitation and offers an entropy-based optimization of CNNs for different tasks reducing the complexity involved in model training. MELTER achieves competitive results against manually tuned or existing tuning-free approaches presented in Section 6.5.

6.4 Experimental setup

Experimentation revolved around testing the efficacy of MELTER as an entropy-based inner-loop optimizer for dynamic learning rate scheduling. A quantitative methodology was employed to collect the test-set classification accuracies for four well-known benchmarking datasets such as MNIST (LeCun et al., 1998), CIFAR-10/100 (Krizhevsky & Hinton, 2009) and STL-10 (Coates et al., 2011). Our hypothesis is that using an entropy-based dynamic and adaptive learning rate scheduler presented as MELTER as Algorithm 1 will provide automatic LR optimization with classification performance outperforming existing approaches. The HPs are in-line with the compared literature and any changes made are mentioned in Section 6.5. Due to limitations of the computing hardware used, the batch size of 128 was adopted, similar to authors in (K. He et al., 2016) and could not be increased without incurring a memory penalties. Slight variations in performance are expected and can be attributed in due part to the loss in precision (float64 to uint8) of the library/hardware function used to calculate the ME measures of convolutional kernels.

6.5 Results

Preliminary experimentation for this chapter was carried out using a single Tesla P100 GPU with 12GB of VRAM and the results presented in this chapter were carried out with a single RTX 2080ti with 11GB of VRAM, generously provided by the New Zealand eScience Infrastructure (NeSI) and InfuseAI Limited (New Zealand) respectively. The software framework consists of scripts written in python which made use of Keras and a tensorflow backend. All experiments were carried out in three separate instances with the reported results being averaged across all the experimental runs to remove any biases in the data, presented in Tables 6.2-6.4.

LR method	Acc. (%)	Epochs	Data Aug.
ResNet-20			
Piecewise Const. (Rolinek & Martius, 2018)	88.6	N/A	N/A
CLR (Smith, 2017)	90.4	N/A	N/A
Manually Tuned	90.65	200	True
MELTER	**91.27**	200	**True**
MELTER w/out ME$\diamond$	90.68	200	True
MELTER	**89.02**	200	**False**
GP EI MCMC (Coates & Ng, 2011)	85.02	N/A	False
ResNet-56			
1Cycle (Smith & Topin, 2019)	92.1*	N/A	N/A
Piecewise Const. (Rolinek & Martius, 2018)	91.2	205	N/A
Super-Convergence (Smith & Topin, 2019)	92.4*	25	N/A
Manually Tuned	**93.48**	200	True
Manually Tuned	88.02	200	False
MELTER	92.65	200	True
MELTER	**90.46**	200	**False**
MELTER w/out ME $\diamond$	91.37	200	True

Table 6.2: Table of results comparing CNN models trained using the MELTER LR and Existing LR methods on the CIFAR-10 dataset
* independent reproduction of the results are inconsistent,
$\diamond$ a small constant decay is used instead of ME calculations

The experimental results are restricted to small batch datasets to derive comparative studies; larger datasets were not considered due to a lack of accurate comparative studies in existing literature. Table 6.2 highlights the relative improvement of our MELTER algorithm over existing techniques for ResNet-20 models on the CIFAR-10 dataset. While the MELTER algorithm does not provide a large uptick in classification performance, but the main aim of MELTER is to reduce model training complexity without requiring time-consuming manual fine-tuning. A similar observation can be witnessed for ResNet-56 models on the CIFAR-10 dataset.

Interestingly, in Table 6.4, on the MNIST dataset, the MELTER algorithm exhibits superior performance at very-low epochs compared to the L^4 and manually tuned ResNet-32 models without real-time data augmentation. Extending model training-time

LR method	Acc. (%)	Epochs	Data Aug.
L^4	98.5	30	unknown
MELTER	**99.63**	**30**	**False**
Manually Tuned	99.44	30	False
Manually Tuned	99.30	200	True
Manually Tuned	99.40	200	False
MELTER	**99.71** ± 0.04	200	**False**
MELTER	99.39	200	True

Table 6.3: Table of results comparing ResNet-32 CNN models trained using the MELTER LR and Existing LR methods on the MNIST dataset

LR method	Acc. (%)	Epochs	Data Aug.
CIFAR-100			
MELTER	66.07	200	False
Manually tuned	60.49	200	False
MELTER	**69.75**	200	**True**
Manually tuned	69.20	200	True
STL-10			
MELTER	68.68	200	False
MELTER	**80.61** ± 0.85	200	**True**
Manually tuned	68.55	200	False
Manually tuned	78.69	200	True

Table 6.4: Table of results comparing ResNet-32 CNN models trained using the MELTER LR and Existing LR methods on the CIFAR-100 and STL-10 dataset

to 200 epochs provides a top-10 state-of-the-art classification performance of 99.75%. Furthermore, for all of our experimentation, the absence of the ME monitoring function degraded model performance even with the introduction of a small constant decay and threshold resets, which provides credence in establishing the validity of MELTER. Furthermore, criticism around the gains in classification performance achieved through decays can be extinguished by observing the results in Table 6.2, where the ME monitoring function played an important role in increasing model performance.

6.5.1 Data Augmentation

In cases where optimal regret minimization is achieved, MELTER might find it difficult to optimize or even in some cases, impede the proper domain/solution space exploration. This impedance can be attributed to restricted semantic information extraction and propagation in the convolutional kernels leading to improper Δ calculations. An example of this is evident in MELTER's performance on the STL-10 dataset which shows a statistically significant improvement when real-time data augmentations are disabled compared to existing optimization methods even at low model training iterations of ten-thousand. However, there were no significant classification performance gains when real-time data augmentation was enabled in all instances primarily due to the fact that a priori entropy measures of the input images were calculated on the unaugmented datasets, providing justification and validity for our hypothesis that using entropy measures is an accurate and robust optimization method for model HP optimization.

It is worthy of mentioning that on a cursory evaluation, the rate of convergence of MELTER models was significantly faster than manually tuned models and comparable to super-convergence and 1Cycle methods. A thorough study of this enhanced rate of convergence will be explored in future iterations of the thesis. Therefore, for prototyping or fast evaluation of classification performance on novel datasets or CNN architectures, MELTER would be an ideal choice since it can easily be reproduced. The MELTER algorithm is designed to explore and exploit any variational drifts in the convolutional kernel data before penalizing model learning to ensure domain/solution space resilience against convolutional kernel learning saturation. Furthermore, MELTER removes the arbitrary choices made by deep learning practitioners in deciding learning rates for CNN model training and can resist sub-optimal configurations as evidenced in Tables 6.2-6.4.

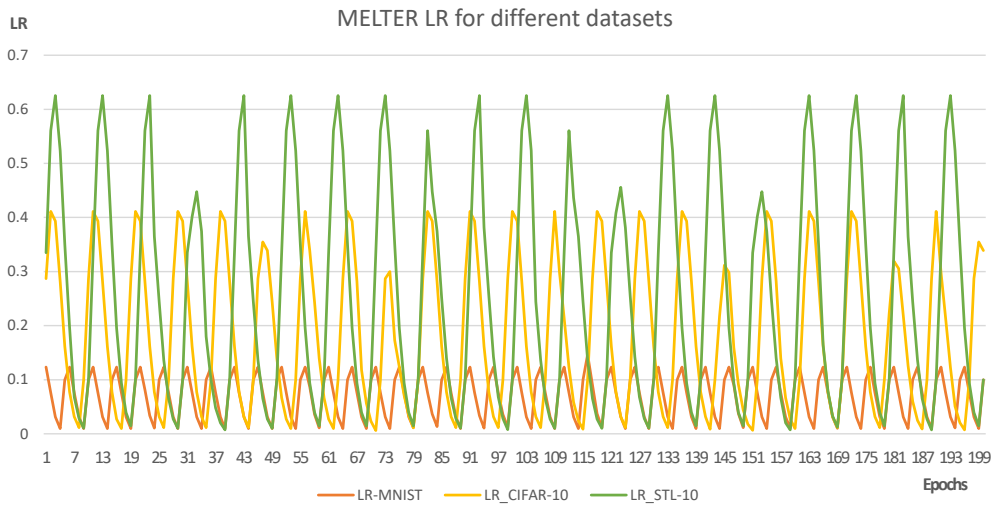

Figure 6.1: Automatically adjusted LR for a ResNet-32 CNN model using the MELTER algorithm for different datasets without real-time data augmentations.

6.5.2 Analysis

Analyzing the data presented in Tables 6.2-6.4, we can conclude that MELTER outperforms existing optimization techniques without the need for manual fine-tuning or the introduction of additional HPs. Furthermore, Figure 6.1 illustrates the automatic LR adjustments made by the MELTER algorithm and the extent of a more comprehensive domain/solution space exploration. Figure 6.2 visualizes the convolutional kernel entropy measures in real-time during model training, highlighting the change in information extracted by the convolutional kernels over time.

There is a clear distinction in the variation of entropic distribution for different datasets. The MNIST dataset for example, shows relatively fewer declines in entropy measures, suggesting optimal minimization is easily reached and thus implying the effectiveness of model learning capacity. The automatic fine-tuning ability of the MELTER algorithm also implicitly provides some regularization but its impact is limited when accurate a priori data distribution of the input dataset cannot be established, subsequently inducing improper LR calculations. Re-calculating and re-initializing

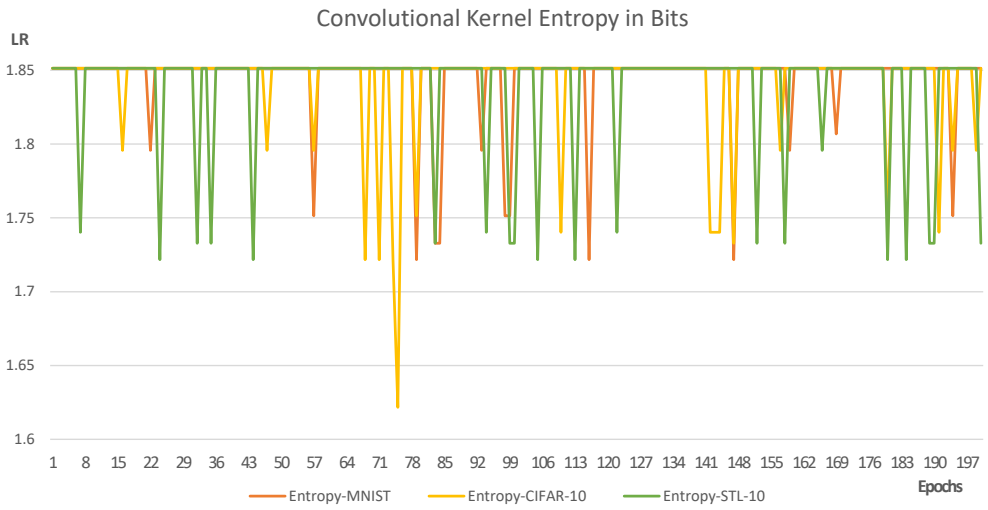

Figure 6.2: The averaged convolutional kernel entropy values for the ResNet-32 CNN model across the total number of kernels used during model training on different datasets.

the convolutional kernel weights using the a priori data distribution of the augmented dataset should increase model performance but since these augmentations are performed in real-time it falls outside the scope of this chapter/thesis.

Figure 6.1 highlights the different LR adjustments required for different entropic distributions of the input dataset. Figure 6.1 also alludes to progress in model learning by mitigating any adverse convergences to saddle points due to periodic high LR values, which should constrain premature convergence. An interesting observation evident from Figure 6.1 is that the LR bounds are unique for each of the benchmark datasets.

Relatively less complex datasets such as MNIST have a shallow band of LRs throughout the learning process, which is in stark contrast to more complex datasets such as CIFAR-10/100, STL-10, where there are greater variations in the learning cycles during model training. Evidence of slight divergence in solution space exploration and subsequently the enhancement of learning rates can be witnessed in Tables 6.2 - 6.4. Although the MELTER algorithm deviates in performance gains compared with an unaugmented dataset, final classification performance is competitive with existing

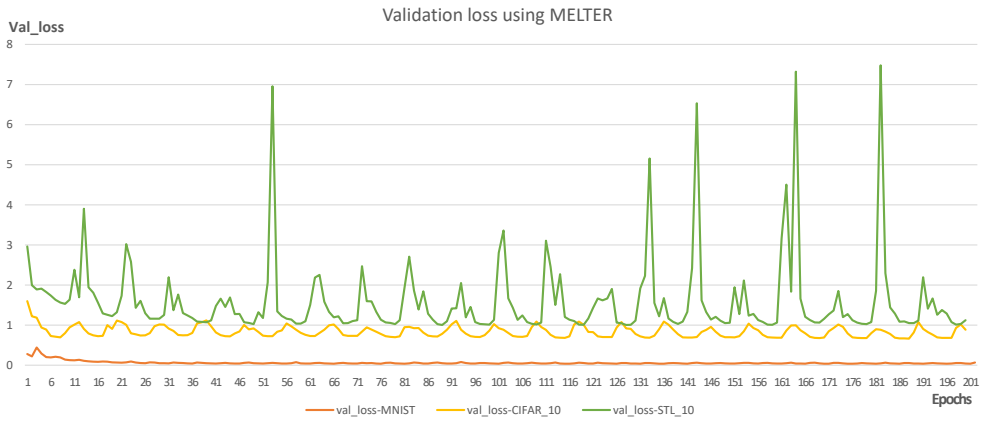

Figure 6.3: The validation loss of the ResNet-32 CNN models using the MELTER algorithms on MNIST, CIFAR-10 and STL-10 dataset without real-time data augmentations.

methods as evidenced by the results presented in Tables 6.2 - 6.4 where no significant compromises are observed.

Figure 6.3 shows periodic negative validation loss spikes for some datasets indicating the model's retreat from a saddle point, but as training progresses, the model recovers quickly and reconvenes to additional local optima, indicating a through exploration of the solution space. Analyzing Figure 6.3 also demonstrates the intuitive knowledge that higher-dimensional complex data have multiple saddle points (STL-10 and CIFAR-10) compared to simplistic datasets (MNIST), for which the CNN can easily converge to the global optimum.

6.5.3 Limitations

The primary limitation of this chapter is that the experimentation is limited in evaluating the efficacy of MELTER for only image classification/recognition. While the authors are confident that MELTER will provide the same if not better results on other NN domain problems such as natural language processing or audio/speech classification, the results need to be experimentally validated independently.

Maximum Entropy (ME) measures of the training data and real-time convolutional kernel data vary only when there is sufficient change in the underlying convolutional kernel weight data. While this invariability in kernel data offers greater stability and resistance to noisy or sparse input data, improper kernel data approximations during ME calculations (loss of precision from float64 to int32/uint8) also inhibit MELTER CNN models from guaranteed convergence to the global optimum. Furthermore, improper Δ calculations could lead to undesirable outcomes and therefore Λ needs to re-evaluated while using data augmentations or any other computer vision task prior to model training.

6.6 Conclusion and Future Work

To overcome the problem posed to researchers and practitioners for CNN model training concerning the complexities involved in determining model hyperparameters such as Learning Rate (LR) and training regimes, we proposed a Maximum Entropy-based (ME-based) inner loop optimizer for residual neural networks using stochastic gradient descent. The algorithm presented in this chapter is a dynamic and adaptive **M**aximum **E**ntropy-based **L**earning ra**T**e **E**nhance**R** or MELTER, which automatically calibrates the LRs during CNN model training. The proposed MELTER algorithm utilizes only the a priori ME measure of the input dataset to monitor and detect variational drifts in the entropic difference in the data distributions of the input dataset and convolutional kernels. MELTER recalibrates the LR in real-time after every epoch for regret minimization of the two entropical data distributions, enabling a more thorough exploration of the solution space and assisting in feature selection by capturing only the most important features thereby improving classification accuracy.

MELTER removes the arbitrary choices made by deep learning practitioners while determining learning rates for CNN model training. Use of ME measures for real-time

convolutional kernel analysis ensures a high degree of stability, and the invariability of kernel data calculations establishes ME as an ideal measure for monitoring variational drifts, which reduces time-consuming pre-training or manual optimizations. MELTER was empirically validated to outperform the test-set classification performance of ResNet models optimized using existing methods such as manual fine-tuning, cyclical learning rates, super-convergence and 1Cycle policies. MELTER provided a top-10 state-of-the-art performance on the MNIST dataset achieving an accuracy of 99.75% without data augmentation or manual fine-tuning using a ResNet-32 model at relatively low epochs. Similarly, MELTER outperformed existing methods on various benchmarking datasets without real-time data augmentation and manual fine-tuning.

As real-time data augmentations are performed during model training, a priori entropical data distribution of the input dataset cannot be ascertained and is out of scope for this chapter/thesis. In future iterations of the research, we aim to improve the MELTER algorithm to become an online algorithm which makes a priori entropical data distribution estimation obsolete. We also aim to address the limitations of improper Δ calculations by exploring other ME calculation libraries and we will use computational hardware capable of performing tensor operations directly on 64-bit floating point numbers eliminating the need for integer casting.

Finally, the use of adequate high-performance computational resources would assist in further empirical validation of the MELTER algorithm for different application domains, employing a wider range of CNN architectures and a larger more diverse set of experiments.

The previous chapter (Chapter 5) focused primarily on optimizing model training times, while this chapter focused around an entropy-based inner-loop optimization of the LR HyperParameter with a subfocus around increasing rate of convergence and thus subsequently decreasing model training times. Both the previous Chapters (Chapter 5 and Chapter 6) does not address the tendency of CNN models to overfit on

the training data which remains unresolved. In the next chapter (Chapter 7) we propose a novel entropy-based regularization utilizing the knowledge gained from the preceding chapters (Chapters 3 - 6) around sensitivity of CNN models to structural noise.

Chapter 7

Entropy-Based Loss for Regularization of Optimization Problems

In the previous chapter (Chapter 6), we validated the idea that fine-tuning and inner-loop optimization of the Learning Rate (LR) HyperParameter with a Maximum Entropy (ME) monitoring function to assess variational drifts in the CNN feature extraction properties can accelerate the rate of model convergence and decrease model training time. Chapter 6, introduced a novel entropy-based optimization algorithm capable of reducing model training complexity. While an inner-loop optimization of the LR HP improves the rate of convergence, the persistent challenge of mitigating overfitting remains.

In this chapter, we focus on mitigating overfitting and improving classification accuracy by exploring Maximum Entropy (ME) and convolutional kernel analysis as an L_2 regularization technique for non-convex optimization problems. ME measures can be utilized as a calibration method to investigate and penalize complex models for excessive deviation of the convolutional kernels relative to the a priori ME measure of the input dataset. Similar to standard L_1 and L_2 regularization involving statistical analyses, we hypothesize that by promoting simpler weight characteristics of the

convolutional kernels with respect to a priori data distributions of the input dataset, and by penalizing overly complex representations, overfitting can be mitigated and classification performance can be enhanced.

7.1 Introduction

Non-convex optimization problems such as multi-class image classification have complicated underlying functional representations compared to Convex optimization problems which have clearly defined structures to their functional outputs. Generalized solutions for convex problems can be easily computed/predicted as they satisfy known a priori knowledge of the input data. Solutions to non-convex optimization problems usually involve numerical differentiation of a cost function where a guaranteed solution exists for Hessian matrices of second order partial derivatives. While, computation of Hessian matrices guarantees model convergence (Yu et al., 2002), practically however, it resolves into an intractable problem limited by the computational resources of current hardware. Therefore, research emphasis is placed on methods optimizing first order partial derivatives (Duchi et al., 2011).

Convolutional Neural Networks (CNNs) and more specifically Residual Neural Networks (ResNets) (K. He et al., 2016) offer state-of-the-art results in computer vision tasks such as image classification (Szegedy et al., 2015) but, susceptible to structural noise corruption for the input training data; a visualization of is illustrated as Figure 7.1. Introduction of noise either in the input data or during information propagation inside hidden layers exacerbate the problem of overfitting. The fundamental learning theory behind CNNs is to approximate an underlying d-dimensional interpolated function $f(\mathbf{X}) \in \mathbb{R}^d$ using information from n number of d-dimensional input vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ where $\mathbf{x}_i = \langle x^1, x^2, \cdots, x^d \rangle$ and $i, d \in \mathbb{Z}_{>0}$ (Maiorov, 2006). The problem of functional approximation for high-dimensional interpolated data is theoretically

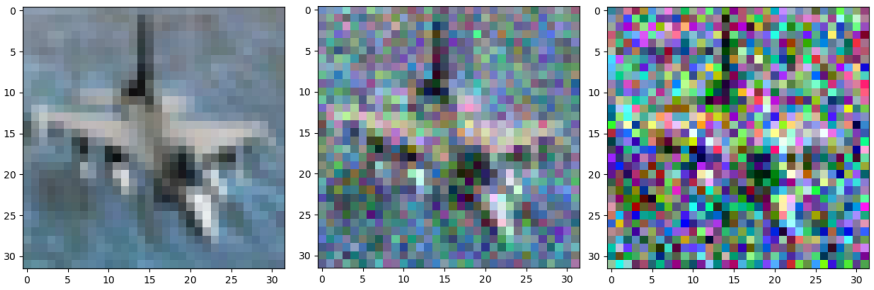

Figure 7.1: Visualization of a random image in the CIFAR-10 dataset displaying Gaussian Structural noise corruption. Left: 0% noise, Middle: 10% noise, Right: 25% noise

non-linear and there is empirical evidence to support the assertion that CNNs simply memorize the input training data (C. Zhang et al., 2016).

CNN models trained without accounting for the unique variances of the input training data are more susceptible to overfitting as the CNN models internal parameters are finely tuned to represent the characteristics of these unique variances (Hawkins, 2004). Misclassification is another problem that occurs when overfit models are unable to distinguish between overlapping variances for different classes of images. Reducing overfitting is also difficult since establishing a theoretical understanding or analyzing the mechanisms of learning in CNNs for non-convex optimization problems such as image classification is generally not well understood (Shamir, 2018).

A simple way to reduce overfitting is to train models using a very large number of images (Shorten & Khoshgoftaar, 2019) such as, the ImageNet dataset consisting of millions of training images used for natural image classification. While using big data solutions might mask the underlying problem of model overfitting, acquisition of clean/noise-free labeled data for supervised model training is challenging. The problem of data acquisition is compounded further by ethical, societal, and practical concerns when dealing with facial datasets, especially for the task of race or gender classification.

Another key challenge while creating datasets are the considerations that need to be made on the distribution of data amongst multiple classes, along with the variability

of data within an individual class. Unbalanced datasets where the data distribution of images is not equal for all the classes introduces unintentional biases during model training (Ganganwar, 2012). The only viable solution to rectify imbalanced datasets is to augment or supplement datasets with new images but, as mentioned before this solution remains an ongoing challenge. To the best of our knowledge, there is no research/work undertaken to regularize entropic data distribution of the convolutional kernel weights using the a priori knowledge of the input dataset in real-time during model training. We hypothesize that regularizing the entropic distribution of the convolutional kernel data during model training, could aide in enhancing classification performance through alleviating the severity of structural noise. Furthermore, L_2 regularization methods adopting entropy measures remains relatively unexplored with existing research focusing on improving kernel learning generally adopting statistical techniques (Cortes et al., 2012).

In this chapter, we propose and empirically validate a novel L_2 regularization technique which utilizes the a priori knowledge of the entropic distribution of the input dataset using Maximum Entropy (ME) (Hartley, 1928) measures and optimizes the entropic distribution of convolutional kernel weights in real-time during model training. Maximum Entropy (ME) measures are widely utilized for monitoring variational drifts and re-calibration of precise, highly sensitive equipment which are affected by structural noise. The novel technique proposed in this chapter is dubbed Maximum Categorical Cross-Entropy (MCCE), which acts as an additional regularization term to the existing and commonly used CCE loss function. MCCE loss calculations are determined by taking into account the entropic distribution of the input dataset proportional to the convolutional kernel weights during model training along with the traditional CCE loss. The pseudo-code for calculating MCCE loss is presented as Algorithm 2 in Section 7.3.

The contributions of this chapter are as follows:

- We propose a novel additional L_2 regularization technique in conjunction with the Categorical Cross Entropy (CCE) loss function using Maximum Entropy (ME) measures to compute a novel Maximum Categorical Cross Entropy (MCCE) loss to enhance classification performance and mitigate model overfitting.
- We empirically validate the MCCE loss function relative to existing loss functions such as CCE, Categorical Hinge and Focal (Lin, Goyal, Girshick, He & Dollár, 2017) on four benchmarking datasets, colorFERET, UTKFace, MNIST and CIFAR-10 with a simple Residual CNN model architecture.
- We demonstrate that the proposed MCCE loss is robust to introduced structural Gaussian noises in the input data compared to the traditional CCE loss.

7.2 Background

As discussed previously in Section 2.1.4, an understanding of how CCE loss is calculated was obtained. Section 7.2.3 details how kernel regularization influence CCE loss with its limitation. Section 7.2.4 provides the theoretical background of Maximum Entropy (ME) and methods to calculate ME along with estimating the reconstruction loss. Given that CNN model training introduces noises during convolutional operations or information propagation and that any inherent noise present in the input data can significantly affect model performance, a noise-robust alternative to CCE could help improve classification performance and mitigate overfitting. This is because, the model could potentially learn to ignore the structural noise of the input data and thus preventing convolutional kernels from being fine-tuned to the unique variances or noise characteristics of the input dataset.

7.2.1 Focal loss

Focal loss described in (Lin et al., 2017), primarily addresses the extreme imbalance in foreground and background object detection. The focal loss function is derived from the previously described Cross-entropy loss in Section 2.1.4. While the authors (Lin et al., 2017) describe introducing a weight factor to address class imbalance, we reserve skepticism around such a simple solution as an accurate determination of the proposed weight factor without affecting overfitting, generalizability and model convergence properties remains elusive. Furthermore, our experimental data presented in Section 7.4 adds credence to our skepticism.

7.2.2 Hinge loss

Hinge loss is primarily reserved for margin classifiers such as support vector machines but, as authors (Jin, Fu & Zhang, 2014) point out, hinge loss can provide enhanced classification performance for traffic detection systems utilizing CNNs. We aim to explore and analyze the performance of models trained using hinge loss in relation to our proposed novel entropy-based loss to establish its efficacy across a wide range of application domains. Quantitative data for such an analysis is collected from our experimental setup and presented in Table 7.1 and discussed in Section 7.5.

7.2.3 Kernel Regularization

The intuition behind regularization is that of Ockham's razor to penalize complex models and to promote simpler models during training with the intention of strengthening model generalizability. Unlike empirical risk minimization which only considers loss minimization, regularization was proposed to minimize structural risk which considers both complexity and loss minimization. The most prominent and simple ker nels that greatly minimize loss are selected (Bilgic et al., 2014). Model complexity is represented

in two ways, as a function of the total number of features with non-zero weights (L_1) or as a function of all the weights of all the features in a model (L_2). L_2 regularization is most commonly used in computer vision tasks for CNN models such as ResNet. Model complexity can be quantified using the L_2 regularization formula given in Equation (7.1), defined by using the sum of squares of all the feature weights as the regularization term (Cortes et al., 2012).

$$||\omega||^2 = \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_{H'\chi'}^2 \tag{7.1}$$

In Equation (7.1), the magnitude of the absolute value of the weight ω indicates complexity. Feature weights close to zero have no significant impact on model complexity, while large outlier weight values have a more pronounced impact on ω. The quantity of feature weights $H'\chi'$ determined using the number of trainable model parameters (a product of the convolutional depth H' and breadth χ') contribute greatly to ω and model complexity. Furthermore, kernel regularization as it is implemented currently for CCE loss utilizes CNN computed label errors and does not take into account the data distribution of the convolutional kernel weights. We hypothesize that, optimizing the kernel data distribution utilizing the a priori knowledge of the input dataset could help enhance classification performance.

7.2.4 Maximum Entropy and Reconstruction Loss

The use of Maximum Entropy (ME) for applications such as convolutional kernel analysis is justified since ME is the only consistent way of selecting a single discrete data point from the set of input data vectors to best fit the regression curve, proven axiomatically in (Shore & Johnson, 1980) and expanded in (Johnson & Shore, 1983). A method of approximating ME for digital images is achieved through the use of distributed normalized histograms (Gonzalez & Woods, 2007; A. K. Jain, 1989). The

open-source SciKit-image processing library written in Python can be used to calculate the ME measures for images (Virtanen et al., 2020).

Entropy in images is related to the complexity contained in a given neighborhood, computed by using a circular disk with a radius of r. The disk is used to measure minute variations in local grayscale level distribution. The maximum entropy for an image depends on the number of gray levels, an 8 bit image has 256 gray levels (0-255) which has a theoretical maximum entropy of $log_2(2^8) = 8$ bits per pixel. Changing the value of r can invariable produce higher or lower ME measure as illustrated in Figure 3.1. Similarly higher or lower ME values will be obtained while measuring convolutional kernel weights. A decrease in ME divergence can be observed in Figure 3.1 for r values of 5 and 50 relative to r values of 1 and 5. A significant difference in spatial/semantic information in the images can be seen with greater r values, which suggests loss in precision during approximation.

ME measures for color images require the computation on each of the three color channels, Red (R), Green (G) and Blue (B) i.e. RGB separately and averaging the result. The averaged ME measures for images in the *colorFERET* and *UTKFace* datasets are *2.09* and *2.25* bits per pixel respectively using an r value of 1. The amount of time taken to calculate the ME measures is insignificant as the ME calculation script can be executed in parallel on the CPU, while CNN model training occurs on the GPU, as *evidenced in the supplementary data uploaded*. Solutions other than ME for image reproduction/reconstruction from noisy or incomplete measurements such as, the use of non-linear variations on fourier transformations fail when convolutional kernels are incorporated (Donoho et al., 1990). Furthermore, ME reconstruction has been shown to provide superior noise suppression while mostly preserving de-emphasized structural noise near the baseline (relative to high signal information) (Donoho et al., 1990).

Accurate reconstructions can be approximated using a 1D projection of any underlying function which is reduced to $g(\mathbf{X}) \in \mathbb{R}^d$ such that $\mathbf{x}_i \in \mathbf{X}$ (Reis & Roberty, 1992). As

discussed in Section 7.1, the underlying functional representation of the input dataset is $f(\mathbf{X})$, the difference between the true representation $f(\mathbf{X})$ and the ME reconstruction approximation $g(\mathbf{X})$ is the reconstruction loss for the input dataset. Results presented in (Reis & Roberty, 1992), indicate that reconstructions using accurate and noisy data had insignificantly small variations compared to the original, attesting to the noise-robust ability of using ME measures for reconstruction. This noise averse characteristic of ME is especially important for image classification as lighting or ISO parameters of the physical hardware can significantly affect the performance of CNN models. Reconstruction loss is described as the convolutional kernel data loss whereas CCE can be characterized as a class label loss.

7.3 Maximum Categorical Cross-Entropy (MCCE)

The classification of data in CNNs primarily depends on the convolutional kernels represented by their weights. Optimization of kernel weights using a loss function is performed to ensure a closer approximation to the underlying function $f(\mathbf{X})$ is achieved. As discussed in Section 2.1.4, CCE is a measure of difference between two probability distributions, the ground truth and CNN computed label for a class C. The drawback of CCE is that it only considers class label errors and does not account for the distribution of the convolutional kernel weights. The estimation of kernel weight probability distributions is critical in knowing the state of model training and learning capacities which could enhance classification performance, and MCCE is proposed to rectify this limitation.

The Maximum Categorical Cross-Entropy (MCCE) loss function monitors the data distribution of convolutional kernel weights using ME measures along with traditional CCE loss and penalizes models which are overly complex. A priori knowledge of the entropic distribution of the input data can be computed using ME measures which is used

as a baseline to monitor convolutional kernel weight distributions and penalize models with greater divergences. It is well understood that maximizing entropy measures using even partial information (such as from convolutional kernel weights) can enhance the estimation of probability distributions (Macqueen & Marschak, 1975) used extensively to calculate CCE loss.

The main criterion for producing high quality reconstruction approximation is the incorporation of two-dimensional convolutions, which is traditionally a computational burden (Wernecke et al., 1977). CNN models implicitly use two-dimensional convolutions to produce feature maps therefore, the computational overheads are eliminated making the computation of MCCE loss very efficient. Furthermore, using MCCE loss a L_1 difference can be calculated between the reconstruction approximation $g(\mathbf{X})$ and ground truth $f(\mathbf{X})$ (CCE error).

Reconstruction error can be calculated using the a priori determined ME of the input dataset and reconstruction approximation $g(\mathbf{X})$. Monitoring the divergence between the CCE error and reconstruction error provides an indication in the degree of deviation of convolutional kernel weights and predicted class labels to the deviation of $f(\mathbf{X})$ to determine if the convolutional kernels are stuck in a global minima/maxima and thus indicating model learning has saturated.

7.3.1 Algorithm

The pseudo-code for MCCE loss is presented as Algorithm 2, which requires the calculation of a 1D linear interpolation output for reconstruction loss is provided in Section 7.3.2.

Algorithm 2 Pseudo-Code for the proposed Maximum Categorical Cross Entropy (MCCE) loss function

1: **Input:** One-hot encoded ground truth (Y) and CNN predicted ($\hat{Y}$) class labels, a priori ME of training images in the dataset (i.e. ME(**X**)).
2: **Output:** Probabilistic logarithmic loss of $\hat{Y}$ with respect to the ground truth Y.
3: **Initialize:** $\Lambda \leftarrow$ ME(**X**), $\mu \leftarrow$ ME($W \mid \omega \in W$)
4: γ = -log($\frac{e^{s_p}}{\Sigma_j^C e^{s_j}}$) {CCE loss, s_p is the CNN score and s_j the ground truth for the class C}
5: $\kappa = \Lambda - \mu$ {Convolutional Reconstruction (CR) loss}
6: κ = Interpolation(κ, (0,Λ), (0,1)) {1d linear interpolation to output κ between 0 and 1 rather than in the range of 0 to Λ}
7: $\Delta = \gamma + \kappa$ {Maximum loss = CCE loss + CR loss}
8: return Δ

7.3.2 1D Linear Interpolation Computation

A one-dimensional interpolation of the reconstruction error/loss is required as the MCCE loss is an extension of CCE loss which outputs values between 0 and 1. A linear interpolant is the straight line between the two known points given by their coordinates (a_0, b_0) and (b_1, b_1) (Davis, 1975). For any value i in the interval (a_0, a_1), the value of j along the straight line can be calculated using the equation of slopes given in Equation 7.2

$$\frac{j - b_0}{i - a_0} = \frac{b_1 - b_0}{a_1 - a_0} \quad or \quad j = \frac{b_0(a_1 - i) + b_1(i - a_0)}{a_1 - a_0} \tag{7.2}$$

7.4 Experimentation

Experimentation revolved around quantitatively measuring classification performance and train-test divergence to determine the degree of overfitting for our proposed novel MCCE loss in conjunction with existing loss functions (such as CCE, Focal and Categorical Hinge) highlighted in Section 7.2. No modifications were made to our CNN model training regime compared to the original implementation presented in (K. He et

al., 2016) apart from using different testing hardware and software frameworks (Keras with a tensorflow backend). Only a single evaluation run could be performed due to the limited computing infrastructure available.

7.4.1 Datasets

To determine the effect of racial bias and the efficacy of our novel MCCE loss function, we select a balanced dataset (UTKFace (Z. Zhang, Song & Qi, 2017)) where each class of race/ethnicity has an equal number of images and an unbalanced dataset (colorFERET (Phillips, Wechsler, Huang & Rauss, 1998)) where the distribution of data across all of the classes is unequal.

The colorFERET dataset contains 11,338
semi-controlled color images of 512×768 pixel size with 13 different poses from 994 test subjects. Due to our limited computing infrastructure, the images needed to be downsampled to 96×96 pixel resolution using cubic interpolation. The original dataset contains nine classes (Asian, Asian-Southern, Asian-Middle-Eastern, Black-or-African-American,
White, Hispanic, Native-American, Other and Pacific-Islander). Due to the very limited number of test subjects and images for four of the nine classes, the dataset was reduced to five classes (Asian, Asian-Middle-Eastern, Black-or-African-American, White, Hispanic) containing a total of 11,172 images.

The original UTKFace dataset contains 23,708 in-the-wild color images of 200×200 pixel size with five ethnic classes (White, Black, Asian, Indian and Others) of all age groups. Only the OECD definition for working age population (15-64) consisting of 18,095 images are considered since the facial variations are not severe enough to cause any unexpected errors like misclassification or underfitting. The images used in our experimentation were downsampled to 96×96 pixel resolution using cubic interpolation.

Finally, classification performance of the models on two additional and commonly used benchmarking datasets, MNIST and CIFAR-10 were analyzed to determine if the proposed MCCE loss is generalizable across different image data. The MNIST dataset (LeCun et al., 1998) includes 28×28 pixel resolution black and white handwritten digits. MNIST consists of 60,000 training and 10,000 test images split equally into ten classes for each numeric character. The CIFAR-10 (Krizhevsky & Hinton, 2009) dataset includes 50,000 training and 10,000 testing natural color images with a 32×32 pixel resolution, split equally into ten classes, which include pictures of airplanes, birds and other such natural image classes.

7.4.2 Experimental Setup and Results

All experiments presented in this chapter were carried out with a single RTX 3070 GPU with 8GB of VRAM, generously provided by InfuseAI Limited (New-Zealand). All models were trained from scratch with the datasets randomly shuffled when reading from storage into memory. The training data was again randomly shuffled during model training to mitigate any variability in the input data. The model parameters utilized for model training are, a ResNet-8 model with a batch size of 128; an SGD optimizer with a learning rate of 0.1 and a dynamic epoch count with an early stopping patience of 10, monitored on the training accuracy. The results presented in Table 7.1, highlight the classification performance of models trained with different loss functions.

7.5 Discussion

Analyzing the data presented in Table 7.1, we clearly identify the effectiveness of classification performance when models are trained with the novel MCCE loss with respect to overfitting (train-test Δ) and test-set classification accuracy. Expanded tests using either Focal or Categorical Hinge losses on all the datasets were not justified

Table 7.1: Results validating the efficacy of training models using the proposed MCCE loss function to enhance performance and mitigate overfitting

Loss Function	***Dataset***	***Train Acc.***	***Test Acc.***	***Train-Test*** Δ	**Training Time**
		Gaussian Noise: **0%**			
MCCE	colorFERET	94.39	90.56%	3.83%	0:42:23
CCE	colorFERET	95.66%	89.22%	6.44%	0:53:29
Focal	colorFERET	86.72%	83.76%	2.96%	1:06:23
Hinge	colorFERET	63.66%	62.19%	1.47%	0:03:14
		Gaussian Noise: **10%**			
MCCE	colorFERET	91.33	88.94%	2.39%	0:50:50
CCE	colorFERET	87.93%	83.09%	4.84%	0:28:36
Focal	colorFERET	78.49%	78.08%	0.41%	0:46:38
Hinge	colorFERET	63.18%	64.12%	-0.94%	0:03:11
		Gaussian Noise: **25%**			
MCCE	colorFERET	83.47%	77.9%	5.57%	0:37:17
CCE	colorFERET	79.19%	75.35%	3.84%	0:20:24
Focal	colorFERET	75.43%	76.2%	-0.77%	1:00:47
Hinge	colorFERET	63.53%	62.69%	0.84%	0:03:25
		Gaussian Noise: **0%**			
MCCE	UTKFace	83.59%	78.28%	5.31%	1:35:48
CCE	UTKFace	80.82%	77.07%	3.75%	1:01:24
		Gaussian Noise: **10%**			
MCCE	UTKFace	80.51%	74.61%	5.9%	0:48:13
CCE	UTKFace	80.56%	76.29%	4.27%	0:50:19
		Gaussian Noise: **25%**			
MCCE	UTKFace	76.16%	70.93%	5.23%	0:50:47
CCE	UTKFace	74.97%	66.93%	8.04%	0:42:09
		Gaussian Noise: **0%**			
MCCE	CIFAR-10	83.75%	78.33%	5.42%	0:58:35
CCE	CIFAR-10	83.37%	75.88%	7.49%	0:48:52
		Gaussian Noise: **10%**			
MCCE	CIFAR-10	77.5%	73.24%	4.26%	1:03:23
CCE	CIFAR-10	76.76%	67.07%	9.69%	0:50:05
		Gaussian Noise: **25%**			
MCCE	CIFAR-10	67.28%	59.68%	7.6%	0:47:31
CCE	CIFAR-10	65.94%	59.00%	6.94%	0:48:04
		Gaussian Noise: **0%**			
MCCE	MNIST	99.16%	98.92%	0.24%	0:27:44
CCE	MNIST	99.31%	98.86%	0.45%	0:34:24
		Gaussian Noise: **10%**			
MCCE	MNIST	99.19%	98.82%	0.37%	0:32:16
CCE	MNIST	99.2%	98.01%	1.19%	0:38:56
		Gaussian Noise: **25%**			
MCCE	MNIST	98.71%	98.17%	0.54%	0:26:08
CCE	MNIST	98.86%	98.03%	0.83%	0:38:40

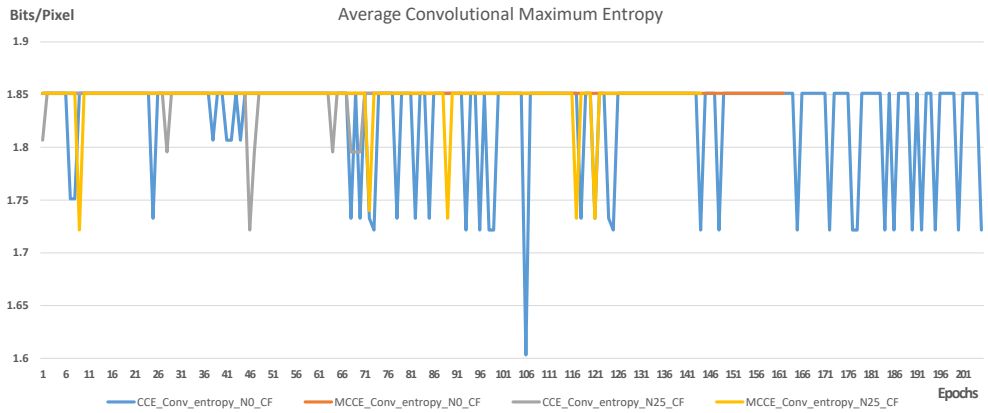

Figure 7.2: Average maximum entropy measures for the convolutional kernel weights during model training computed using $r = 1$ for the colorFERET dataset when 0% (N0) and 25% (N25) Gaussian structural noise is introduced

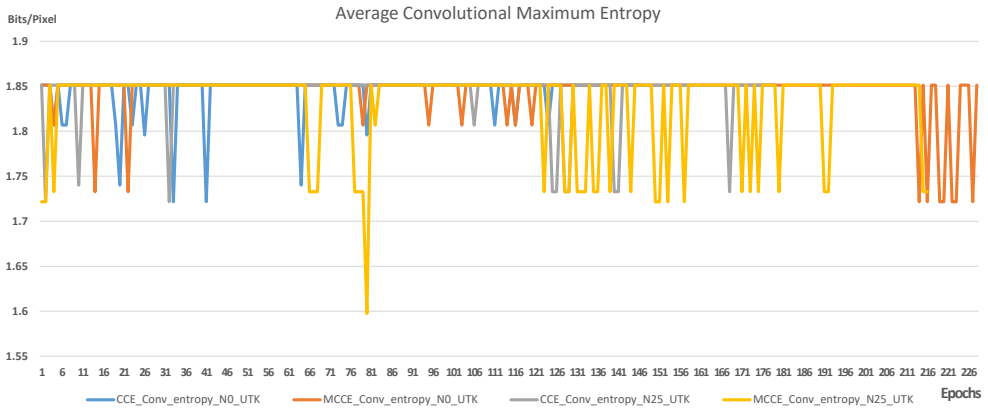

Figure 7.3: Average maximum entropy measures for the convolutional kernel weights during model training computed using $r = 1$ for the UTKFace dataset when 0% (N0) and 25% (N25) Gaussian structural noise is introduced

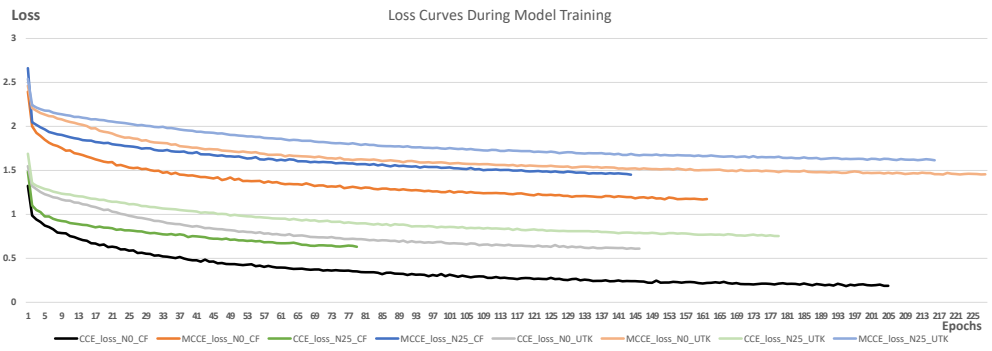

Figure 7.4: Loss curves during CNN model training with MCCE and CCE loss functions on the colorFERET and UTKFace datasets when 0% (N0) and 25% (N25) Gaussian structural noise is introduced

since models trained using these loss functions failed to adequately converge relative to MCCE and CCE trained models. Hinge loss was evaluated on the MNIST dataset inline with the discussion in Section 7.2.2. Generally, MCCE performed slightly better compared to CCE when no structural Gaussian noise was introduced and fared markedly superior when 10% and 25% noise was implemented.

Analyzing Figures 7.2 and 7.3 in conjunction with the data presented in Table 7.1, we can determine there exists a strong correlation between divergences from the baseline average convolutional ME and model performance. Examining Figure 7.2, we can assert that the CNN models trained with the MCCE loss function exhibit a tendency to deviate from the ME baseline of 1.85 less frequently and to a smaller degree relative to models trained with CCE loss. Furthermore, when no structural noise is introduced, MCCE models displayed a steady-state condition to the baseline ME whereas, CCE models were unsteady and thus required a greater number of epochs to fully converge. Although deviations from the baseline ME does not in itself indicate non-ideal convergence patterns, it does highlight exploration of the search space as is necessary for non-convex optimization problems.

Figure 7.4 visualizes the loss curves for both MCCE and CCE trained CNN models on both the colorFERET and UTKFace datasets. MCCE trained models had a relatively higher loss of 1.17 and 1.45 compared to 0.18 and 0.61 for the colorFERET and UTKFace datasets respectively, when no noise was introduced. The introduction of 25% noise produced losses of 1.45 and 1.62 for MCCE trained models relative to 0.63 and 0.75 for CCE trained models on the colorFERET and UTKFace datasets respectively. These high losses for MCCE trained models suggests a greater degree of L_2 kernel regularization is being employed by the algorithm. Higher loss also indicate MCCE trained models have not fully converged and greater performance enhancements can be achieved with manual HyperParameter fine tuning. MCCE models also converged faster in some instances highlighting learning capacity of the MCCE models, which can

be improved with exploration of additional techniques to improve convergence such as dynamic learning rates.

Results in Table 7.1 also emphasize the noise-robustness of MCCE trained models relative to CNN models trained with CCE loss. On the colorFERET dataset, MCCE trained models demonstrated on average a 3.25% enhancement in classification performance with a 1.11% reduction in overfitting relative to CCE trained models. However, on the UTKFace dataset, although MCCE trained models performed on average 1.18% better in terms of test-set classification accuracy; there was a slight increase in overfitting of 0.13% compared with CCE trained models. This marked increase in overfitting can be attributed to an anomaly in the experimental data when a 10% Gaussian noise was introduced where, the MCCE model's performance deteriorated by 1.68%. Increasing the sample size should resolve this anomaly as it could be simply due to sub-optimal weight initialization of the convolutional kernels.

On the CIFAR-10 dataset, MCCE trained models in general showed leading classification performance compared to CCE trained models with an average accuracy improvement of 2.65% and a 2.72% reduction in overfitting. A similar trend of broad advancements for MCCE trained models for the MNIST dataset can be observed where a 0.34% increase in classification performance and a 0.44% decrease in overfitting was achieved in comparison to CCE trained models. Overall, CNN ResNet-8 models trained with the proposed MCCE loss implemented as Algorithm 2 and described in Section 7.1 demonstrated on average a 1.85% enhancement in test-set classification accuracy with a 1.04% reduction in model overfitting across the four benchmarking datasets detailed in Section 7.4.1.

7.6 Conclusion and Future Work

In this chapter, we proposed a novel L_2 regularization technique utilizing a priori knowledge of the entropic distributions of the input datasets; in conjunction with the commonly used Categorical Cross Entropy (CCE) loss function dubbed, Maximum Categorical Cross Entropy (MCCE). While CCE evaluates the probability distributions of the CNN predicted and ground truth class labels, MCCE extends this evaluation to include the entropic distribution of convolutional kernel weights during model training.

MCCE provides a robust noise-averse method of calculating model loss since partial knowledge of the entropic distribution of the input data is determined a priori. Furthermore, large divergences of the entropy measures of the convolutional kernel weights from that of the input dataset are penalized in real-time during model training promoting simpler representations.

In other words, MCCE loss takes into account the label loss and convolution kernel weight distribution or reconstruction loss, penalizing model training if either of these distributions greatly diverge from each other. MCCE loss has been empirically validated on ResNet-8 models for four benchmarking datasets; colorFERET, UTKFace, CIFAR-10 and MNIST datasets to enhance classification accuracy by up to **6.17%** and minimize overfitting by up to **5.43%**. MCCE trained models also offered leading classification performance when structural Gaussian noise was introduced in the input training data compared with models trained with the traditional CCE loss.

Finally, reproducibility and implementation of the MCCE loss function is comparatively effortless since, the MCCE algorithm (provided as Algorithm 2 in Section 7.3) leverages existing publicly available computational libraries (Scipy, Numpy, Keras and Tensorflow) thereby involves minimal modifications to existing code bases. Additionally, the required calculations for evaluating MCCE loss such as computation of entropy measures are accelerated functional evaluations with extremely efficient

scientific computational libraries.

Furthermore, the general improvements in rate of convergence of MCCE trained models tend to offset any additional computational overheads that might influence model training times. As such, the authors are confident that integrating the proposed MCCE loss function by deep learning practitioners pose no significant challenge to the community.

This chapter concludes the experimental chapters of this thesis by addressing CNN model overfitting, the final unresolved challenge outlined in the introduction chapter (Chapter 1, Section 1.1). The next chapter (Chapter 8), presents a contextual summary of the contributions, implications and limitations of the research presented in this thesis.

Chapter 8

Conclusion

8.1 Summary

In this thesis, we empirically evaluated the key underlying principles of Convolutional Neural Networks (CNNs) and their effectiveness as feature extractors in achieving state-of-the-art performance in image classification tasks. We set out to address a few of the unresolved challenges outlined in Section 1.1 utilizing quantitative metrics for optimization, primarily in answering our overarching research question: *how could entropy measures, specifically Shannon's Entropy (SE) and Maximum Entropy (ME) be utilized to optimize CNN model training?*.

In Chapter 3, we quantitatively examined the process of feature extraction with respect to neural configurations. We utilized two metrics, Maximum Entropy (ME) and Signal-to-Noise Ratio (SNR) to aid us in understanding information propagation within the hidden layers of a CNN. We found that, convolutional feature extraction and thereby classification accuracy is limited through two different mechanisms, information underflow and overflow. These phenomena can be characterized by a relative discrepancy between the amount (quantified through ME) and quality of information (calculated by SNR) extracted through the convolutional layers with respect to the input dataset.

Although both these mechanisms restrict adequate feature extraction, we empirically demonstrated that, information underflow is preferable which predominantly occurs with input datasets having high ME and SNR measures. The research evidence also indicated that, convolutional kernels (basic units of CNN feature extraction) might prematurely saturate for datasets with low ME and SNR measures.

In Chapter 4, we evaluated the predominance of kernel saturation for datasets with low ME and SNR measures. We found that, by augmenting datasets with images containing similar semantic information but different structural data representations, kernel saturation could be mitigated. Based on this observations, we proposed a novel data augmentation technique to mitigate saturation, thereby improving model performance. Quantitative examination on different datasets produced an average model performance improvement of 5.07%. Through our experimental findings, the problem of neural sparsity due to over-parameterized CNN models became apparent, implying training times could be decreased by optimizing feature extraction and abstraction.

In Chapter 5, we delved into the working principle behind feature extraction and abstraction in CNNs, specifically, CNN architectures utilizing residual/skip connections. We found that, CNN architectures incorporating residual connections behave as a collection of ensemble networks with over-parameterized deep CNN architectures resolving into a collection of independent feature extractors, making the process of feature extraction redundant since, skip connections facilitate only the most prominent features to be retained. We proposed an entropy based heuristic utilizing the apriori data distributions of input the dataset to offer a simplified approach in restricting CNN model depth. Constrained convolutional layers enforces feature compression, thereby mitigating the problem of over-parameterization, aiding in accelerating model training. Our empirical results demonstrate that, a CNN model's depth can be constrained up to its maximum compressibility factor determined through SE without affecting model performance with an average of 45.22% reduction in training time. While model

depth is an important HyperParameter (HP), model Learning Rate (LR) dictates the convergence or divergence patterns for CNN models and as such fine-tuning the LR HP could improve model performance and further decrease training times.

In Chapter 6, we investigated the problem of manually fine-tuning model LR especially when CNN architectures are complex and sub-optimal determination of model LR could induce divergent behavior. We found that, dynamically adjusting a model's LR based on the entropic distribution of the convolutional kernels and the input dataset during model training maintains if not outperforms other existing fine-tuning methods. As such, to eliminate unintentional sub-optimal configuration of the learning rate parameter, we proposed a maximum entropy-based inner loop optimizer for residual neural networks. Furthermore, our novel entropy-based optimizer produces top-10 state-of-the-art model performance without the need for manual fine-tuning or real-time data-augmentation techniques. However, the fundamental problem of employing optimization techniques for a single objective (classification accuracy) is that, it can induce unintentional overfitting to the internal parameters of a CNN.

In Chapter 7, we analyzed a method of optimization which provides a more comprehensive and robust optimization of HPs without introducing significant overfitting. We found that, the susceptibility of CNN models to structural noise in the input dataset or introduced during information propagation within the hidden layers significantly affects model performance and overfitting. By utilizing additional L_2 regularization of the convolutional kernels with respect to the entropic data distribution of the input dataset we could improve the accuracy of loss calculations with respect to overfitting, improving generalization by up to 5.4.% with an enhancement in model performance by up to 6.17%.

8.2 Significance and Implications

8.2.1 Classification Performance

Classification/recognition of digital data is heavily utilized in multiple application domains such as content filtering, Fraud detection, anomaly identification, medical diagnoses, data organization and so on. The real-world implications of enhanced classification accuracy in the aforementioned application domains is quite intuitive and obvious. In instances where false positives/negatives could mean the difference between life and death with respect to medical diagnoses or guilt and innocent in terms of crime scene investigations, having accurate deep learning models is critical. We believe that our proposed algorithms could help in this regard.

8.2.2 Training Time

The rapid progress in CNN model performance has been achieved through increasingly complex architectures and training regimes requiring an exponential increase in computational resources. The direct effect of this attrition-based approach in increasing model performance is that most research is unable to be independently verified due to the lack of access to high-performance computational resources. Our research has shown to significantly reduce model training times with no additional computational demands or detriment to model classification performance. Utilizing our proposed heuristic and algorithms, practitioners can deploy deep learning models in computationally constrained environments such as embedded and mobile systems which have far reaching benefits. Researchers similarly can expeditiously test novel hypotheses which can accelerate research in the domain.

8.2.3 Generalization

Most deep learning models overfit on the training data to some extent, limiting their applicability in different domains or requiring the collation of large data samples. Increased generalizability also serves to mitigate the problem of unintentional biases in sensitive applications such as facial recognition where, deep learning models could have a skewed representation favoring a specific sub-class, primarily due to overfitting. Our proposed algorithms and loss functions can help mitigate the problem of bias and enable implementation of deep learning models in a more diverse set of application domains where, data procurement is limited due to privacy or other restrictions that might be imposed.

8.3 Limitations and Future work

Although we have made significant strides in trying to optimize CNNs, the methods outlined in this thesis might not be effective for different CNN architectures, dissimilar datasets such as time-series or distinct computer vision tasks like image captioning, object detection or segmentation. Furthermore, the methods and algorithms presented in thesis are validated empirically only for a limited set of benchmarking datasets and CNN architectures. The rapid progress in the field of computer vision could possibly render CNN models obsolete which makes it imperative to apply the principles proposed in this thesis to different domains and state-of-the-art architectures which is reserved as future work. Furthermore, the process of feature extraction, abstraction and classification in CNN models is still not well understood especially with respect to structural, topological and semantic characteristics respectively. Analyzing these areas from a mathematical perspective with regards to the convolution operation is both intriguing and exciting which is reserved as future work.

References

Ardakani, A., Condo, C., Ahmadi, M. & Gross, W. J. (2018). An Architecture to Accelerate Convolution in Deep Neural Networks. *Transactions on Circuits and Systems I: Regular Papers*, *65*(4), 1349–1362.

Arora, R., Basu, A., Mianjy, P. & Mukherjee, A. (2016). Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*.

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

Battiti, R. (1989). Accelerated backpropagation learning: Two optimization methods. *Complex systems*, *3*(4), 331–342.

Bebis, G. & Georgiopoulos, M. (1994). Feed-forward neural networks. *Potentials*, *13*(4), 27–31.

Becker, D. J., Sterling, T., Savarese, D., Dorband, J. E., Ranawak, U. A. & Packer, C. V. (1995). BEOWULF: A parallel workstation for scientific computation. In *Proceedings, international conference on parallel processing* (Vol. 95, pp. 11–14).

Becker, S., Le Cun, Y. et al. (1988). Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the connectionist models summer school* (pp. 29–37).

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A. & Hjelm, D. (2018). Mutual information neural estimation. In *International conference on machine learning* (pp. 531–540).

Belkin, M., Hsu, D., Ma, S. & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, *116*(32), 15849–15854.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade* (pp. 437–478). Springer.

Bergstra, J. & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281–305.

Bilgic, B., Chatnuntawech, I., Fan, A. P., Setsompop, K., Cauley, S. F., Wald, L. L. & Adalsteinsson, E. (2014). Fast image reconstruction with L2-regularization. *Magnetic resonance imaging*, *40*(1), 181–191.

Bottou, L. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, *91*(8), 12.

Canziani, A., Paszke, A. & Culurciello, E. (2016). An analysis of deep neural network

models for practical applications. *arXiv preprint arXiv:1605.07678*.

Caruana, R., Lawrence, S. & Giles, C. L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems* (pp. 402–408).

Chrabaszcz, P., Loshchilov, I. & Hutter, F. (2017). A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*.

Coates, A., Ng, A. & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the international conference on artificial intelligence and statistics* (pp. 215–223).

Coates, A. & Ng, A. Y. (2011). Selecting receptive fields in deep networks. In *Advances in neural information processing systems* (pp. 2528–2536).

Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L. & Batra, D. (2015). Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*.

Collins, M. & Duffy, N. (2002). Convolution kernels for natural language. In *Advances in neural information processing systems* (pp. 625–632).

Cortes, C., Mohri, M. & Rostamizadeh, A. (2012). L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Davis, P. J. (1975). *Interpolation and approximation*. Courier Corporation.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y. & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems* (pp. 1269–1277).

Donoho, D. L., Johnstone, I. M., Stern, A. S. & Hoch, J. C. (1990). Does the maximum entropy method improve sensitivity? *Proceedings of the National Academy of Sciences*, *87*(13), 5066–5068.

Duchi, J., Hazan, E. & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Einstein, A. (1933). *On the method of theoretical physics: The herbert spencer lecture; delivered at oxford 10 june 1933*. Clarendon Press.

Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, *2*(4), 42–47.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Gonzalez, R. C. & Woods, R. E. (2007). Image processing. *Digital image processing*, *2*.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). Deep learning. book in preparation for mit press. *URL¡ http://www. deeplearningbook. org*, *1*.

Gull, S. F. & Skilling, J. (1984). Maximum entropy method in image processing. In *Iee proceedings f (communications, radar and signal processing)* (Vol. 131(6), pp.

646–659).

Hartigan, J. A. & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108.

Hartley, R. V. (1928). Transmission of information. *Bell Labs Technical Journal*, *7*(3), 535–563.

Hawking, S. (2009). *A brief history of time: from big bang to black holes*. Random House.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, *44*(1), 1–12.

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

He, Y., Zhang, X. & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the ieee international conference on computer vision* (pp. 1389–1397).

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Hoffer, E., Hubara, I. & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in neural information processing systems* (pp. 1731–1741).

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4700–4708).

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jain, A. K. (1989). *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall,.

Jain, P. & Kar, P. (2017). Non-convex optimization for machine learning. *arXiv preprint arXiv:1712.07897*.

Jenssen, R. (2009). Kernel entropy component analysis. *Transactions on pattern analysis and machine intelligence*, *32*(5), 847–860.

Jin, J., Fu, K. & Zhang, C. (2014). Traffic sign recognition with hinge loss trained convolutional neural networks. *IEEE Transactions on Intelligent Transportation Systems*, *15*(5), 1991–2000.

Johnson, R. & Shore, J. (1983). Comments on and correction to" Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy". *Transactions on Information Theory*, *29*(6), 942–943.

Karpathy, A. (2016). *Connecting images and natural language* (Unpublished doctoral dissertation). Stanford University.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kraskov, A., Stögbauer, H. & Grassberger, P. (2004). Estimating mutual information. *Physical review E*, *69*(6), 066138.

Krbcova, Z. & Kukal, J. (2017). Relationship between entropy and SNR changes in image enhancement. *EURASIP Journal on Image and Video Processing*, *2017*(1), 83.

Krizhevsky, A. & Hinton, G. (2009). *Learning multiple layers of features from tiny images.* Citeseer.

Krizhevsky, A. & Hinton, G. (2010). Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, *40*.

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

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Huang, F. J. & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Computer vision and pattern recognition, 2004. cvpr 2004. proceedings of the 2004 ieee computer society conference on* (Vol. 2, pp. II–104).

LeCun, Y., Kavukcuoglu, K., Farabet, C. et al. (2010). Convolutional networks and applications in vision. In *Iscas* (Vol. 2010, pp. 253–256).

Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the ieee international conference on computer vision* (pp. 2980–2988).

Loshchilov, I. & Hutter, F. (2016). SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Luo, J.-H. & Wu, J. (2017). An entropy-based pruning method for CNN compression. *arXiv preprint arXiv:1706.05791*.

Luo, L., Xiong, Y., Liu, Y. & Sun, X. (2019). Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*.

Luo, W., Li, Y., Urtasun, R. & Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 4898–4906).

Macqueen, J. & Marschak, J. (1975). Partial knowledge, entropy, and estimation. *Proceedings of the National Academy of Sciences*, *72*(10), 3819–3824.

Maiorov, V. (2006). Approximation by neural networks and learning theory. *Journal of Complexity*, *22*(1), 102–117.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150203.

Marr, B. (2018). How much data do we create every day? the mind-blowing stats everyone should read. In *Forbes*.

McLuhan, M. & Gordon, W. T. (2013). *Counterblast: 1954*. Ère.

Moshkovitz, M. & Tishby, N. (2017). Mixing complexity and its applications to neural networks. *arXiv preprint arXiv:1703.00729*.

Petrovici, M.-A., Damian, C. & Coltuc, D. (2018). Maximum Entropy Principle in Image Restoration. *Advances in electrical and computer engineering*, *18*(2), 77–84.

Phillips, P. J., Wechsler, H., Huang, J. & Rauss, P. J. (1998). The FERET database and evaluation procedure for face-recognition algorithms. *Image and vision computing*, *16*(5), 295–306.

Pierce, J. R. (1980). *An Introduction to Information Theory. Symbols, Signals and Noise., Revised edition of Symbols, Signals and Noise: the Nature and Process of Communication (1961).* Dover Publications Inc., New York.

Rathore, P., Kumar, D., Rajasegarar, S. & Palaniswami, M. (2017). Maximum entropy-based auto drift correction using high-and low-precision sensors. *Transactions on Sensor Networks (TOSN)*, *13*(3), 24.

Reis, M. & Roberty, N. C. (1992). Maximum entropy algorithms for image reconstruction from projections. *Inverse Problems*, *8*(4), 623.

Reza, F. M. (1994). *An introduction to information theory*. Courier Corporation.

Ripley, B. D. (2007). *Pattern recognition and neural networks*. Cambridge university press.

Robbins, H. & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Rolinek, M. & Martius, G. (2018). L4: Practical loss-based stepsize adaptation for deep learning. In *Advances in neural information processing systems* (pp. 6433–6443).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211-252. doi: 10.1007/s11263-015-0816-y

Sabour, S., Frosst, N. & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3856–3866).

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Shamir, O. (2018). Distribution-specific hardness of learning neural networks. *The Journal of Machine Learning Research*, *19*(1), 1135–1163.

Shamir, O., Sabato, S. & Tishby, N. (2010). Learning and generalization with the information bottleneck. *Theoretical Computer Science*, *411*(29-30), 2696–2711.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, *5*(1), 3–55.

Shannon, C. E., Weaver, W. & Burks, A. W. (1951). *The mathematical theory of communication.*

Shermer, M. (2011). *The believing brain: From ghosts and gods to politics and conspiracies—how we construct beliefs and reinforce them as truths*. Macmillan.

Shore, J. & Johnson, R. (1980). Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *Transactions on information theory*, *26*(1), 26–37.

Shorten, C. & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 60.

Shwartz-Ziv, R. & Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Skilling, J. & Bryan, R. (1984). Maximum entropy image reconstruction-general algorithm. *Monthly notices of the royal astronomical society*, *211*, 111.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Ieee winter conference on applications of computer vision (wacv)* (pp. 464–472).

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Smith, L. N. & Topin, N. (2019). Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications* (Vol. 11006, p. 1100612).

Snoek, J., Larochelle, H. & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Soltanolkotabi, M., Javanmard, A. & Lee, J. D. (2018). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, *65*(2), 742–769.

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

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A. & Le, Q. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2820–2828).

Tan, M. & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

Tishby, N. & Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)* (pp. 1–5).

Tsai, D.-Y., Lee, Y. & Matsuyama, E. (2008). Information entropy measure for evaluation of image quality. *Journal of digital imaging*, *21*(3), 338–347.

van Hemmen, J. L. & Zagrebnov, V. A. (1987). Storing extensively many weighted patterns in a saturated neural network. *Journal of Physics A: Mathematical and General*, *20*(12), 3989.

Van Ooyen, A. & Nienhuis, B. (1992). Improving the convergence of the back-propagation algorithm. *Neural networks*, *5*(3), 465–471.

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G. & Lacoste-Julien, S. (2019). Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in neural information processing systems* (pp. 3732–3745).

Veit, A., Wilber, M. J. & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems* (pp. 550–558).

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … Contributors, S. . . (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: https://doi.org/10.1038/s41592-019-0686-2

Wang, Z. & Bovik, A. C. (2002). A universal image quality index. *Signal processing letters*, *9*(3), 81–84.

Welvaert, M. & Rosseel, Y. (2013). On the definition of signal-to-noise ratio and contrast-to-noise ratio for fMRI data. *PloS one*, *8*(11), e77089.

Wernecke, S. J. et al. (1977). Maximum entropy image reconstruction. *IEEE Transactions on Computers*, *26*(4), 351–364.

Wiatowski, T. & Bölcskei, H. (2018). A mathematical theory of deep convolutional neural networks for feature extraction. *Transactions on Information Theory*, *64*(3), 1845–1866.

Wu, X., Ward, R. & Bottou, L. (2018). WNGrad: learn the learning rate in gradient descent. *arXiv preprint arXiv:1803.02865*.

Wu, Z., Shen, C. & Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, *90*, 119–133.

Xie, S., Girshick, R., Dollár, P., Tu, Z. & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1492–1500).

Yu, X., Efe, M. O. & Kaynak, O. (2002). A general backpropagation algorithm for feedforward neural networks learning. *Transactions on neural networks*, *13*(1), 251–254.

Zagoruyko, S. & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, X., Yu, Y., Wang, L. & Gu, Q. (2019). Learning one-hidden-layer relu networks via gradient descent. In *The 22nd international conference on artificial intelligence and statistics* (pp. 1524–1534).

Zhang, X., Zhou, X., Lin, M. & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6848–6856).

Zhang, Z., Song, Y. & Qi, H. (2017). Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5810–5818).

Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2881–2890).

Zhao, J., Liang, J.-m., Dong, Z.-n., Tang, D.-y. & Liu, Z. (2020). Accelerating information entropy-based feature selection using rough set theory with classified

nested equivalence classes. *Pattern Recognition*, *107*, 107517.
Zheng, K. & Wang, X. (2018). Feature selection method with joint maximal information entropy between features and class. *Pattern Recognition*, *77*, 20–29.

Appendix A

Notations

Data distribution $\mathcal{D}$ The underlying data distribution from which the independent input vectors $\mathbf{x}_i$ are sampled.

Input Space The input dataset $\mathbf{X}$ consists of n number of input images denoted as $\mathbf{x}_{1-n}$ sampled from an underlying data distribution $\mathcal{D}$. Each of the n number of images $\mathbf{x}_{1-n}$ has an associated class label y_{i-n}, whose aggregated collection is denoted as Y, such that, $\mathbf{x}_i, y_i \in \mathbf{X} \times Y$.

Output Space $\hat{Y}$ is the computed class probability in the interval [0,1] for Y using $\mathbf{X}$.

Multi-Dimensional Data The images are interpolated and multi-dimensional (constrained to a finite d-dimensions), represented as $\mathbf{x}_{1-n}^{1-d}$.

Underlying functional representation The underlying functional representation of $\mathbf{X}$ can be denoted by some function f, which maps the input to the output for each of the n number of images i.e. $f : \mathbf{X} \rightarrow Y$.

Computed Approximation Function $\hat{f}$ is the computed approximation function calculated from the input dataset $\mathbf{X}$ with the ground truth Y.

Search or Problem Space The underlying function f for $\mathbf{X}$ could be any of the finite set of possible functions $\mathcal{F}$ which is the search space.

Solution Space Every possible function in $\mathcal{F}$ has an associated solution in the solution

space $\mathcal{Y}$. Ideally, $\hat{f} = f$ therefore, $\hat{Y} = Y$ for $\mathbf{X}$, this solution is the global optimum. If $\hat{f}$ is approximated to any of the possible functions in $\mathcal{F}$ other than the glabal optimum, it has converged to a local optimum.

Non-Convex Optimization Problems Non-convex optimization problems can be characterized by the presence of multiple local optima (minima for loss minimization functions and maxima for generalization functions) and the optimization of non-convex functions can converge to any local optima.

Loss Function The loss function $L(\hat{Y}, Y)$ is used to quantitatively determine the error in approximation between $\hat{f}$ and f.

True Functional Approximation $\hat{f}$ is computed over $\mathcal{F}$ and not the actual underlying function f since f is a hypothetical function which exists in the set of all possible functions $\mathcal{F}$. Therefore, a neural network or any traditional function solver must traverse the whole search space which exponentially increases as a function of complexity and converge to an approximation which is generally a local optimum. True convergence to the global optimum is theoretically possible but, it is impractical in the real-world due to noise and representational limitations of digital data.

Function Mapping Computing algorithms traverse the search space (i.e. learn) by mapping/evaluating the approximated function $\hat{f}$ using $\mathbf{X}$ and generating $\hat{Y}$, subsequently calculating the error $L(\hat{Y}, Y)$ utilizing the ground truths and minimizing this error by adjusting the internal parameters to converge to a global optimum and achieve a closer approximation.

Mathematical Representation of The Human Brain Humans have excellent pattern recognition skills developed through decades of evolution, mimicking nature, mathematical abstractions of the brain's synapses and neural architecture was proposed dubbed Artificial Neural Networks (ANNs). ANNs have neurons with weights ($\mathbf{W}$) to signify the strength of the synapses and interconnected hidden

layers to achieve the goal of function mapping, illustrated in Figure 2.1.

ANN Notations The total number of hidden layers or depth of the network is denoted as H. The total number of neurons in any given hidden layer $h_i|h_i \in H$ is denoted as χ. Collection of all the weights for a given hidden layer is represented as $\mathbf{W}_i|\mathbf{W}_i \in \mathbf{W}$ with individual neural weights of a given layer as $\omega_{i-\chi}|\omega_{i-\chi} \in \mathbf{W}$.

Parameter Space The total number of trainable parameters for a neural network model usually **W** and other HyperParameters (HPs) discussed below.

Functional Optimization Optimizing the computed approximation function $\hat{f}$ is carried out by constraining $\hat{f}$ to only differentiable functions allowing the computer to compute gradients ∇ and using these gradients to update internal parameters.

Non-Linearty An ANN performs well for linear approximations but, since $\mathcal{F}$ contains non-linear functions, a method to introduce non-linear functional approximations are needed. Non-linear activations are introduced to allow for closer approximations to $\mathcal{F}$. Non-linear activation functions σ such as the commonly used Rectified Linear Unit (ReLU) thresholds all negative value to zero and behaves as a linear function for all positive inputs.

Back-Propagation Weight adjustments/updates for **W** to minimize the loss function L are performed using the first order partial derivatives of $\hat{f}$ to achieve closer approximations to $\mathcal{F}$. The chain-rule is applied to compute the gradients θ for $\hat{f}$ provided the inputs **X**, Y.

Learning Rate The learning rate η for a neural network is the scaling factor for weight updates which determines the rate of convergence to closer functional approximations.

Stochastic Gradient Descent (SGD) SGD is the algorithm most commonly used to optimize the various HPs of the neural network such as the learning rate, gradient step-size i.e. the direction and magnitude for traversing the problem space.

Cnvolutional Neural Network (CNN) A CNN typically consists of a convolutional

block which uses the convolution operation to extract feature information along with a classification/fully connected block, which is a traditional ANN as illustrated in Figure 2.2.

Neural Breadth (χ) and Convolutional Breadth (χ') The number of neurons in a given hidden layer and convolutional filters/kernels/units in a given hidden layer respectively.

Neural Network Depth (H) and Convolutional Depth (H') The number of hidden layers in a given NN model and the number of convolutional layers in a given CNN model respectively.

Weight vector W The collection of weight matrices associated with all (breadth×depth) the neurons, convolutional kernels/units/channels $omega_i$

Feature Collection The collection of feature vectors D extracted using the convolutional block in a CNN.

Overfitting A function mapping method of converging to a closer approximation has a major drawback of learning the underlying functional representations for only the given input space. The error is minimized to such a degree that when a new set of inputs (more generalized) are provided which is close to the original input images (as an example, assume different species of cats or dogs are provided. Humans can easily distinguish these new images to their respective categories as cats or dogs) the computer fails to recognize these new images. The error in recognition/classification accuracy is the generalization error.

Regularization To mitigate overfitting, regularization is used to optimize some parameters such as complexity of the model and feature weights which allow for better generalization to unseen examples.

Dimensionality Reduction As image data is d-dimensional, reducing the dimensions could reduce the search space significantly and allow for faster and accurate

convergence. Dimensionality reduction is achieved in a CNN employing numerous lower-dimensional convolutional kernels which extract a single feature but, when multiple of these convolutional kernels are utilized in single layer, multiple feature vectors can be extracted. More complex features are abstracted by increasing the convolutional layers which take as an input the previous layer's feature extractions rather than the input image allowing for hierarchical feature abstractions.

Feature Space The lower-dimensional subspace of the search space which contains different functional representations of the input space.

Vector Space A vector space is a collection of vectors, which may be added together and multiplied by scalars.

Lower-Dimensional feature vector ϕ The hypothetical vector space which can precisely reduce the input space linearly to the output space. In other words, the collection of all the vectors extracted from the feature space which can accurately reduce the input space to the output space.

Feature Map Vector (φ) The collection of all feature map vectors generated using the feature vectors ϕ to compute closer approximations to f from all possible functional representations in $\mathcal{F}$.

Training afflictions The obvious problem with stacking multiple convolutional layers in series is that computing gradients for the initial layers becomes challenging since error propagation and weight updates are performed from the final layer to the initial layers. This could lead to afflictions like vanishing and exploding gradients, where the computed gradients are either insignificant so that they can essentially be ignored or large enough that it would induce model divergence.

Residual or Skip Connections To enhance error propagation to the initial layers, residual connections which allow for gradients to skip intermediate hidden layers and in the later layers perform identity mapping. These connections enable very

deep networks to be trained.

Regression Vector (ϑ) A linear vector at the final classification layer (predominantly SoftMax) which separates the computed lower-dimensional feature vectors $\phi(X)$ using the n number of input image vectors $\mathbf{x}_{i-n}|\mathbf{x}_i \in \mathbf{X}$.