Towards Robust Convolutional Neural Networks in Challenging Environments

Md Tahmid Hossain

A dissertation submitted in fulfilment of the requirements for the degree of **Doctor of Philosophy**

School of Engineering, Information Technology and Physical Sciences
Federation University, Australia

27 September 2021

I certify that except where due acknowledgement has been made, that this dissertation is of the author alone; includes nothing, which is the outcome of work done in collaboration except where specifically indicated in the text; has not been previously submitted, in part or whole to any university or institution to qualify for any degree, diploma, or other qualification; that the content of the dissertation is the result of work that has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and that ethics procedures and guidelines have been followed.

Md Tahmid Hossain

27 September 2021

Shyh Wei Teng (Principal Supervisor)

27 September 2021

Dedicated to my parents and brothers for their love and encouragement.
Dedicated to my wife for her patience, care and relentless support from the inception to completion of this dissertation.

Acknowledgements

Praise be to Allah, the most gracious, the most merciful, who blessed me with the intellect, courage and patience to undertake this research. My sincerest gratitude to my supervisors Associate Professor Shyh Wei Teng, Professor Guojun Lu and Associate Professor Ferdous Sohel. I express my profound indebtedness to them for their tireless efforts in ensuring the quality of my research and guiding me in the right direction. Without their constant inspiration, encouragement and mentoring this research would not have been completed. My heartfelt love and gratitude to my parents Professor Md. Akhtar Hossain and Mrs. Dilruba Hossain, for their inspiration, mental support and keeping me in their prayers. My elder brother, Tauhid Hossain, has always guided me in the best possible way and never shied away from going the extra mile for me. My eldest brother, Tanvir Hossain, has always been a role model who never fails to set shining example in everything he does. My elder sisters Dr. Fouzia Kashem and Tamanna Zaman have always looked after me through the ups and downs. My loving nephews Tawseef and Arish have always inspired me in their own joyful way. I am also thankful to my parents-in-law Abu Taher and Mrs. Sara Lulufar Banu for their persistent love and support. My joyful sisters Tasnuva Sharmin, Anika Tahsin, and Saraban Tahura have always been there to inspire me in my journey.

It is hard to express the patient, steady and generous support I received from my wife, Tasfia Shermin. Despite being on her own PhD journey, Tasfia provided endless support at times when it mattered the most. I thank my friends, Jadeed, Junaid, Navid, and Shounak who still support me in every possible way. I thank Tahmid Rahman for numerous research discussions. I thank Professor Manzur Murshed for his constant support and insights in research discussions.

Md Tahmid Hossain was supported by an Australian Government Research Training Program (RTP) RTP Fee-Offset Scholarship through Federation University Australia. Capstone Editing provided assistance in editing this thesis.

Abstract

Image classification is one of the fundamental tasks in the field of computer vision. Although Artificial Neural Network (ANN) showed a lot of promise in this field, the lack of efficient computer hardware subdued its potential to a great extent. In the early 2000s, advances in hardware coupled with better network design saw the dramatic rise of Convolutional Neural Network (CNN). Deep CNNs pushed the State-of-The-Art (SOTA) in a number of vision tasks, including image classification, object detection, and segmentation. Presently, CNNs dominate these tasks.

Although CNNs exhibit impressive classification performance on clean images, they are vulnerable to distortions, such as noise and blur. Fine-tuning a pre-trained CNN on mutually exclusive or a union set of distortions is a brute-force solution. This iterative fine-tuning process with all known types of distortion is, however, exhaustive and the network struggles to handle unseen distortions.

CNNs are also vulnerable to image translation or shift, partly due to common Down-Sampling (DS) layers, e.g., max-pooling and strided convolution. These operations violate the Nyquist sampling rate and cause aliasing. The textbook solution is low-pass filtering (blurring) before down-sampling, which can benefit deep networks as well. Even so, non-linearity units, such as ReLU, often re-introduce the problem, suggesting that blurring alone may not suffice.

Another important but under-explored issue for CNNs is unknown or Open Set Recognition (OSR). CNNs are commonly designed for closed set arrangements, where test instances only belong to some 'Known Known' (KK) classes used in training. As such, they predict a class label for a test sample based on the distribution of the KK classes. However, when used under the OSR setup (where an input may belong to an 'Unknown Unknown' or UU class), such a network will always classify a test instance as one of the KK classes even if it is from a UU class.

Historically, CNNs have struggled with detecting objects in images with large difference in scale, especially small objects. This is because the DS layers inside a CNN

often progressively wipe out the signal from small objects. As a result, the final layers are left with no signature from these objects leading to degraded performance.

In this work, we propose solutions to the above four problems. First, we improve CNN robustness against distortion by proposing DCT based augmentation, adaptive regularisation, and noise suppressing Activation Functions (AF). Second, to ensure further performance gain and robustness to image transformations, we introduce anti-aliasing properties inside the AF and propose a novel DS method called blurpool. Third, to address the OSR problem, we propose a novel training paradigm that ensures detection of UU classes and accurate classification of the KK classes. Finally, we introduce a novel CNN that enables a deep detector to identify small objects with high precision and recall. We evaluate our methods on a number of benchmark datasets and demonstrate that they outperform contemporary methods in the respective problem set-ups.

Contents

List of Figures

List of Tables

Acronyms

AA Anti Aliasing

AF Activation Function

ANN Artificial Neural Network

AUROC Area Under Receiver Operating Characteristic

BNS Batch Normalisation Statistics

C-ReLU Clipped ReLU

C-ReLU Clipped ReLU

C-ReLU Clipped Rectified Linear Unit

C2AE Class Conditioned Auto-Encoder

CAE Convolutional Autoencoder

CNN Convolutional Neural Network

CS Confidence Subnetwork

CS Cosine Similarity

DAB-Pool Depth Adaptive Blur-pool

DCT Discrete Cosine Transform

DS Down-sampling

EVT Extreme Value Theory

EV Encoded Vector

FC Fully Connected

FFB Fundamental Functional Block

FP Flip Probability

GAN Generative Adversarial Network

GSA Grid Search Attack

HFc Higher Frequency corruption

HVS Human Visual System

KK Known Known

KUT Known Unknown Trainer

KU Known Unknown

LFc Low Frequency corruption

LP-ReLU Low Pass ReLU

MBP Max Blur-Pool

MMD Maximum Mean Discrepancy

OOD Out-of-Distribution

OSR Open Set Recognition

PSNR Peak Signal-to-Noise Ratio

RKHS Reproducing Kernel Hilbert Space

RPN Region Proposal Network

ReLU Rectified Linear Unit

SABP Spatially Adaptive Blur-Pool

SGD Stochastic Gradient Descent

SIFT Scale Invariant Feature Transform

SOTA State-of-the-art

SURF Speeded-Up Robust Features

UU Unknown Unknown

i.i.d. independent and identically distributed

Chapter 1

Introduction

Image classification is one of the fundamental tasks in computer vision and is at the core of more complex tasks such as deep object detection. It is a process of assigning target labels to an image based on the visual information available in the pixels. Real-world applications such as autonomous driving, facial recognition, visual searching, and medical diagnostics, use image classification in some form or another.

With the recent advances in deep learning and Convolutional Neural Network (CNN), the State-of-The-Art (SOTA) in image classification is regularly reaching new heights. However, CNNs are still vulnerable to a broad array of challenges including distortion, transformation, and minuscule and unknown objects. Improving CNN robustness to these challenges is, therefore, an important domain of research which we address in this work.

1.1 Motivation

In pursuit of improving the classification accuracy on clean benchmark datasets such as ImageNet [1], CNNs' vulnerability to some common challenges is under-explored. For example, for humans, the presence of minor distortion in an image does not affect our classification ability. Therefore, intelligent classifier machines such as a CNN are also expected to correctly classify images with minor distortion. However, in practice, even SOTA CNNs falter and misclassify such images. Likewise, trivial image transformations such as shift are another challenge that should not dramatically affect CNNs' performance. The presence of an unknown class image should not trigger irrational output. Small-scale objects that are human recognisable should be recognised

by a CNN as well. In reality, however, all of the above mentioned challenges occur frequently and induce misclassification. This raises serious credibility concerns, which motivate us to revisit the functional blocks of a CNN and its training paradigm to increase its robustness.

One of the most popular ways of improving robustness against distortion is data augmentation. At its core, it is a process of interpolating new data from the training set to represent potential variations in the test set. This typically requires a heuristic knowledge of the test set, e.g., if the test images are likely to be noisy, different types of noise-based augmentation can be used during training. However, in reality, an image can get distorted in numerous ways and in a conventional augmentation approach, each of the distortion has to be explicitly introduced in the augmented dataset. This is a time-consuming brute-force solution and requires prior knowledge of the expected distortions. In this work, we propose a novel augmentation methodology based on Discrete Cosine Transform (DCT) that can address a wide range of distortions without any prior knowledge. We also improve an adaptive Drop-out regularisation technique for further performance gains.

While augmentation improves robustness to some degree, our investigation shows that the fundamental functional blocks of a CNN, such as Activation Functions (AFs) and Down-Sampling (DS) layers lack noise suppression properties. For example, Rectified Linear Unit (ReLU) is the most popular AF which is unbounded in nature and does not suppress noisy signal. Max-pooling, a popular DS method, does not account for the Nyquist sampling theorem and thus leads to signal aliasing. This results in performance degradation not only on distortions but also on trivial image transformations such as shift. Improving these fundamental functional blocks can lead to substantially better performance against all these challenges. Based on our observations, in this work, we strive to further improve the robustness of CNNs by proposing novel AFs and redefining the DS methods.

Data augmentation and improved fundamental functional blocks enable CNNs to achieve better classification results on challenging distorted datasets. However, regardless of a model's robustness, the closed set nature of conventional classification completely ignores the open set samples or 'Unknown Unknown' (UU) classes. As a

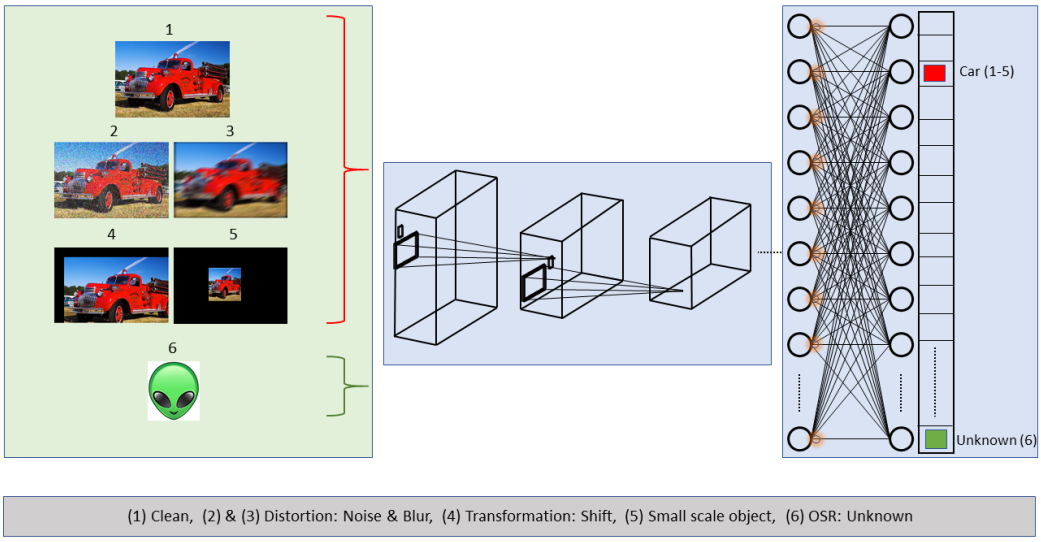

Figure 1.1: Illustration of research objectives. In this thesis, our aim is to develop a deep CNN that can correctly classify not only clean images (1), but also images with distortion (2 and 3), transformation (4), small scale object (5), and totally unknown images (6).

result, a CNN always assigns a 'Known Known' (KK) class to any input, even if it is from a UU class. Training a network with 'everything else' in the world as one extra 'other' class is unrealistic. In this work, we design a novel training paradigm where a network can identify unknowns while maintaining high classification accuracy for the knowns.

As mentioned earlier, deep detectors use CNN classifiers as the backbone of the network. Any vulnerability in a CNN classifier, by extension, is a vulnerability for a detector. Our investigation in this work shows that a CNN trained to classify and detect objects of moderate size performs well on medium to large objects but struggles detecting smaller ones. Our experiments in this regard show that the signal from such small objects often gets attenuated along the way in a typical feed-forward CNN. To address this challenge, in this work, we design a novel backbone CNN that prevents signal from getting lost, even if the source object is small in scale.

1.2 Research Objectives

This research aims to develop a set of techniques for CNNs so that they:

- are robust to image distortions such as noise and blur.
- are robust to image transformations such as shift.
- show robustness in the presence of UU classes.
- show robustness in the presence of small objects.

1.3 Contribution Overview

As illustrated in Figure 1.1, our aim is to develop CNNs that are robust to a variety of distortions, transformations, and object scales. In addition, unknown samples should be detected at test time without compromising accuracy for the known samples. The contributions made in this work can be summarised as follows:

- Develop a novel data augmentation method and improve regularisation to enhance CNN's robustness against distortion. (Chapter 3)
- Further improve distortion robustness by proposing an AF with built-in low pass filtering. This AF complements the DCT augmentation and provides robustness against a wide array of distortions and perturbations. (Chapter 4)
- Propose a novel AF and DS method by instilling anti-aliasing properties in them. This further improves performance on a variety of image distortions and transformations. (Chapter 5)
- Develop a novel training paradigm capable of open set image recognition. This includes a novel confidence sub-network (CS) that detects unknowns and enables the base classifier to classify knowns with high accuracy. (Chapter 6)
- Analyse why CNNs are susceptible to detecting small objects and propose a novel backbone CNN for deep detectors. This backbone enables a smooth signal propagation for small objects and ensures even small objects are detected at test time alongside medium and large ones. (Chapter 7)

1.4 Publications

The following peer-reviewed and under-review publications arose from this thesis:

- M. T. Hossain, S. W. Teng, D. Zhang, S. Lim and G. Lu, "Enhancing the Effectiveness of Local Descriptor Based Image Matching," *in 2018 Digital Image Computing: Techniques and Applications (DICTA), 2018, pp. 1-8, doi: 10.1109/DICTA.2018.8615800.* [Link]

- M. T. Hossain, S. W. Teng, D. Zhang, S. Lim and G. Lu, "Distortion Robust Image Classification Using Deep Convolutional Neural Network with Discrete Cosine Transform," *in 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 659-663, doi: 10.1109/ICIP.2019.8803787.* [Link]

- M. T. Hossain, S. W. Teng, and G. Lu,"BackNet: An Enhanced Backbone Network for Accurate Detection of Objects with Large Scale Variations," *in 2019 Pacific-Rim Symposium on Image and Video Technology (PSIVT), 2019, pp. 52-64, doi:10.1007/978-3-030-34879-3_5.* [Link]

- M. T. Hossain, S. W. Teng, F. Sohel and G. Lu, "Robust Image Classification Using a Low-Pass Activation Function and DCT Augmentation," *IEEE Access, vol. 9, pp. 86460-86474, 2021, doi: 10.1109/ACCESS.2021.3089598.* [Link]

- M. T. Hossain, S. W. Teng, F. Sohel and G. Lu, "Anti-aliasing Deep Image Classifiers using Novel Depth Adaptive Blurring and Activation Function." *Submitted to IEEE Transactions on Multimedia.* [Under Review]

- M. T. Hossain, S. W. Teng, F. Sohel and G. Lu, "A novel network training approach for open set image recognition." *Submitted to Computer Vision and Image Understanding*. [Under Review]

1.5 Thesis Structure

In this section, we outline the organisation of the thesis chapters. Chapter 2 contains the relevant literature review. It first discusses two different approaches to image

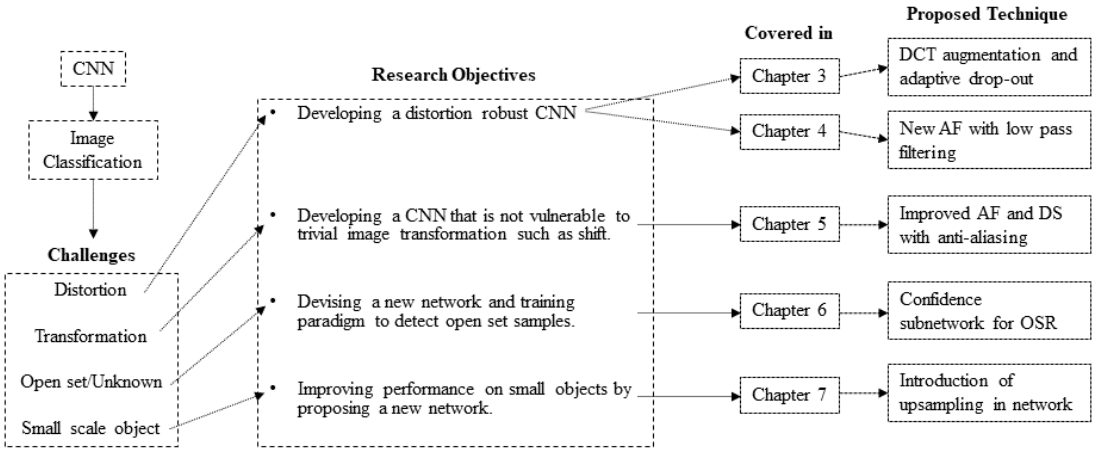

Figure 1.2: Contributions and structural overview of the thesis.

classification namely hand-crafted and machine learning. After providing a brief details of two prominent hand-crafted methods, the basics of machine learning based CNNs are discussed. To this end, an overview on the evolution of CNNs is provided to highlight what challenges have influenced the design choices of these networks over time. Next, we discuss four existing challenges that we want to address in this work.

A schematic diagram of our RO and the corresponding chapters are presented in Figure 1.2. We address CNNs' vulnerability to distortion in Chapters 3 and 4. While Chapter 3 explores an augmentation and regularisation based solution, Chapter 4 investigates ways of improving existing AFs for improved robustness.

Chapter 5 puts emphasis on solving the aliasing problem in existing CNNs, which is partially responsible for their vulnerability to distortion and corruption. Here, conventional AF and DS are redefined to serve as anti-aliasing units.

Chapter 6 analyses the importance of Open Set Recognition (OSR) and presents a novel network training paradigm to detect unknown samples at test time. To achieve this, a novel network that we call OSRNet is proposed that has a dedicated Confidence Sub-network (CS) to deal with UU classes.

Chapter 7 analyses the reasons behind CNN's lack of accuracy in classifying small scale objects. To this end, we analyse the feature map foot-prints from such objects and propose a new network architecture with feature upsampling layers. We call this

network BackNet as we only modify the backbone of the network. BackNet shows demonstrated improvement in dealing with small scale objects.

Chapter 8 discusses our research outcomes and provides insight on the compatibility of the methodologies proposed in the preceding chapters.

Chapter 2

Literature Review

Image classification is an important computer vision problem that has been extensively studied. Over the years, the methods used for image classification and the challenges associated with these methods have evolved enormously. In this chapter, we discuss a wide range of relevant methods in the literature. An outline of the content of this chapter is presented in Figure 2.1. Accordingly, the rest of the chapter is organised as follows:

Section 2.2 outlines the hand-crafted feature extraction based image classification. This includes a detailed overview of two widely used feature extraction methods namely Scale Invariant Feature Transform (SIFT) [2] and Speeded-Up Robust Features (SURF) [3]. In Section 2.3, we highlight why machine learning-based methods are receiving substantial attention followed by the details of a CNN-based classifier. This section sheds light on the fundamental functional blocks of a CNN. Section 2.4 discusses the evolution of CNNs over the last decade. This provides an overview of five popular CNN architectures and the inspiration behind their design choices. Since the goal of this thesis is to improve CNN's robustness under challenging circumstances, Section 2.5 presents a number of such existing challenges and the solutions found in the literature. We also outline the deficiencies in the existing solutions setting the context up for our contribution chapters. We draw conclusions from the literature review of this chapter in Section 2.6.

2.1 Image Classification

Image classification, i.e., categorisation of an image based on its content, has long been an important research domain in the field of computer vision. Prior to 2010,

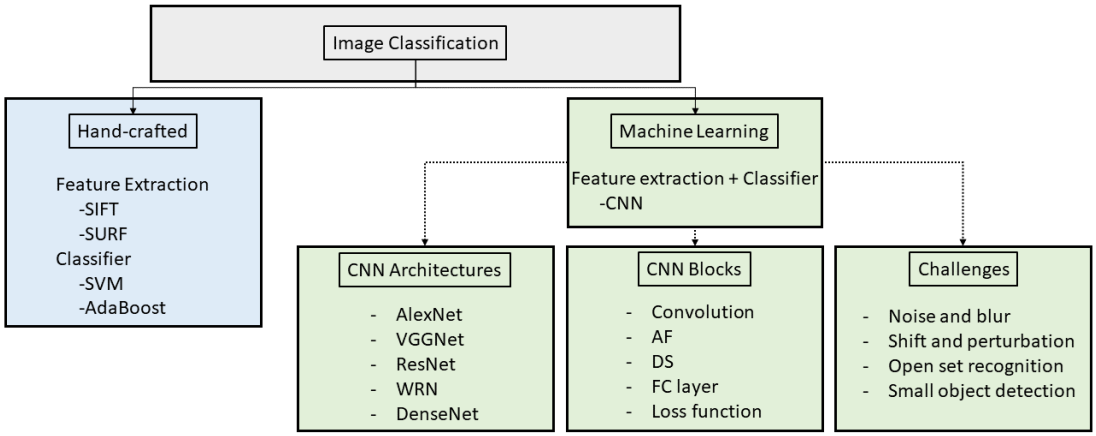

Figure 2.1: Chapter content outline.

image classification was heavily reliant on hand-crafted features. In such an approach, feature extraction is followed by classifying them in a conventional classifier, e.g., Support Vector Machine (SVM) or AdaBoost. Although hand-crafted features are still commonly used to perform a variety of tasks, machine learning-based feature extraction and classification has been proven to be more effective. In this chapter, we first discuss the hand-crafted feature based approach and then delve deeper into the machine learning-based approach for image classification.

2.2 Hand-crafted Approach

To classify an image, it has to be represented by a set of meaningful features. Hand-crafted feature extraction process refers to a set of predefined rules that determines how features are constructed from the raw pixels. What qualifies as a feature and the design choices for their construction solely depend on the Architect of the method and hence, these methods are collectively termed as hand-crafted. For example, Oliva et al. [4] built features of a scene from global properties such as roughness, naturalness, openness etc. Likewise, Hays et al. [5] used colour and texton histograms to build global features. As opposed to global features, local features involve extraction and description of small local patches of an image. These descriptors are hand-crafted as well and encoded in high dimensional vectors, e.g., 128-dimensional features in SIFT.

SIFT and SURF are two methods that have been widely used in a number of vision tasks including image classification. In this section, we will first provide details of SIFT and how it is used for classification. Later, we present a brief overview of SURF as well.

2.2.1 SIFT

Lowe et al. [2] proposed SIFT back in 2004. From a high level, SIFT extracts small patches from an image and builds a descriptor around these. These descriptors can be thought of as a set of features for a particular key point. Formally, building SIFT features requires the following steps: key point detection and localisation, dominant orientation assignment, and descriptor construction. These steps are described below.

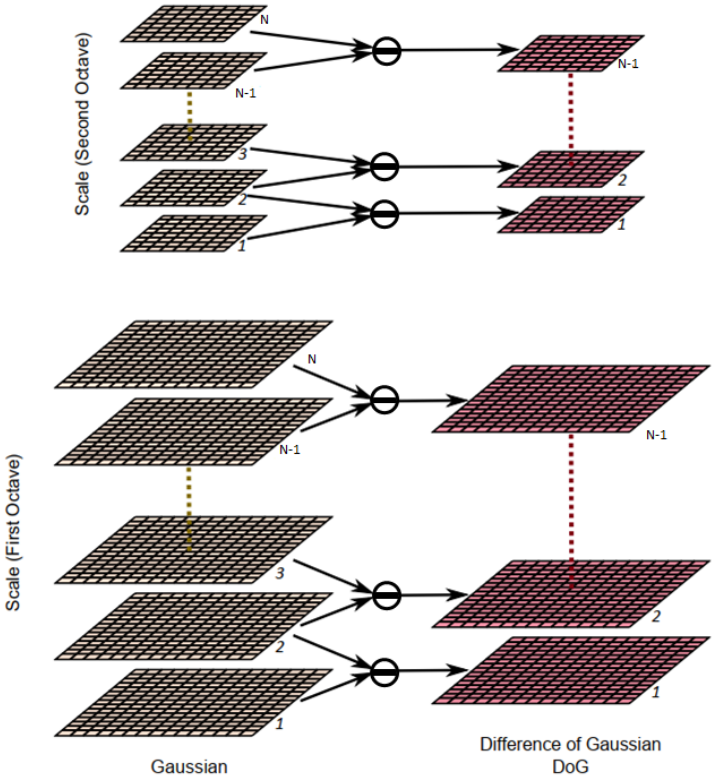

Figure 2.2: Construction of SIFT scale-space.

2.2.1.1 Detection and Localisation of Key Points

As depicted in Figure 2.2, in each octave, the initial image is repeatedly blurred using Gaussian filters. After each octave, the images is down-sampled by removing every alternate pixels making the resultant image half in the spatial dimensions. These sets of octaves are called the scale space. In SIFT, the scale-space has four octaves each with five increasing levels of blur. In order to find interesting points or key points in each octave, the Laplacian of Gaussian (LoG) of the blurred images is required. However, the computational overhead of LoG is too high and hence, instead of LoG, the Difference of Gaussian (DoG) for two consecutive images in an octave is calculated. DoG is a close approximation of LoG with substantially lesser computational complexity.

For each of the DoG image, a 3×3 block is taken and each pixel is compared to its 26 adjacent neighbours. A pixel is treated as a key point if it is either the Maxima or Minima of the neighbouring pixels. After iterating this process for rest of the pixels in the scale space, a set of key points are extracted for further processing.

2.2.1.2 Dominant Orientation Assignment

A SIFT descriptor is a numerical summary of magnitude-weighted gradient histograms. Gradients are found by observing the change of pixel intensity within an image. As gradients are not rotation invariant, a dominant orientation is found from the histogram and the descriptor is built with respect to that orientation. This enables the SIFT descriptor to be rotation invariant.

2.2.1.3 Descriptor Construction

SIFT descriptor is built on a 4-by-4 cell square spatial grid. It is centered around the corresponding key point and scale. The gradient and magnitude information of pixels in each cell in the grid is represented by an 8-bin orientation histogram, leading to a descriptor of $4 \times 4 \times 8 = 128$ dimensions. These descriptors are the features used later for classification and registration. SIFT also has a key point matching step, which we improved in [6] by introducing an adaptive clustering-based matching technique. In this thesis, we stick to our machine learning-based contributions and hence do not go into details of our work in [6].

2.2.1.4 Classification

Once an image is represented by a set of key points and their corresponding 128-dimensional feature vectors, the next step is to train a classifier such as SVM or AdaBoost to distinguish these features into different categories. However, in practice, the raw features are usually further encoded in a Bag-of-Visual Words (BoV) model [7] or Fisher Vector (FV) [8]. For example, Sanchez et al. [8] used SIFT-based FVs as final features. They used linear SVMs as the classifier and trained them with the features using Stochastic Gradient Descent (SGD). Kulkarni et al. [9] also used SIFT features but employed AdaBoost instead of linear SVMs as the classifier.

2.2.2 Speeded-Up Robust Features (SURF)

SURF is another efficient way of detecting and describing key points proposed by Bay et al. [3]. SURF has certain similarities with SIFT but differs in operational fundamentals. It detects key points by approximating Gaussian second order derivatives with box filters. SURF detector and descriptor operate on integral images to speed up the calculation. Moreover, rather than using gradients, SURF harnesses Haar wavelet response on a SIFT-like region of interest. SURF is computationally much faster than SIFT and achieves impressive repeatability and distinctiveness. SURF features can be used for classification in similar ways to SIFT. Nonetheless, it is not as robust as SIFT in challenging conditions such as change in illumination or image transformation.

2.3 Machine Learning Approach: CNN

CNNs are deep learning models that do not need any explicit engineering to learn what qualifies as a feature and what does not. A CNN, in pursuit of reducing error by optimising the cost function, learns both global and local features [10]. This, in essence, resembles how humans learn from mistakes, hence the name 'machine learning'. Interestingly, it extracts low level features such as colour, edge, and contours in the shallower layers and high level features such as a human face and car wheel in the deeper layers. In other words, a CNN initially learns low-level features and gradually

learns more complex set of features as the depth increases. The final dense layer features of a CNN are usually much higher in dimension compared to hand-crafted features which gives them greater expressiveness and distinguishability. This is why machine learning-based classifiers have started outperforming hand-crafted methods.

A CNN, in its simplest form, is a function that takes an image as input and maps it to an output class. In other words, $\mathcal{F}(x) \rightarrow y$ can represent a CNN function, where x represents the input image and y denotes the predicted class x belongs to.

From a functional point of view, a CNN is a composition of multiple modular units that can extract features from images and classify the images in an end-to-end fashion. At the core of the feature extraction process is the convolution operation run by learnable filters. Once the features are extracted, an Artificial Neural Network (ANN) performs the classification task.

Often, a CNN's capacity is associated with the depth of these layers. Greater depth usually results in richer features and better classification accuracy. However, a number of other functional units play a vital role in the overall performance. In the following section, we describe the fundamental functional blocks inside a CNN.

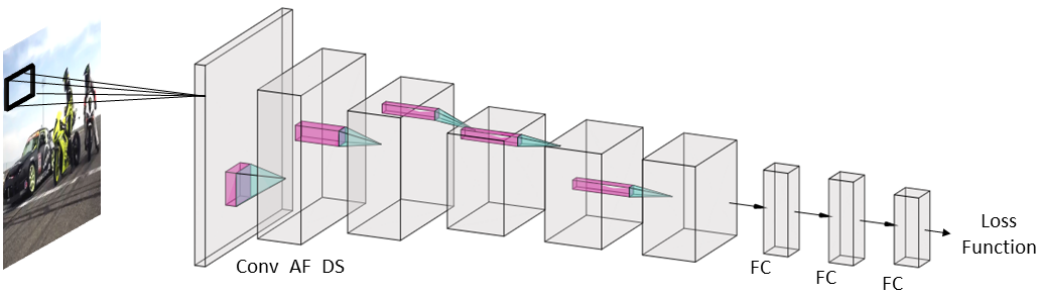

Figure 2.3: A standard CNN architecture outlining different fundamental functional blocks. In a conventional CNN, each convolution layer is followed by an AF. DS is used in-between some of the intermediate layers. FC layers are placed after the final convolution layer followed by the loss function.

2.3.1 CNN Basics

A CNN consists of a number of fundamental functional blocks, as depicted in Figure 2.3 In this section, we briefly discuss five such blocks.

2.3.1.1 Convolution

Convolution is a popular image processing technique that involves convolving an image with a predefined filter or kernel. The convolution operation in CNN, however, casts the fixed size kernel weights as learnable parameters. These parameters are randomly initialised but over the course of the training, the parameters are updated via back-propagation.

To put these in formal notation, let x be an input image and W be a kernel with a resolution of $m \times n$. Now we can define our convolution operation as follows:

$$O(i,j) = \sum_{p=1}^{m-1} \sum_{q=1}^{n-1} (x_{i+p,j+q}) \cdot (W_{p,q}) \tag{2.1}$$

where $O(i,j)$ is the output at location (i,j).

2.3.1.2 Activation Function (AF)

Adding non-linearities in the form of AFs has been an integral part of CNNs learning process. Initially, Sigmoidal functions were widely used but were later sidelined as they suffer from the vanishing gradient problem[1] [11]. Sigmoidal functions squash values within a finite range (typically $[0,1]$ or $[-1,1]$) which leaves saturation points on both sides making it impossible to find a slope in these regions. This halts the error backpropagation, and the network struggles to reach the global minima. This issue becomes particularly severe in very deep networks where the loss has to travel back a long way to have meaningful learning. ReLU does not suffer from the vanishing gradient problem and has long been the 'de-facto' AF in deep networks. We will next discuss the pros and cons of different AFs found in the literature. A pictorial depiction of these AFs is presented in Figure 2.4

ReLU. ReLU [12, 13] is a piece-wise linear monotonic function. It simply does not have any response on the negative side, and any positive value remains unchanged (see Equation 2.2).

[1] When a first order derivative of the AF cannot be calculated for some input, it is called the vanishing gradient problem.

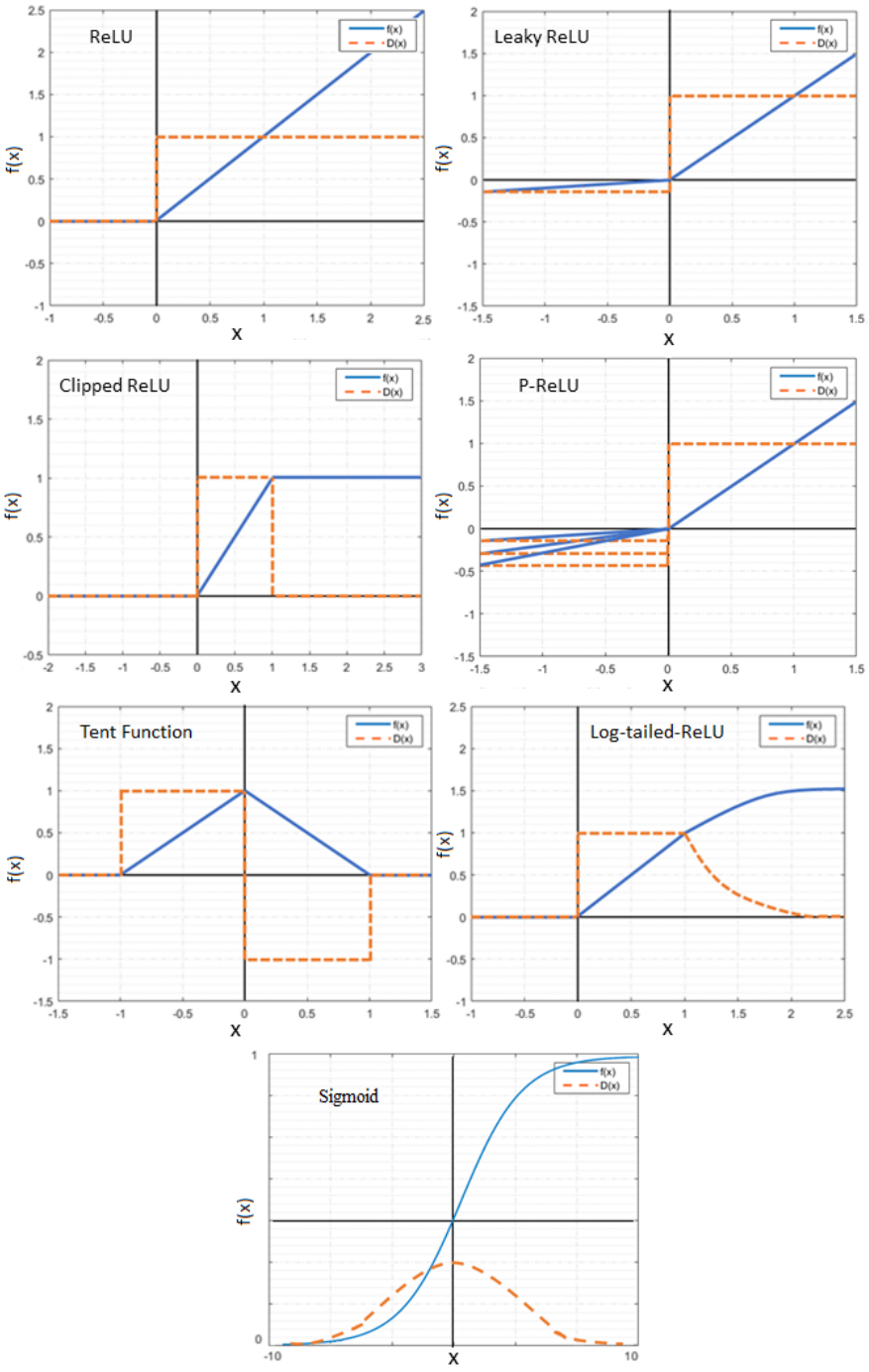

Figure 2.4: Activation functions

$$f(x) = \max(0, x) \tag{2.2}$$

ReLU is easy to compute [14], a derivative is available everywhere along with the positive range, which makes it a great choice for deep networks. It is also hypothesised to mimic the biological neuron firing process to justify its effectiveness. This correlation, however, is disputed in [15] based on the Integrate and Fire Model [16] in biological neurons. Unlike ReLU, where the output slope is always constant ($45°$, see Figure 2.4), according to the Integrate and Fire Model, the biological neuron's output slope varies depending on the resistance present in the cell membrane.

Although widely used, ReLU's unbounded nature leads to feature sparsity. This lets a marginally distorted signal drift away from its true distribution. This results in misclassification. Put simply, ReLU achieves impressive accuracy on clean data sets but is vulnerable to data corruptions.

Leaky ReLU and Parametric ReLU. Because ReLU has a zero response for negative values, a large number of neurons may never fire in the absence of proper initialisation, and hyper-parameter setup [17] resulting in 'dead neurons'. Leaky ReLU has been proposed [17] to address the dead neuron problem by introducing a small constant slope ($\alpha = .01$) on the negative side (see Equation 2.3). Leaky ReLU has a derivative on both sides of the origin (see Figure 2.4). In Parametric ReLU [18], the slope α is set up as a learnable parameter rather than a constant and accuracy improvement is reported [18]. However, leaky variants do not consistently outperform vanilla ReLU across datasets.

$$f(x) = \begin{cases} x, & x > 0, \\ \alpha x, & \text{otherwise} \end{cases} \tag{2.3}$$

Clipped ReLU. Clipped ReLU (C-ReLU) [19] is simply vanilla ReLU clipped to a constant based on a threshold A (Equation 2.4). C-ReLU was initially proposed in speech recognition [19] to increase training stability by avoiding gradient explosion. Later, it found its way into computer vision and other domains of deep learning as well. For example, C-ReLU's finite output range is found to be handy in applications where number representation capability is limited [20]. Despite C-ReLU's different set of use-cases, it incorporates a hard low pass filtering which leaves large saturation points in the bounded region (see Figure 2.4). Consequently, vanishing gradient remains a

problem, and there is no way to discriminate features beyond the threshold A.

$$f(x) = \begin{cases} \max(0, x), & 0 < x \leq A, \\ A, & x > A \end{cases} \tag{2.4}$$

Tent Function. Tent AF [21] is built from two ReLU units and it is symmetric around the origin (see Equation 2.5). It is designed to resist adversarial attacks but unlike ReLU, this AF is not monotonic and has large saturation regions on both sides beyond a threshold δ (see Figure 2.4).

$$f(x; \delta) = \max(0, \delta - |x|) \tag{2.5}$$

Log-tailed ReLU. As the name suggests, Log-tailed ReLU [22] is identical to ReLU up to a threshold A. The growth of the function is logarithmic thereafter $[A, \infty)$. However, this tail part rapidly converges to C-ReLU with increasing x as can be seen from Equation 2.6 and Figure 2.4.

$$f(x) = \begin{cases} 0, & x \leq 0, \\ x, & 0 < x \leq A, \\ A + \log(x - A), & x > A \end{cases} \tag{2.6}$$

Tanh Function. Hyperbolic tangent function or $tanh$ is sigmoidal in shape and squashes values inside a finite range of $[-1, 1]$ (see Equation 2.7). $tanh$ is centred around the origin. It has faster convergence compared to other sigmoidal functions because of the wider output range [23]. However, it still has large saturated regions on both sides of the origin.

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.7}$$

Swish. This AF [24] is represented by Equation 2.8. Here β is either a constant or a trainable parameter that defines the final shape of the function. For example, when β is close to 0, Swish becomes a scaled linear function $f(x) = \frac{x}{2}$. For increasingly large β, Swish starts imitating ReLU. However, regardless of the β value, the positive range of

Swish remains unbounded.

$$f(x) = x \cdot \sigma(\beta x), \quad \sigma(z) = (1 + \exp(-z))^{-1} \tag{2.8}$$

2.3.1.3 Downsampling (DS)

In a typical CNN, the number of convolutional filters increase with network depth and hence, the computational overhead increases too. One way to address this problem is to periodically downsample the intermediate feature maps by reducing their spatial dimensions. However, faster computation is not the sole purpose of DS. DS also provides translation invariance to the network. Figure 2.5 presents three of the most popular DS strategies in the literature– max-pooling (max-pool), average pooling (avg-pool), and strided convolution.

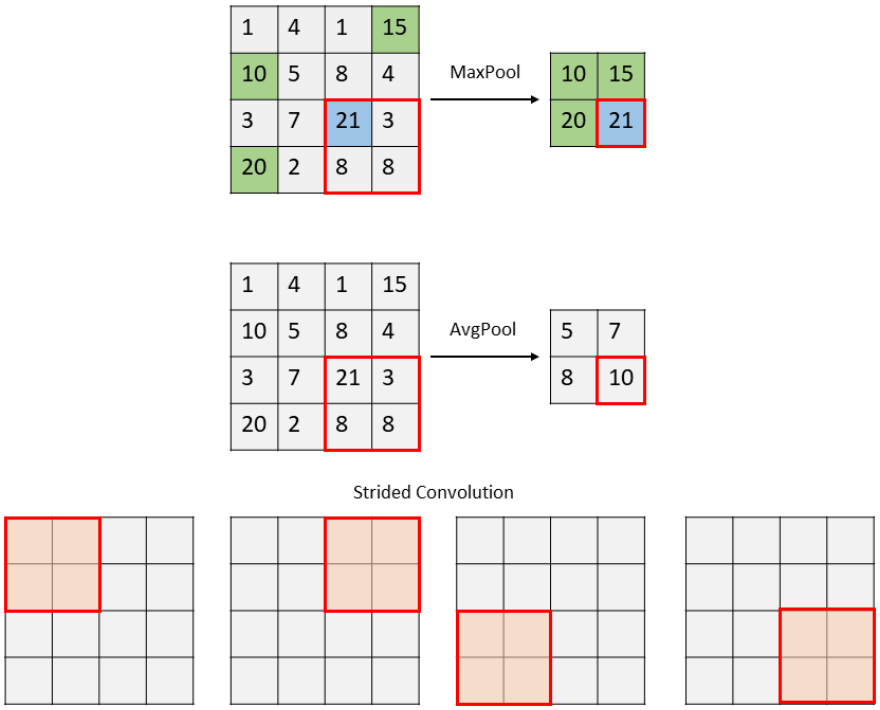

Figure 2.5: Different DS strategies. A stride size of 2 is used in all three examples.

As shown in Figure 2.5, max-pool operation selects the largest value in a predefined sub-grid. In max-pool, the degree of DS depends on the spatial dimension of the sub-grid (2×2 sub-grids are used in this example). Larger sub-grids result in a greater

DS factor. On the other hand, strided convolution refers to convolving a kernel with a stride greater than one. A stride of one preserves the original spatial dimension and larger stride results in a proportionately greater DS factor. Avg-pool is another DS method that operates in a similar way to max-pool. However, rather than choosing the maximum value in each sub-grid, it outputs the average (see Figure 2.5). As discussed later in Section 2.5.2, avg-pool does have some advantageous properties but cannot match the task performance of max-pool and strided convolution.

2.3.1.4 Fully Connected Layer

Fully Connected (FC) layers consist of artificial neurons analogous to a ANN. FC layers are usually followed by the last convolution layer in a CNN. ANNs with multiple hidden layers of neurons are called Multi-layer Perceptrons (MLP). One such perceptron or neuron's output can be represented as Equation 2.9.

$$y = \sigma(b_0 + \sum_{i=1}^{n} x_i w_i) \tag{2.9}$$

Here, x_i denotes the inputs from the previous layer and w_i denotes the corresponding weights. n is the number of total neurons in the previous layer. b_0 is the bias and σ is the AF. An MLP has a number of these units connected to each of the units in the previous and next layer.

From a high level, features extracted from the convolution layers enter the classification stage through the FC layer. The depth of FC layers is a design choice, e.g., AlexNet has two FC layers whereas ResNet has only one. It is worth noting that the dense connections among the FC layers and the preceding convolution layer lead to significantly high number of parameters. As a result, modern CNNs strive to keep FC layer depth to a minimum.

2.3.1.5 Loss Function

Loss function is the end point in a CNN where the prediction of the network is evaluated against the ground truth in a quantifiable manner. The task of the loss

function is to determine how 'far off' the prediction is for a given input. Cross-entropy loss is a popular choice for this task where the ground truth is one hot vector, e.g., in a 10-way classification task, the ground truth vector for one sample has 0s everywhere except in one index. The loss increases when the predicted probability distribution diverges from the ground truth distribution and vice-versa. Based on the back-propagation algorithm, higher loss transpires to greater change in the model parameters or weights. Since the weights are updated in the backward fashion, it is called back-propagation.

2.4 Evolution of CNN Architectures

Starting from AlexNet– the first deep CNN to win the ImageNet challenge, the architecture and training paradigm of deep networks have evolved significantly. At one point, CNN was able to surpass human level accuracy in the same classification challenge. Presently, CNNs are used in increasingly complex tasks including object detection and semantic segmentation. However, the improvements in classification accuracy on clean datasets do not proportionately transpire to better performance when datasets deviate from the original distributions– even when this deviation is visually imperceptible. This calls for a more robust learning paradigm that leads to CNNs capable of maintaining performance consistency even when the test distribution is not identical to the train distribution.

Before delving deeper into the challenges and existing solutions in Section 2.5, we provide an overview of five popular CNN architectures and their design principles. Figure 2.6 provides a pictorial depiction of these five networks.

2.4.1 AlexNet

AlexNet [14] consists of five convolution layers followed by max-pool layers in layer 1,2, and 5. Two FC layers (each with 4,096 neurons) follow the last convolution layer. A softmax layer takes in the input from the last FC layer and outputs a 1000-way class probability for the designated 1000 classes in ImageNet. The first convolution layer uses 11×11 filters, the second layer 5×5 filters, and the last three layers 3×3 filters.

For regularisation, the authors used data augmentation and drop-out. ReLU is used as the AF as opposed to sigmoid function.

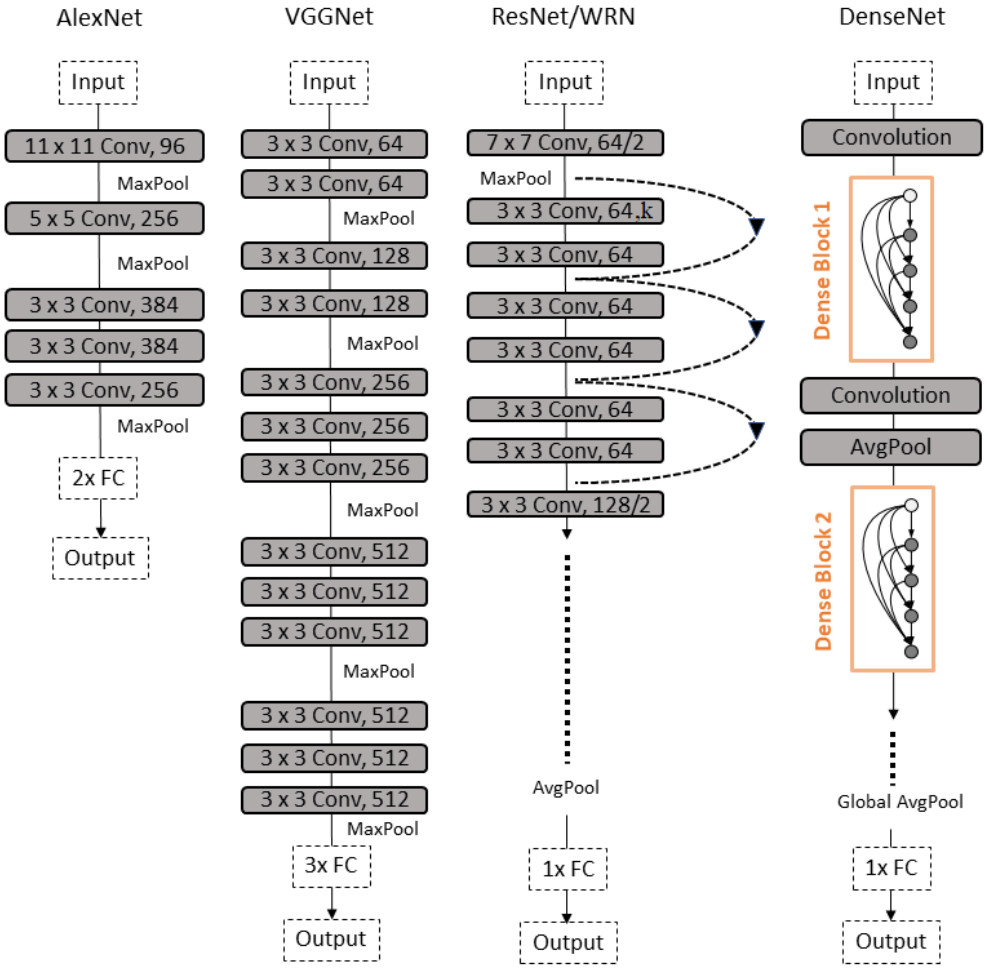

Figure 2.6: Architecture of AlexNet, VGGNet, ResNet/WRN, and DenseNet. The difference between ResNet and WRN is the widening factor k. $k = 1$ denotes ResNet and $k > 1$, e.g., $k = 8$ denotes a WRN with eight times more learnable filters in each convolution layer.

2.4.2 VGGNet

VGGNet [25] is a family of networks offering CNNs of varying depth including 11, 13, 15, and 19 learnable layers. VGG16 is the most famous one in the family with 13 convolution layers and three FC layers, as shown in Figure 2.6. It has 2×2 DS layers in between convolutions with stride 2. As opposed to 11×11 kernel with stride of 4 in the first convolution layer of AlexNet, VGGNet uses much smaller 3×3 kernels

with stride 1. This enables VGGNet to form and sustain larger feature maps at greater network depth. As a result, VGGNet is able to construct a richer and more complex set of features. This network architecture lets VGGNet achieve greater classification accuracy.

2.4.3 ResNet

The success of VGGNet prompted the realisation that deeper CNNs in general perform better than shallower ones. Thereafter, the focus shifted towards designing networks with more convolution layers. However, vanishing gradient problem and unnormalised features make it hard to train very deep CNNs and using smaller kernels like VGG16 does not suffice. ResNet [26] is the first deep CNN that enabled training networks with hundreds of layers with the help of skip connections and successful adoption of Batch Normalisation. A skip connection, in simple words, is a short-cut or direct path between two distant layers (see Figure 2.6). This path provides a way for smooth feature propagation and prevents gradients from getting vanished during back propagation. Batch Normalisation is used immediately after each convolution layer to keep the feature values within a range.

2.4.4 Wide ResNet

ResNet laid the foundation to train CNNs with hundreds of convolution layers. However, it has to incur a significant computational overhead as each convolution layer has to be followed by several other layers such as AF, DS, and Batch Normalization (BN). Moreover, in [27], it is argued that greater depth, after a certain level, does not provide substantial performance gain. To this end, they proposed Wide-ResNet (WRN) which cuts down the depth and uses wider convolution layers – all while using the same functional blocks as in a conventional ResNet (a widened layer has more convolution filters). WRN-40-8, for example, has 40 convolution layers, and 8 is the widening factor (considering ResNet has a widening factor 1). This means WRN-40-8 has eight times more filters in each layer as opposed to its equivalent ResNet. WRNs are faster to train and perform better.

2.4.5 DenseNet

While ResNet, and by extension WRN, get around the vanishing gradient problem by incorporating skip connections between two distant layers, DenseNet [28] goes one step further and proposes a network with even denser skip connections among the convolution layers. To be specific, in DenseNet, a dense block consists of several convolutional sub-blocks. To maintain the feed-forward nature, feature maps within a sub-block are concatenated and fed forward to all the subsequent sub-block layers. This helps a smooth flow of features from shallow to deeper layers of the network. During loss back-propagation, the dense connections ensure the gradients can easily flow backwards with getting vanished.

2.5 Challenges in Image Classification

Most of the benchmark datasets in image classification provide clean samples for training and testing, e.g., ImageNet [1]. This paradigm, although led to rapid developments in CNN architecture, left a caveat in the form of degraded performance on challenging test samples. As discussed in Chapter 1, real world applications are expected to face such challenges in the form of distortions, transformations, unknown samples, and small-scale objects. In this section, we discuss the literature relevant to these challenges and proposed methods to overcome them.

2.5.1 Noise and Blur

Dodge et al. [29] evaluated SOTA CNNs on distorted images. They found popular CNNs such as VGG16, GoogleNet, and Resnet, despite high classification accuracy on ImageNet, fail to perform well on distorted datasets. Nguyen et al. [30] further showed that often these distortions are visually imperceptible but the network misclassifies such samples as something else with surprisingly high confidence – some as high as 99.99%.

Upon realising CNNs' susceptibility to distortions and corruptions, Hendrycks et al. [31] recently published a range of benchmark datasets with 19 common corruptions.

A simple fix to this problem is to train a CNN with all the expected distortions [32, 33]. Some works [31, 34] also explored adversarial training to strengthen CNN's robustness against such corruptions but it is argued in [35] that adversarial training does not provide substantial defence against common corruptions. In this section, we provide an overview of different proposed methods for improving CNN robustness to such distortions and corruptions.

Data Augmentation. Vasiljevic *et al.* [32] followed the simplest augmentation approach for blur distortion and achieved reasonable performance after training a CNN with half clean and half blurred images. Similarly, Zhou *et al.* [33] improved performance by augmenting the dataset with both noisy and blurry images.

AutoAugment [36] is a more recent augmentation method, initially designed to boost clean performance. It incorporates a dynamic choosing from a pool of image processing policies during training. Recently, AutoAugment is found to be effective in improving CNN's robustness against common distortions as well [37, 38]. Hendrycks et al. [38] improved on AutoAugment and proposed AugMix. In AugMix, a series of different augmentation techniques (e.g., rotate, translate, and flip) is applied to an image one after another. The number of steps in a particular series determines how far an image drifts with respect to the original image. These images are argued to be closer to the original images, unlike some of the other works [39, 40, 41, 42]. For instance, in CutOut [42], a random portion of an image is occluded. In CutMix [40] and MixUp [41], image patches are interchanged, i.e., a portion of one image is overlaid on another. Images augmented via CutOut, MixUp, and CutMix appear quite unrealistic. It is worth noting that some of the augmentation techniques overlap with the test set. For instance, Gaussian and Speckle noise augmentations appear in [35, 33] despite their presence in the test data. In Chapter 3, we propose a novel DCT-based augmentation method and provide a detailed comparative performance evaluation.

Network Modification. Dodge *et al.* [43] proposed a mixture of experts-based model called MixQualNet for robust image classification. It consists of independent expert networks each trained on a particular type of distortion. A gating network is trained to select the most appropriate expert network at test time. MixQualNet performs

better compared to augmentation applied on a single CNN [33, 32]. However, it is an ensemble of N identical CNNs where N is the number of distortion types the model is trained on. Such ensembling makes the entire model parameter heavy. Moreover, it has one million additional gating network parameters, all resulting in a slow training process. Diamond *et al.* [44] and Yim *et al.* [45] took different routes and proposed additional network layers to undistort the images by denoising and deblurring. To undistort an input, one has to know the distortion characteristics in advance. This is often unavailable, and such requirements limit the application scope. Self-supervised training, in addition to conventional supervised training, has been reported to yield promising results in [46]. In [46], an auxiliary 4-way head was trained to predict the rotation angle of an input which is found to improve overall robustness as well. Sun et al. [47] further improve on [46] and enable test time parameter update, i.e., online learning. Some of the recent works [48, 49] treat CNN's lack of robustness as a domain adaptation problem and propose to use rectified Batch Normalization Statistics (BNS) [48].

Activation Function. AFs are primarily designed to introduce non-linearity in a CNN. The de-facto AF ReLU [13] is widely used for its impressive performance on clean datasets. ReLU is unbounded by design and therefore, does not suffer from the vanishing gradient problem. However, ReLU's unbounded nature also allows propagation of noise resulting in misclassification. The role of AFs in noise suppression is an important but under-explored area of research. There are a number of bounded AFs such as C-ReLU and sigmoidal functions that have the potential to mitigate this issue but have not been thoroughly investigated in the literature. Rozsa et al. [21] proposed Tent AF as part of a defence mechanism against adversarial attacks. This AF bounds the input in a finite range but cannot maintain consistent classification accuracy across datasets. In Chapter 4, we show how our proposed AF with built-in low pass filtering can improve CNNs' robustness to noise.

2.5.2 Shift and Perturbation

Recently, CNNs' robustness has been put to test in various ways, ranging from adversarial attacks [50, 51, 52, 53, 54] to exposing CNNs to common corruptions [43, 31, 46, 55, 37, 56, 57, 58]. For humans, trivial image transformations such as shift does not affect our ability to classify an image so long as the key attributes of the image remain intact. However, CNNs are often found to flip prediction on single pixel shifts [59]. Any machine learning model capable of performing complex tasks is expected to effortlessly perform basic tasks. Therefore, CNNs' vulnerability to shift is concerning.

Commonly used DS methods and AFs are largely responsible for shift variance in CNNs. The role of DS is discussed at length in [60] and [59].

Downsampling (DS). Spatial dimension reduction has been an integral part of CNNs as it reduces the computational overhead and provides local translation invariance [61]. Lately, max-pooling and strided convolution have been predominantly used owing to their superior task performance [62]. Nevertheless, these operations do not use a blur-prefix, and in effect, violate the sampling theorem [60, 59]. Interestingly, average pooling – a well known DS method [63], is effectively a moving blur-filter, and resists aliasing, but cannot match max-pool's superiority in vision tasks [62]. Upon realising the potential, Zhang et al. [59] extended the notion of average pooling, and encouraged the use of stronger filters, e.g., a bilinear or Laplacian [64]. Here, the authors also provided ways to improve the task performance, and yet, used a single filter (MaxBlur-Pool (MBP)) for all layers, which is sub-optimal. In Spatially Adaptive Blur Pooling (SABP) [65], Zou et al. stressed the importance of an adaptive blur scheme but did not explore the relation with network depth; instead, they used a separate kernel for each local neighbourhood. While this method does alleviate performance, using too many filters per feature map is computationally exhaustive, and more importantly, leads to overfitting. Hypothetically, using a dataset without any location bias could avoid such overfitting; however, in practice, a bias-less dataset hardly exists, e.g., 90% of dog images in ImageNet have the main subject in the centre [60], due to a well-known phenomenon – photographer's bias [60]. Such bias is prevalent in almost all large-scale datasets, as such, learning separate and spatially local filters do not generalise well.

Kayhan et al. [66] argued that CNN filters exploit absolute spatial location due to image boundary effect – a form of convolution irregularity at image borders, and leads to shift variance. Here, an extension of the standard padding scheme is proposed (termed Full-Convolution or F-Conv) to lessen the boundary effects. Nevertheless, performance gain remains marginal.

As an alternative to blurring in the spatial domain, WaveCNet [67] does so in the frequency domain via Discrete Wavelet Transform (DWT). Here, the authors replaced common DS layers with the Low-Low (LL) DWT output. Although effective to some extent, a single DWT low-pass filter is used throughout the network, limiting the overall improvement. Moreover, the back-and-forth wavelet transforms add significant computational overhead. Similar to WaveCNet, Ryu et al. [68] also operated in the Fourier domain but only replaced the last average pooling layer with cropped Fourier coefficients (in ResNet). To address this, in an upgraded variant (DFT^{+}) [68], Fourier features from shallower layers were extracted in a separate sub-network and fused with the backbone features. Later, SVM was used for final prediction. Even so, DFT^{+} only marginally improves performance while incurring a lot of additional parameters.

In Chapter 5, we investigate layer-wise Fourier properties of the feature maps and propose a depth adaptive DS mechanism that outperforms contemporary solutions.

Activation Function. AFs inject non-linearity in otherwise linear CNNs, and enable a deep classifier to draw highly non-linear decision boundaries, in high dimensional input space. ReLU is the de-facto AF in modern CNNs [25, 26, 28]. Although studied for years [16, 13, 12], it came into light when Krizhevsky et al. [69, 14] successfully replaced Sigmoidal functions with it. Unlike its predecessors, ReLU offers early sparsity – a much-desired property that enables training very large and deep networks. Such sparsity gives a network enough 'room' to draw accurate decision boundaries among distinct classes. On the flip side though, it lets high frequency noise slip through – leaving the network vulnerable to aliasing, even with blur routines in place [60]. C-ReLU [69, 19] – a bounded variant of ReLU, could be used instead, as it caps the output to a constant threshold α. However, C-ReLU leaves a sudden and large saturated region beyond α, where a derivative is absent. Furthermore, finding an optimal α is

difficult as the buffer between the signal and noise is unknown. This highlights the importance of a 'soft' capping procedure – much like how signal rolls-off in a physical low-pass filter [70]. Different to rectified units, Sigmoidal functions, e.g., hyperbolic *tangent* ($tanh$ [71] in Equation 2.10), squashes the input and keeps the output within a finite range.

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.10}$$

In *tanh*, the roll-off effect can be regulated as well, but ensuring early feature sparsity remains a problem [69]. To strike the right balance between early sparsity and noise filtering, in Chapter 5, we propose a novel anti-aliasing AF.

2.5.3 Open Set Samples

Theoretically, for any unknown input, a CNN classifier should uniformly diffuse the prediction probability among the known classes. Such a uniform probability distribution should act as a sign that the input might be unknown. In practice, however, closed set CNNs trained to distinguish among a finite set of classes produce erratic results on unknown class samples. This is also known as the Open Set Recognition (OSR) problem. Here, we discuss the literature that addresses this issue.

OSR. OSR methods can be categorised into two main types: CNN-based and non-CNN-based. Scheirer et al. [72, 73] first formalised the OSR problem and proposed an SVM-based solution. Extreme Value Theory (EVT) is used in a number of works [74, 75] to reject Unknown Unknown[2] (UU) instances receiving probability score lower than a threshold. Dang et al. [76] proposed an OSR model where edge exemplars are selected for every class based on local geometrical and statistical properties [77]. Later, EVT-based rejection rule is adopted to reject any UU input that lies outside the KK class boundaries. Recently, deep learning-based networks are found to be more effective in OSR.

OSR with Counterfactual Images (OSRCI) [78] uses GAN to produce Counterfactual Images (CI) by morphing instances from a KK dataset (D_{KK}) to an extent where they

[2]Unknown Unknown (UU) refers to a class that is only available in the testset. Known Known (KK) refers to the known classes used in the trainset. Known Unknown (KU) refers to the unknown classes used as unknown trainer but do not appear during testing.

no longer are recognizable as a true class object. Later, these CI are used as the Known Unknown Trainer dataset D_{KUT} (i.e., CI $\approx D_{KUT}$). An additional $(N+1)^{\text{th}}$ 'other' class is introduced to accommodate images in D_{KUT} during training. At inference time, the classifier is expected to classify UU instances as the 'other' class. Although the network is expected to classify real-world objects, CI used as D_{KUT} lack visual characteristics of natural images, thus limiting the effectiveness of OSRCI.

Bendale et al. [79] replaced the SoftMax function with OpenMax. It is argued that forcing a network's total output probability to sum up to 1 leads the network to put undue probability score to UU instances at test time. It has been reported that test instances from D_{KK} put high prediction scores on the true class while the leftover probability is distributed to visually similar classes. However, the output probability distribution does not exhibit the same pattern for unknown instances. Inspired from this observation, the penultimate layer features (Activation Vector or AV) are extracted, and a mean vector (AVM) is calculated for each class (class-wise images are fed to the CNN). At test time, the AV of an image is extracted, and its distances to all the AVMs are calculated. Later, these distances are used with a threshold to detect whether the image belongs to a KK class or not. Ge et al. [80] supplemented OpenMax with Generative OpenMax (G-OpenMax) where they used GAN-generated data for OSR training. Classification-Reconstruction learning for OSR (CROSR) [81] is also an extension of OpenMax, but a different route is followed. A two-part deep network is used: a KK classifier and a UU detector. Multiple intermediate layers of the main CNN classifiers are treated as latent features, and a decoder is used to reconstruct the input. The UU detector and the classifier, both exploit the latent space features jointly to output detection decision and class label respectively.

Class Conditioned Auto-Encoder (C2AE) [82] also adopts an Encoder Decoder-based reconstructive approach. An encoder network is first trained on D_{KK}, and the penultimate layer is used as the encoded vector (EV). For each class in D_{KK}, one such EV is stored as the class condition vector. A decoder is later used to reconstruct the input from the EV. For each input to the encoder, the EV output is compared against all the stored EVs. The decoder reconstructs the input as perfectly as possible whenever a match is found between the output and stored EVs. However, for UU images (with no EV matches), the decoder is designed to perform a poor reconstruction so that the

reconstruction error is high at inference time. C2AE is not an end-to-end unit and leaves room for further improvement.

Geng et al. [83] proposed a visual and semantic prototypes-jointly guided CNN (VSG-CNN) to achieve the task of OSR. Instead of using conventional cross-entropy loss, a distance-based cross-entropy loss is used to find out the probability of a test instance belonging to each KK class. Once this probability set is at hand, the overall entropy is calculated and based on a threshold, the test instance is either rejected as a UU class or classified as one of the KK ones.

In Chapter 6, we propose a novel OSR training paradigm that produces high KK classification accuracy and UU detection.

Out-of-Distribution Detection. Out-of-Distribution (OOD) or Anomaly Detection is correlated to OSR. OSR deals with identifying real but UU images while classifying any KK instance. On the other hand, OOD detectors focus on anomalous outlier detection. Sometimes, these anomalies can be visually unrecognisable [30] or 'rubbish'. Some methods only detect the outliers first, and a separate classifier is used later for classification only if an input is deemed as KK by the detector [84, 85].

A common way to tackle the OOD problem is to simply augment an additional class to an existing CNN so that all OOD instances are classified as the 'other' class [86]. However, adding an additional class for all other images in the world does not perform consistently well on different benchmark datasets [30]. Hendrycks et al. [87] used the SoftMax probabilities as a heuristic to detect outlier images. A simple thresholding technique is applied based on the assumption that in distribution samples will always have higher probabilities, and OOD instances would not trigger high confidence predictions. However, this assumption is inaccurate. SoftMax thresholding does not work well as CNNs often misclassify an OOD image with high probability. As a solution, some detection methods introduce OOD samples in the training data and employ a custom loss function to uniformly diffuse probability on OOD training samples [88, 89, 90, 91]. These loss functions constrain the CNN from wrong overconfident predictions, but overall accuracy is compromised. Moreover, these methods use one benchmark dataset as D_{KK} and other entire publicly available benchmark datasets [88] (D_x) as D_{KUT} (i.e.,

$D_x \approx D_{KUT}$). We argue that mining only the hard Known Unknown (KU) negatives from D_x into D_{KUT} is a better option (i.e., $D_{KUT} \subset D_x$). This is the basis of our proposed OSR method in Chapter 6.

Li et al. [92] investigated the statistical properties of different CNN layer features to find a distinguishing pattern between In and Out of Distribution images. It was reported that the convolution outputs for D_{KK} and OOD instances have subtle difference. The difference is so subtle that even the most impactful dimensions in PCA (PCA head) fail to capture the difference. However, the tail (less informative eigen dimensions) of PCA shows a difference in the pattern. These features are used to train a cascade classifier for OOD detection.

GAN is used for OOD [89, 93, 90, 80, 78]. Lee et al. [90] used GAN to produce a D_{KUT} that neither belongs to D_{KK}, nor lies far away. A custom loss function (L) based on Kullback-Leibler divergence between a uniform distribution U and prediction on OOD instances is used for probability diffusion.

2.5.4 Object Scale

While an image classifier only outputs the label of the input, a detector, in addition to the label, outputs bounding boxes as well. However, the backbone of a detector is a CNN classifier network and hence, detectors share the same challenges present in CNN classifiers. For example, detectors are also vulnerable to distortions, transformations, and open set samples. Another common challenge for any CNN-based network is detecting objects with large variation in scale – especially small scale objects. This vulnerability can be attributed to successive DS layers in a CNN where signal from small objects can get progressively lost. Hence, detecting small objects has been considered a challenging task for both one and two stage detectors. While one stage detectors such as YOLO [94, 95, 96] and SSD [97] provide faster output, two stage detectors such as [98] have shown better accuracy. In this section, we first give details of the Region-based CNN (RCNN) family of two stage detectors and then present how the relevant literature addresses this issue.

2.5.4.1 RCNN and Fast RCNN

Girshick et al. [98] first proposed RCNN for object detection. Rather than brute force cropping of image segments and feeding them to a trained deep classifier, RCNN makes use of an external object proposal method (Selective Search [99]) to create 2,000 RoIs (Region of Interest) for each image. Since the input spatial dimension of a trained CNN classifier is fixed, the regions provided by Selective Search are warped into a fixed size and then fed to a CNN. RCNN is extremely slow in training and has high inference time as each of the region proposals is processed by a CNN separately. It means the feature extraction process is repeated 2,000 times for a single image. Instead of repetitive feature extraction from scratch, Girshick et al. later proposed Fast RCNN [100] that computes the convolution layers once per image and RoI pooling is done on the last layer feature maps, based on the regions proposed by Selective Search.

2.5.4.2 Faster RCNN

RCNN and Fast RCNN rely on a slow external object proposal method which becomes the detector bottleneck. To further speed up the detection process and improve the overall accuracy, Ren et al. presented Faster RCNN [101], where a Region Proposal Network (RPN) replaces Selective Search as the object proposal generator in Fast RCNN. An overall work-flow of the Faster RCNN detector is depicted in Figure 2.7. RPN shares the convolution layers with a conventional ImageNet pretrained backbone for computational efficiency. It takes the last layer feature maps from the backbone as input and outputs a set of object proposals. A predefined set of anchors are slid over the original image to check against the ground truth bounding boxes. This is to generate the object proposals and discard the backgrounds. Based on the proposals, the RoI pooling layer pools with a fixed size from the same feature maps and feeds the features to the fully connected layer. Finally, the class label and bounding box are predicted.

2.5.4.3 Small Object Detection

Although detecting medium and large objects is relatively easy, detecting smaller ones is extremely challenging for the detectors. Various techniques have been proposed to

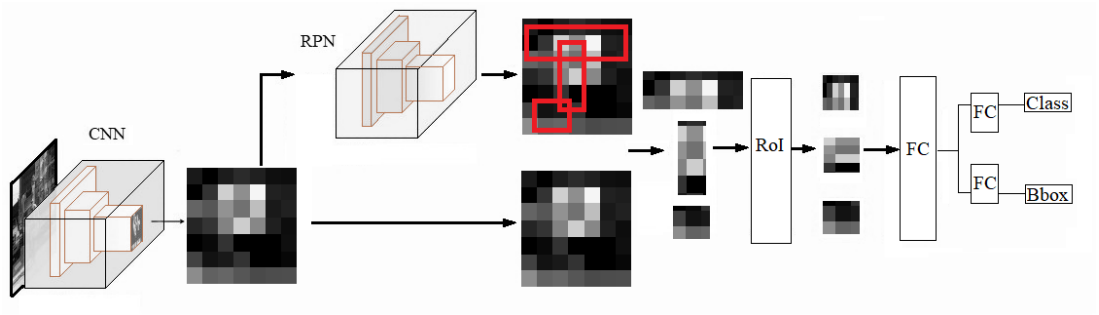

Figure 2.7: An overview of the Faster RCNN detector [101]. The final layer feature maps from the backbone (ImageNet pretrained CNN) are fed to the RPN and it generates a set of object proposals (red rectangles). The Region of Interest (RoI) pooling layer pools and warps the features into a predefined fixed size compatible with the backbone network. These features ultimately traverse through the FC layers and a class label and a bounding box are predicted as the detector output.

alleviate this particular problem. Among these, feature pyramid representation and contextual information embedding are two popular ways.

Feature Pyramid. Multi-scale feature pyramid representation is one way to improve performance [102, 103, 104]. For example, Lin et al. proposed to use a Feature Pyramid Network (FPN) [102] in Faster RCNN. This FPN exploits the inherent pyramid representation in a CNN as the DS layers reduce the the size of feature maps at different stages. For each training image, the lowest resolution feature maps are up-sampled and merged laterally with the previous feature maps. This backward up-sampling and merging process is repeated upto the first convolution layer to form a pyramid of features. Finally, FPN produces predictions on each level of the pyramid. One major drawback of FPN is that the shallower layer features are generic in nature and combining them with the deeper ones can degrade the overall detection performance.

In [105], the detector is trained with images of different resolutions for each training sample. For example, an input of size 224×224 is transformed into the following resolutions: 224×224, 448×448, 896×896 and 1200×1200. Such a pyramid representation of the input ensures variety in object scale during testing. The authors argued that while this method adds variety, it also introduces domain shift since large objects become too large in a high-resolution version of the original image and small objects become too

small in a low-resolution version. They proposed a module to ensure that RPN does not generate object proposals beyond a predefined limit to avoid the domain shift effect.

Contextual Information. Bell et al. [106] followed a different approach and conducted RoI pooling from several layers at a time. Since smaller objects are harder to detect, adding contextual information to these objects via multi-layer pooling may improve performance. The pooled features are concatenated and 1×1 convolution is used to reduce depth. Using shallow features is a concern here as well.

Wang et al. [107] proposed a new expansion layer that helps the RoI pooling layer to extract background context along with the small object. The surrounding area of the object is used as a cue for accurate detection.However, inconsistent and ambiguous background context can deteriorate the performance of these detectors.

Miscellaneous. Li et al. [108] subscribed to the idea of keeping large feature maps intact. However, this is done by avoiding pooling and dimension reduction layers which are pivotal for the detector's translation invariance capability. Fattal et al. [109] aimed to find the most informative and important feature maps for detecting small objects by frequency spectrum analysis (small objects have high frequency). Gao et al. [110] showed that in addition to the original three predefined anchors (9 in total) used in Faster RCNN, two additional smaller anchor boxes with sizes 32^2 and 64^2 improve the overall accuracy for tiny vehicles. Yang et al. [111] made use of scale-dependent pooling in order to accurately detect objects of all sizes. VGG16-based Faster RCNN is employed and the height of the RPN object proposals are used to choose the layer for RoI pooling. Shorter object proposals make use of shallower layers for pooling and larger ones use deeper layers for pooling. This can be attributed to the fact that small objects' information or activation is highly unlikely to reach the deeper layers due to spatial dimension reduction. A cascaded rejection classifier is used to eliminate negative proposals for better performance.

Cai et al. [112] combined several sub-networks in the backbone of a Faster RCNN detector. Since relatively shallower layers retain the information of small objects, multiple output layers from the subnetworks are used as the RPN input. Maintaining several additional subnetworks heavily adds to the memory complexity.

In Chapter 7, we propose a novel backbone network architecture for Faster RCNN that preserves features from both small and large objects.

2.6 Conclusion

In this chapter, we have thoroughly reviewed existing image classification methods found in the literature. A profound understanding of the history, evolution, and recent advancements is a prerequisite to contribute to the research domain which we presented in this chapter. We also highlighted a paradigm shift from hand-crafted to machine learning-based solutions in the field of computer vision – to be more specific in image classification. To this end, we outlined the weaknesses of modern CNNs under a number of specific challenges and discussed how the existing literature addressed these issues. We also pointed out notable deficiencies in these works that try to provide robustness to CNNs against such challenges. This provides the basis for our contributions in subsequent chapters.

Chapter 3

Distortion Robust CNN with DCT Augmentation

In Chapter 2, we discussed a number of challenging conditions for CNNs including distortion, shift, OSR, and small object detection. We also shed light on existing solutions. In this chapter, we focus on making CNNs robust to common forms of distortion. To this end, we first analyse the lack of robust and yet efficient augmentation methods and then propose a novel method called DCT augmentation. We also devise an adaptive regularisation method for further gains. Later, we conduct an in-depth investigation to reveal why our method works better. The main contribution of this work was accepted and presented at the 2019 IEEE International Conference on Image Processing (ICIP) [55].

The rest of this chapter is organised as follows. Section 3.1 provides the chapter introduction. In Section 3.2, we introduce and formulate the proposed DCT augmentation and adaptive dropout. In Section 3.3, we discuss the test datasets and provide a performance comparison of existing works with our proposed approach. Section 3.4 investigates the underlying reasons behind DCT-Net's success and why conventional CNNs lack robustness against distortion. Section 3.5 concludes this chapter.

3.1 Introduction

Most of the CNNs' test paradigms presume that the images are going to be artefact-free and high quality. However, in real-life, images can get distorted during acquisition, transmission or even by deliberation. For example, in low light conditions, the captured

images can exhibit noise. Motion or Gaussian blur can occur if the camera or subject is moving/shaking or due to focusing error. In transmission, packet-loss can potentially result in missing regions of the image, noise, or missing frequencies, depending on how the image is encoded. There are also situations where surveillance images are taken in challenging weather conditions (e.g., rain and snow) or the device used is of substandard quality resulting in degraded visual data. Additionally, with the advent of a wide range of cellular phones and hand-held devices, the requirement of high-quality images to perform different computer vision-related tasks may need to be relaxed. Distortion or substandard image quality can also degrade the performance of other CNN-based computer vision tasks as well, e.g. object detection, image retrieval, registration, and segmentation. Real world applications like autonomous driving and facial recognition-based security systems can also be affected by such distortions.

As outlined in Chapter 2, CNN architecture has evolved over the years starting from relatively shallow AlexNet [14] to deeper networks like VGGNet [113], ResNet [26], WRN [27], and DenseNet [28]. Despite increasing complexity and depth, CNNs are still susceptible to distortion. Often, these distortions are visually imperceptible [29, 43, 52, 45, 114, 115]. It is observed that a negligible amount of distortion can lead the network to misclassify an object as something else with surprisingly high confidence rate: some as high as 99.99% [52]. Figure 3.1 provides two examples of how CNNs fare against increasing level of distortion. Both of these distorted images are easily recognised by a human, whereas CNNs struggle. As can be seen, a WRN's correct class probability drops sharply and fluctuates inconsistently on both distorted samples. Often, the misclassified labels are not even visually close to the true class [43]. This indicates that misclassification due to distortion is not the same as common misclassification events. For example, often look-alike objects like cat and dog are misclassified as one another owing to their close proximity in the input space. However, something deeper is at play in case of distortion which we investigate in Section 3.4.

Data augmentation based on predefined distortions is a potential way to tackle distortion. However, knowing the types of distortions in advance is often unrealistic. Even if we allow the pre-emptive knowledge, training a network on all possible distortions separately makes it even more undesirable. All these facts culminate to an intriguing question:

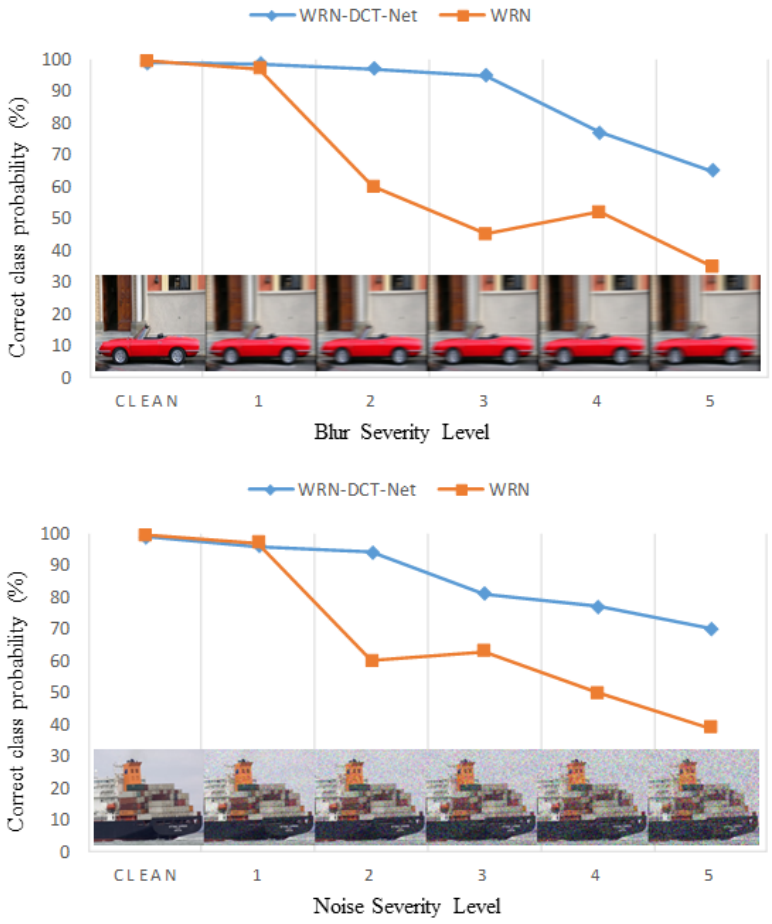

Figure 3.1: A Wide ResNet (WRN) trained on ImageNet struggles in classifying images with increasing Gaussian blur (Top) and noise (Bottom). Our proposed DCT augmentation and adaptive dropout in WRN-DCT-Net result in greater accuracy. Sample images are taken from ImageNet testset and scaled to fit inside the plots.

Is it possible to attain a network that is blind to any explicit distortion type and becomes robust against unseen distortions after being trained only once on the training data?

In this chapter, we propose a DCT-based data augmentation, which significantly increases the deep network's robustness against a variety of unseen distortions. For higher performance gain, we transform the input from spatial to frequency domain and drop low-impact DCT coefficients in random order. This, in turn, discards information from images and helps the deep network to learn from more diverse training data.

DCT-Net[1] no longer requires additional distortion information which is a much desired property.

In addition to DCT augmentation, we account for another well-known issue with CNNs, i.e., data overfitting. In this chapter, we further improve DCT-Net's performance by suppressing data overfitting. Rather than using conventional constant dropout [116] probability throughout the entire training period, we incorporate an adaptive scheme (adaptive dropout) for further performance gain.

Gaussian noise and blur are the two most common forms of image quality degradation. In addition to these two, we evaluate our network on salt and pepper noise, motion blur and speckle noise. Speckle noise is often inherent to sound/electromagnetic wave-based imaging systems but has similarity to Gaussian granular noise [117].

CNN's struggle in handling distortion and lack of proper explanation raise questions over our understanding of these networks. In this chapter, we provide insights and theoretical justification behind the proposed DCT-Net's success. We use manifold approximation, Convolutional Autoencoders (CAEs) and signal-to-noise ratio analysis to show why our proposed method works well.

To summarise, the main contributions of this chapter are as follows:

- We show that the current CNNs are indeed vulnerable to a number of common distortions.
- We propose a novel DCT-based data augmentation method to enhance CNNs' robustness against distortion.
- We also incorporate an adaptive dropout scheme to avoid overfitting to clean training data.
- We demonstrate our method's effectiveness through extensive experimental analysis across several datasets and distortion types.
- We conduct an in-depth investigation to find out the challenges posed by distorted images to conventional CNNs and provide insights to why DCT-Net works so well in addressing such challenges.

[1] DCT-Net refers to a deep network that incorporates DCT augmentation and adaptive dropout. Later in this chapter, we prefix the specific network backbone where required, e.g., VGG-DCT-Net denotes a DCT-Net based on VGG16.

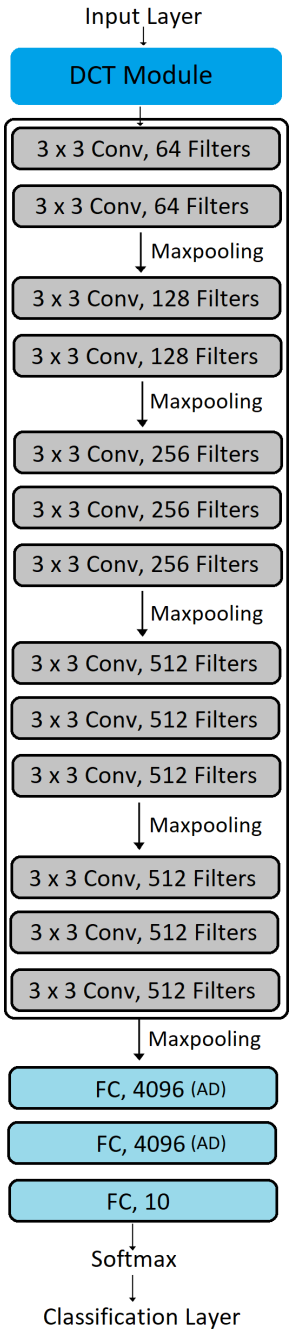

Figure 3.2: Overview of the proposed DCT-Net. The input layer precedes the proposed DCT module responsible for data augmentation. This module transforms input images from spatial to frequency domain and vice versa according to Algorithm 3.1. While in frequency domain, information is dropped based on DCT coefficients adding diversity in the training data. The Adaptive Dropout (AD) scheme is used in both of the Fully Connected (FC) layers. VGG16 network is used here for a simplified illustration (layers inside the rectangular box are identical to VGG16 layers).

3.2 Proposed Method

In this section, first, we introduce our proposed augmentation method and later elaborate on the adaptive dropout scheme.

3.2.1 Overview of DCT Augmentation

At training time, we simply incorporate a DCT module preceded by the input layer and followed by the first convolution layer (see Figure 3.2 for details). The pictorial depiction adopts a VGG16 [113] network because of its much simpler architecture compared to residual block-based networks like ResNet [26] or WRN [27]. As shown later in Section 3.3, WRN works even better with adaptive dropout. The ultimate goal of DCT augmentation is to eliminate the need for distortion specific training data as it might lead to cherry-picking distortions ahead of time. Rather, our classifier network is trained on a generic set of images augmented by the DCT module. The DCT module selects and eliminates a set of frequency coefficients from each of the training images. It is worth noting that DCT augmentation is network agnostic and can be used in the same way as shown in VGG16 or WRN networks.

3.2.2 Discrete Cosine Transform

DCT is a widely used technique to analyse the signal in the frequency domain. It has found its way into numerous applications, from lossy compression of audio (e.g. MP3) and image (e.g. JPEG) to spectral methods for the numerical solution of partial differential equations. To perform the Forward DCT (FDCT) in a standard JPEG compression [118], each image is divided into 8×8 blocks; effectively a 64-point discrete signal. However, it is found that this block-wise DCT operation may lead to undesired properties like blocking artefacts [119, 120]. Therefore, we consider only one block with dimensions equivalent to the height (H) and width (W) of the original input image. FDCT takes $H \times W$ signals as its input and outputs the corresponding basis-signal amplitudes or "DCT coefficients". The DCT coefficient values can thereby be regarded as the relative amount of the 2D spatial frequencies contained in the original input signal, which in our case is an image. One of the most critical features

of FDCT is that it concentrates most of the signal energy in a few transformed DCT coefficients in the lower spatial frequencies [121, 118]. In other words, the number of DCT coefficients with substantially high magnitude is very low, and the smaller coefficients are far greater in number. More often than not, the bulk of the information in a natural image is represented in lower frequencies. Higher frequencies generally encode sharp changes that add extremely fine details to the image.

There are a number of ways to perform DCT [122]. We make use of Fast Cosine Transform (FCT) [123, 124] because of its computational efficiency (*NLogN*). We make use of Equation 3.1 on an input Image A for FDCT and Equation 3.2 for Inverse DCT (IDCT) to obtain the reconstructed Image.

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad 0 \leq p \leq M-1, 0 \leq q \leq N-1 \tag{3.1}$$

where,

$$\alpha_p = \begin{cases} \sqrt{\frac{1}{M}}, & p = 0. \\ \sqrt{\frac{2}{M}}, & 1 \leq p \leq M-1 \end{cases}$$

$$\alpha_q = \begin{cases} \sqrt{\frac{1}{N}}, & q = 0. \\ \sqrt{\frac{2}{N}}, & 1 \leq q \leq N-1 \end{cases}$$

And the IDCT is performed by:

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad 0 \leq m \leq M-1, 0 \leq n \leq N-1 \tag{3.2}$$

where,

$$\alpha_p = \begin{cases} \sqrt{\frac{1}{M}}, & p = 0. \\ \sqrt{\frac{2}{M}}, & 1 \leq p \leq M-1 \end{cases}$$

$$\alpha_q = \begin{cases} \sqrt{\frac{1}{N}}, & q = 0. \\ \sqrt{\frac{2}{N}}, & 1 \leq q \leq N-1 \end{cases}$$

Here M and N are the row and column sizes respectively of the input and output images.

Algorithm 3.1 : Input to DCT Module: Image *RGB* $(H \times W \times C)$, and Output: DCT augmented image *D* $(H \times W \times C)$

I = rgb2Ycbcr(RGB)
for all channels c = 1 to C **do**
  $DCT\_Coeffs[c]$ = DCT(I), using Eq. 3.1
  $Abs\_DCT\_Coeffs[c]$ = ABS ($DCT\_Coeffs[c]$)
  Y =Uniform_Random_Threshold($0, b$)
  for all $DCT\_Coeffs[c] < Y$ **do**
    $DCT\_Coeffs[c] = 0$
  end for
  $O[c]$ = IDCT ($DCT\_Coeffs[c]$) using Eq. 3.2
end for
D = Ycbcr2rgb(O)

3.2.3 DCT Module Integration

DCT module (see Figure 3.2) transforms each of the training images using FDCT and produces a set of DCT coefficients. A cut-off threshold Y is chosen for each training image from a predefined range of $[0, b]$ based on heuristics (grid size and image dimensions) [118]. Y is effectively a random DCT coefficient threshold. All the coefficients lying below Y are turned zero. From a high-level point of understanding, when Y is equal to or close to 0, the input image undergoes no or minimal transformation, which means there is hardly any loss of input information. On the other hand, if Y is a large number close to b, this thresholding step removes most of the high frequencies from an image. Since a large part of the signal strength is stored in the lower spectrum, the loss of information takes away mostly sharp changes and finer details of the image pertaining to different edges and contours. Along with the omission of most of the high frequencies, some of the lower frequencies with little visual impact on the input image get discarded as well in the process. It is worth noting that the thresholding considers the absolute values of the coefficients. Inverse DCT or IDCT is performed on the remaining DCT coefficients to reconstruct the transformed image. This DCT transformed image is then fed forward to the first convolutional layer. The modus-operandi of the proposed DCT module is presented in Algorithm 3.1. Based on heuristics [118], we set the upper bound for Y, i.e., b as 40 for CIFAR10/100 and 60 for ImageNet.

The DCT module is free of trainable parameters, and no backpropagation based learning takes place within this module. Once the deep network is trained, this module

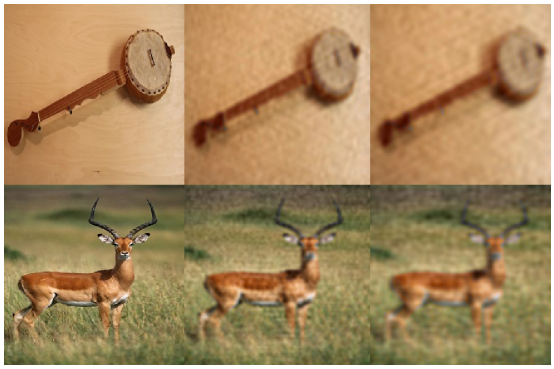

Figure 3.3: Sample output of DCT module. Each input is transformed into the frequency domain via forward DCT. Later, (from left to right) information is discarded in increasing magnitude with DCT coefficient threshold set to 0 (Left), 25 (Middle) and 50 (Right). Lower and upper left ones are the original image. Inverse DCT reconstructs the image from remaining coefficients. It should be noted that the images do not lose overall object structure. The sample pair belongs to ImageNet with class label Banjo (Top) and Impala (Bottom).

is removed from the network and test images directly enter into the first convolutional layer. Figure 3.3 depicts a sample image pair of DCT transformed training data from ImageNet. It can be visually observed that higher the threshold value, the greater the loss of information.

3.2.4 Implementation Details

VGG16 and WRN are widely used as the backbone network in the relevant literature [43, 38, 36]. Therefore, we use these two particular CNNs as the backbone of our DCT-Nets. VGG16 uses 3×3 filters in all 13 convolutional layers, ReLU is used as activation function, and 2×2 max pooling with stride 2 is used. As for the other network, we use WRN-40-2, where 40 is the number of convolution layers, and 2 is the widening factor (considering ResNet has a widening factor of 1). Contrary to the single FC layer used in conventional WRNs [26], we use two such layers as this configuration maintains the clean classification accuracy and gives room to employ adaptive dropout. The original 1,000-way softmax and classification output layers are replaced with 10 and 100 for CIFAR-10 and CIFAR-100 respectively for both networks. We use SGD with momentum value 0.9 and data shuffling before every epoch. We train the network for 200 epochs with minibatch size set to 128. The initial learning rate is set to 0.01 with

a scheduled learn rate drop factor of 0.1 at 60, 120, and 160 epochs. We make use of the proposed adaptive dropout (described in the next section) and L2 regularisation to counter overfitting [125, 126].

3.2.5 Adaptive Dropout

Dropout is a well-known technique to counter the effect of data overfitting [116]. What dropout essentially does is it deliberately ignores a random number of neurons in the FC layers based on some constant probability P. The idea is not to let the network become too reliant on a specific set of features and thereby avoiding overfitting. Even so, the application of a constant P limits the deep network's ability to adapt since each training dataset is different and comes with its challenges. We extend the idea of dropout and show that an adaptive scheme serves better. The dropout probability P is initialised with 0.1 at the beginning of training and updated from a range of [0.1,0.5] with the smallest increment unit of 0.1. The update only comes into effect when the network seems to converge to training data, and the possibility of overfitting is inevitable. P is updated when the minibatch training accuracy reaches and stays above 80% for an entire epoch. The value of P depends on the remaining number of epochs which is divided into five equal intervals. As the training proceeds forward, P increases in each of the five epoch intervals by 0.1.

3.3 Performance Evaluation

In this section, we present a comparative performance analysis on several benchmark datasets. In addition to the original test dataset, we evaluate the networks on five different types of distortions, namely Gaussian noise, Gaussian blur, salt and pepper noise, motion blur, and speckle noise.

3.3.1 Datasets

We consider CIFAR-10, CIFAR-100 [127] and ImageNet [1] as our benchmark datasets. CIFAR-10 consists of 60,000 $32 \times 32 \times 3$ images in 10 classes, with 6,000 images per class.

Table 3.1: Performance comparison of different networks on clean and distorted test datasets of CIFAR-10/100 and ImageNet. Accuracy over each type of distortions is the average over all distortion levels specified in Section 3.3.2. Overall accuracy is the network's average accuracy across clean and distorted datasets (best accuracy is highlighted in bold).

CIFAR-10									
CNN Model	VGG16	WRN	Original	Gaussian Noise	Salt & Pepper Noise	Speckle Noise	Gaussian Blur	Motion Blur	Overall Accuracy
M_{clean} (VGG16 [113])	✓		88.43	40.53	42.60	45.10	61.43	56.35	55.74
M_{Gblur} [32]	✓		68.84	29.33	30.25	38.36	73.36	50.51	48.44
M_{BN} [33]	✓		83.36	59.06	51.87	55.08	69.36	52.98	61.95
MixQualNet [43]	✓		81.56	60.69	58.99	57.36	70.26	62.89	65.29
DCT-Net (ours)	✓		86.97	57.06	61.23	62.05	77.15	70.91	69.23
M_{clean} (WRN [27])		✓	**96.00**	38.42	39.87	42.11	58.50	52.77	54.61
AutoAugment [36]		✓	84.22	60.65	55.80	57.04	69.95	58.11	64.30
AugMix [38]		✓	94.10	62.10	61.15	64.45	76.54	70.36	71.45
DCT-Net (ours)		✓	95.78	**65.08**	**62.14**	**66.32**	**78.05**	**71.55**	**73.15**
CIFAR-100									
CNN Model	VGG16	WRN	Original	Gaussian Noise	Salt & Pepper Noise	Speckle Noise	Gaussian Blur	Motion Blur	Overall Accuracy
M_{clean} (VGG16 [113])	✓		67.49	25.52	29.62	32.91	50.39	48.30	42.37
M_{Gblur} [32]	✓		56.81	19.69	22.26	27.35	62.37	42.77	38.54
M_{BN} [33]	✓		62.44	46.08	38.28	33.03	52.46	35.88	44.70
MixQualNet [43]	✓		63.50	45.89	58.01	57.61	55.22	49.83	55.01
DCT-Net (ours)	✓		65.39	44.85	62.33	60.65	64.05	55.91	58.86
M_{clean} (WRN [27])		✓	**79.08**	22.11	26.45	28.78	46.17	45.22	41.30
AutoAugment [36]		✓	72.10	45.22	55.25	54.06	51.36	45.12	53.85
AugMix [38]		✓	77.67	49.20	62.80	63.65	61.70	54.33	61.56
DCT-Net (ours)		✓	78.51	**52.27**	**63.07**	**65.98**	**68.01**	**58.82**	**64.44**
ImageNet									
CNN Model	VGG16	WRN	Original	Gaussian Noise	Salt & Pepper Noise	Speckle Noise	Gaussian Blur	Motion Blur	Overall Accuracy
M_{clean} (VGG16 [113])	✓		92.60	32.45	15.32	24.98	29.53	25.67	36.76
M_{Gblur} [32]	✓		55.36	16.24	18.91	22.63	39.88	30.69	30.62
M_{BN} [33]	✓		75.77	54.25	42.30	38.65	30.14	27.85	44.83
MixQualNet [43]	✓		82.22	51.88	49.73	48.45	42.10	37.35	51.96
DCT-Net (ours)	✓		87.50	52.98	51.30	50.98	43.90	41.32	54.66
M_{clean} (WRN [27])		✓	**95.89**	30.22	15.09	22.54	26.45	22.98	35.53
AutoAugment [36]		✓	94.03	50.98	47.15	45.21	38.15	27.35	50.48
AugMix [38]		✓	94.46	52.10	51.90	52.30	45.33	41.01	56.18
DCT-Net (ours)		✓	95.26	**54.87**	**53.02**	**56.05**	**48.25**	**44.16**	**58.60**

The split is 50,000 training images and 10,000 test images. CIFAR-100 has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. Dimensions of these images are the same as CIFAR-10. In ImageNet, there are 1,000 object classes and approximately 1.2 million training images, 50,000 validation images and 100,000 test images.

3.3.2 Test Data

Five types of distortions are introduced in the test dataset. A set of sample test image pairs with increasing level of distortion can be seen in Figure 3.4. To prepare the

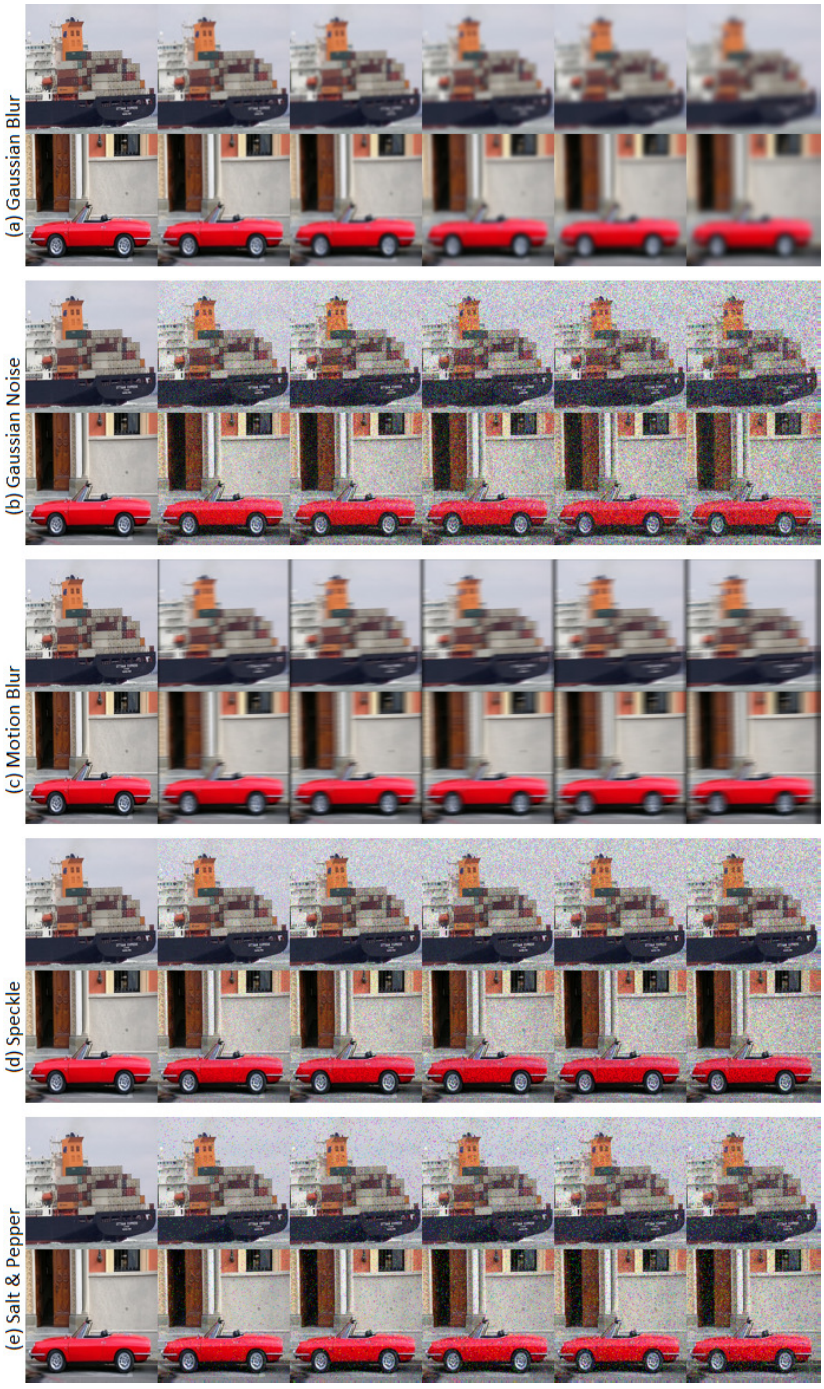

Figure 3.4: Progressively distorted image pairs from ImageNet test dataset. From top to bottom: (a) Gaussian Blur, (b) Gaussian Noise, (c) Motion Blur, (d) Speckle, and (e) Salt & Pepper. Networks are tested against these five types of distortions.

distorted test data, we follow the testing protocol in [43] for available distortions.

- Gaussian noise is added to an image with the standard deviation (σ) ranging from 0 to 100.
- For the Gaussian blur kernel, σ is set between 0 to 5 for CIFAR-10/100 and 0 to 10 for ImageNet. σ is greater for ImageNet because of the higher resolution of images in this dataset compared to CIFAR-10/100. A similarly high σ for CIFAR-10/100 could render the objects visually unrecognisable. The above-mentioned range of σ is enough to attain visually similar blur in lower resolution images.
- Salt and pepper noise are replicated by turning on and off pixels with a predefined probability. We add salt and pepper noise to an image pixel with a probability varying from 0 to 0.5.
- For motion blur kernel in CIFAR-10/100, we use motion angle ranging from 0 to 22.5 degree because of the small spatial dimension of these two datasets. The number of pixels is set to 10 as linear motion parameter. As the spatial dimension is greater in ImageNet, we use a motion angle range of 0 to 45 degree with the number of pixels set to 15 as linear motion parameter.
- We add multiplicative speckle noise to Image I and produce noisy Image J, where $J = I + n * I$, n is a uniformly distributed random noise. Variance ranging from 0.1 to 0.5 is used for speckle noise.
- All the test images are tested at five increasing distortion levels. The levels are chosen uniformly, i.e., the defined range for each distortion is divided into five equally spaced ranges.

3.3.3 Performance Comparison

We denote M_{clean} as the network trained on the clean dataset. M_{BN} [33] is the model which is fine-tuned on both Gaussian blur and noise. MixQualNet is the ensemble of individual distortion expert models with a gating network [43].

Table 3.1 compares the classification accuracy of DCT-Net with others. All of the deep networks are tested against five increasing severity levels of the corresponding

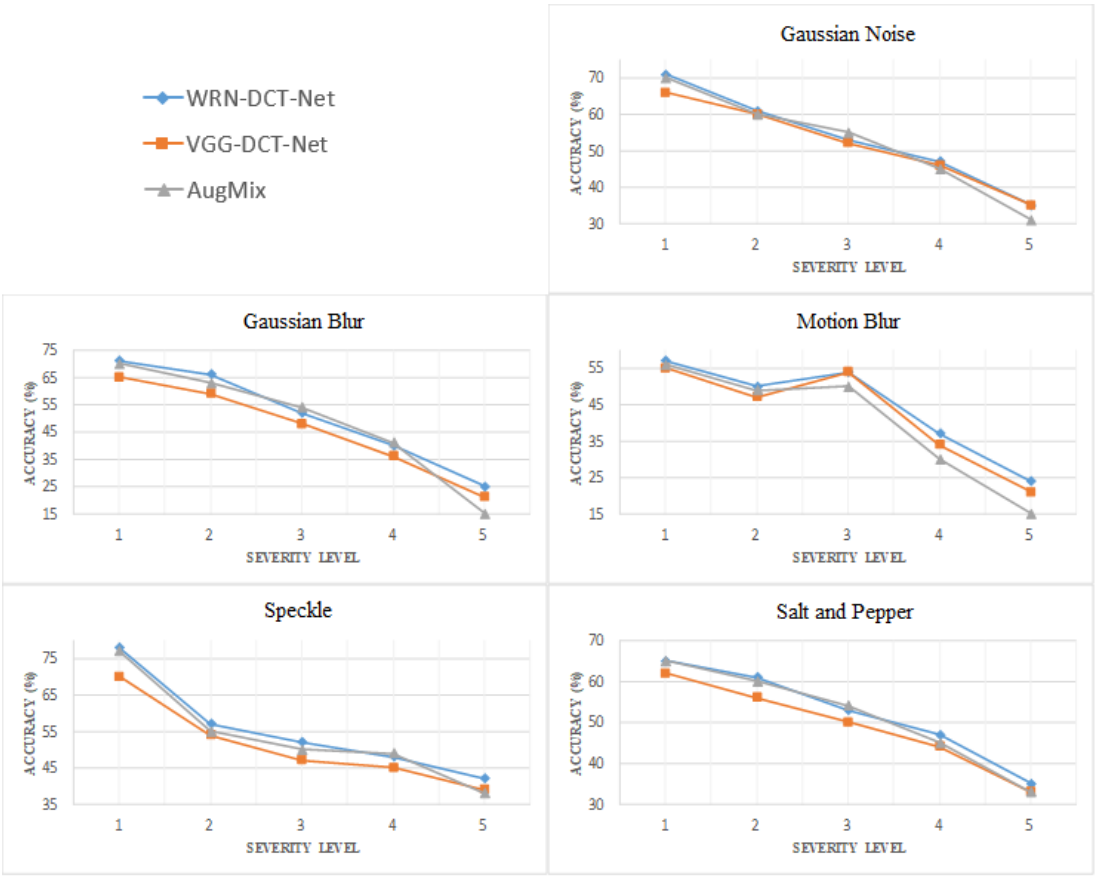

Figure 3.5: Comparative performance analysis over five increasing distortion levels (ImageNet) specified in Section 3.3.2. WRN-DCT-Net exhibits better consistency across datasets compared to VGG-DCT-Net and AugMix [38].

types of noise and blur. The levels are uniformly chosen between the minimum and maximum distortion range specified in Section 3.3.2. Accuracy averaged over all severity levels is reported in Table 3.1. For a particular network, the original accuracy in Table 3.1 is computed on the corresponding clean test dataset, whereas the overall accuracy in the last column is the numerical average over all six individual accuracies. We first discuss methods based on the VGG16 backbone. Later, we discuss methods based on the more advanced WRN backbone. It is worth noting that we incorporated adaptive dropout in all the networks in Table 3.1 for fair comparative performance analysis. Understandably, WRN-based methods exhibit better robustness against distortions due to their much more complex and deeper network architecture.

VGG16-based Networks. It is evident from Table 3.1 that a network trained on one specific type of distortion (M_{Gblur} [32]) performs well on that distorted test dataset. However, it struggles to generalise well to other types of distortion and also the accuracy on the clean testset drops. On the other hand, M_{BN} [33] has a mediocre performance on all the test datasets. This is because M_{BN} has exposure to both types (noise and blur) of distortions.

WRN-based Networks. Our proposed DCT-Net based on WRN-40-2 (WRN-DCT-Net) is found to outperform other networks. WRN-DCT-Net achieves the best overall accuracy in all three datasets (73.15% in CIFAR-10, 64.44% in CIFAR-100 and 58.60% in ImageNet). It also demonstrates consistent accuracy on each type of distortion while maintaining competitive performance on the clean test dataset. AugMix also exhibits comparable accuracy across distortions. However, in AugMix, a number of predefined parametric transformations are performed in series. In contrast, DCT-Net simply trains on one simple augmentation methodology.

Figure 3.5 compares the performance of WRN-DCT-Net, VGG-DCT-Net, and AugMix [38] on ImageNet over increasing level of distortions. It is evident that with increasing distortion severity, all three networks struggle to maintain classification accuracy. WRN-DCT-Net shows better consistency in each of the plots.

3.4 Explanation

From a data classification perspective, CNNs are nothing but a highly non-linear function $\mathcal{F}(x) \rightarrow \mathbb{C}_i$, where x is the input and $\mathbb{C}_i$ denotes the output class label. Image classification involves extremely high dimensional input space and distinguishing different samples with a complex decision boundary in this vast space is indeed a challenging task. Since CNNs are proven to be universal function approximators [128], they are capable of producing impressive results, especially on the clean dataset. However, as stated earlier in this chapter, images even with slight distortion lead the network to predict one class object as something else with a surprisingly high probability. Our proposed DCT augmentation improves the network's accuracy significantly on distorted dataset while maintaining impressive results on clean data. In this section,

we argue that deep CNNs' lack of robustness against distorted images has a lot to do with the intriguing nature of high dimensional input space and the decision boundaries learned by a CNN. We investigate the notion of data manifold and signal to noise ratio analysis to gain more insight. We explain how DCT augmentation relates to all these aforementioned concepts and improves overall classification performance.

3.4.1 High Dimensional Input Space and Data Manifold

CNN classifiers take raw image pixels as input. Each of these pixels is effectively an independent variable and an axis in the input space. Therefore, the space that accommodates all the training images is directly proportional to the image resolution, and the input space increases rapidly with the image dimensions. ImageNet images, for example, create an input space with a staggering $227 \times 227 \times 3 = 154,587$ axes. Nonetheless, natural images cluster together in a relatively small sub-space. This sub-space is also known as the data manifold [50, 129, 130].

To further clarify the notion of manifold, imagine a hypothetical set-up where we have an image dataset D_I (with 1,000 items) and each image in D_I has a spatial dimension of 4×4, i.e., 16 dimensions in total ($[1, 2, 3, ..., 16]$). Further assume only the first two dimensions (1 and 2) have large variance across D_I compared to the remaining dimensions ($[3, ..., 16]$). This means images in D_I cluster together in a smaller subspace leaving large unknown space around it. This smaller subspace is known as the manifold. Likewise, clean images of a large-scale dataset also reside in a smaller subspace or manifold (say M_{clean}) surrounded by unknown space [131, 50]. The decision boundary learned by a CNN during training is quite accurate inside M_{clean} since the clean training data belongs to M_{clean}. At inference time, as long as a test sample belongs to M_{clean}, a CNN is likely to classify it correctly. Interestingly, CNN's decision boundary is not confined within M_{clean} only. Rather, the learned decision boundary is open-ended and extend infinitely [131, 50] into the unknown space. This indicates an unreliable class prediction on a sample drawn from this unknown space (see Figure 3.6 for a conceptual illustration of this phenomenon). Note that a distorted image might still be classified correctly if it happens to be under the correct (extended) decision boundary. CNN's decision boundary extrapolation beyond the scope of M_{clean}

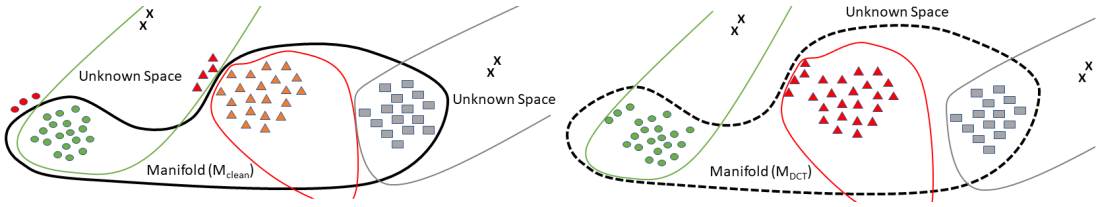

Figure 3.6: Triangle, rectangle, and circle represent sample data points (images) of three different classes in a hypothetical set-up. Cross represents rubbish images (images without any meaningful object). (Left) Triangles just beyond the clean manifold (M_{clean}) are the distorted duplicates. A CNN trained on M_{clean} does not account for the unknown space. Although the decision boundary is quite robust inside M_{clean}, it is open-ended and extends beyond M_{clean}. Such open decision boundaries may host unknown data points (e.g., triangles falling inside the circle class decision boundary) resulting in misclassification. (Right) DCT augmentation supposedly expands the manifold ($M_{clean} \rightarrow M_{DCT}$). M_{DCT} accommodates the distorted duplicates previously lying outside M_{clean} and a CNN trained on M_{DCT} can classify distorted images with greater accuracy.

can be attributed to the piece-wise linear activation function, i.e., ReLU. ReLU's linear nature means that the open-ended learned decision boundary gets the chance to extend into the unknown space. As distorted images are scattered in the unknown space, this also answers why a visually unrecognisable image or rubbish image (far away from M_{clean}) often gets misclassified with high probability.

Considering rubbish images lie outside of M_{clean}, one might ask where the distorted images belong in this high dimensional space. As discussed by Goodfellow et al. [50], and others [130, 132], even a tiny perturbation (e.g., noise or blur) can push an image out of M_{clean} and the direction of this push is more important than the absolute distance from M_{clean}. This means an image with visually imperceptible distortion can also get classified into a wrong class despite residing close to M_{clean}. In other words, a distorted duplicate can be visually similar to the original image and yet misclassified if it is outside of M_{clean}.

Each clean image (x) has multiple distorted duplicates (x') outside M_{clean}. For example, given a clean image of a dog, it is easily possible to reproduce a number of duplicates by altering the pixels (e.g. Fast Gradient Sign Method (FGSM) [50]). These duplicates (x') would still look like a dog with minimal change in visual appearance. However, such x' reside in the unknown space and hence may be misclassified by a CNN. Our proposed DCT module transforms each training image according to

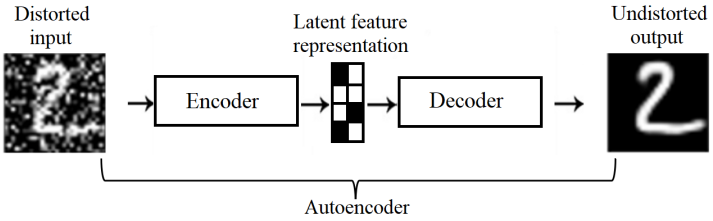

Figure 3.7: Overview of a CAE capable of reconstructing a cleaner version from a distorted image.

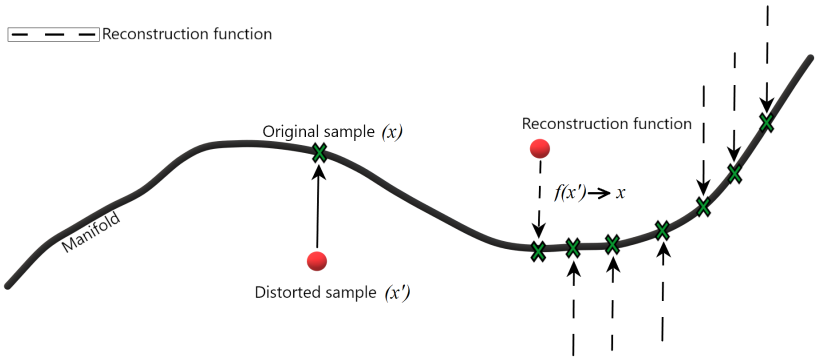

Figure 3.8: DCT augmentation represents a variety of common distortions. As a result, an Autoencoder trained with DCT transformed data as input and clean data as output can project unseen distorted duplicates back to the clean manifold.

the process described in Section 3.2. The output of the DCT-module expands M_{clean} (say M_{DCT} is the expanded manifold). Consequently, a wide range of distortions previously residing outside M_{clean} gets admission into M_{DCT}. As a result, the re-calibrated decision boundary of the DCT-Net (inside M_{DCT}) encompasses both the clean and distorted images and can classify them with greater accuracy.

All the distortions considered in this chapter, i.e., Gaussian noise, blur, salt and pepper, and speckle noise are simply distorted duplicates. They do not belong to M_{clean}. As a result, conventional CNNs struggle in classifying these images. The proposed DCT-Net overcomes this challenge by expanding the clean manifold hosting clean and most of the distorted images.

3.4.2 Autoencoders and Manifold Approximation

The proposed DCT-Net's ultimate goal is to correctly classify images having a variety of distortions without knowing the specific attributes of a distortion type in advance. This is achieved by a generic dataset augmented through the DCT module during training. Section 3.3 has already shown the efficacy of our method. An in-depth analysis of the data manifold in high dimensional space is used in the last section to explain the proposed network's superior performance. We now use CAE to provide another perspective of DCT-Net.

CAE [133, 134] consists of an encoder and a decoder (see Figure 3.7). The encoder has a set of conventional convolution steps that compresses the input into a latent feature space. The decoder generally hosts a set of deconvolution or upsampling steps to reconstruct the output from the latent representation. In CAE, when the output is forced to be a clean version of a distorted input, the intermediate latent layer learns meaningful features in a lower dimension like PCA [135] or t-SNE [136]. .

Let X be the set of all clean images $x_i \in X$, $i = \{1, 2, 3, ...n\}$ and let X'_d be the corresponding distorted set where $\forall x \exists x' (x'_i \in X'_d)$, $i = \{1, 2, 3, ...n\}$ and d is the distortion type.

An Autoencoder trained with DCT-transformed images becomes a function that can project a distorted image (x') back to the manifold (x), i.e., $f_{DCT}(x'_i) \rightarrow x_i$, $i = \{1, 2, 3, ...n\}$ (see Figure 3.8).

Our aim here is to demonstrate why the DCT transformed images are capable of representing all five types of distortions. To show this, we do the following.

- A CAE is trained with a set of DCT transformed images (X^{DCT}) as input and the corresponding clean versions (X) as output ground truth. Once trained, if the CAE becomes capable of producing clean images from the distorted input X'_d (i.e., $f_{DCT}(X'_d) \rightarrow X$), it can be implied that the DCT-transformed images in X^{DCT} indeed represented all those distortions.
- To test our hypothesis, the distorted dataset (X'_d) is fed to the CAE, and the reconstructed images (X^{CAE}) are analysed.

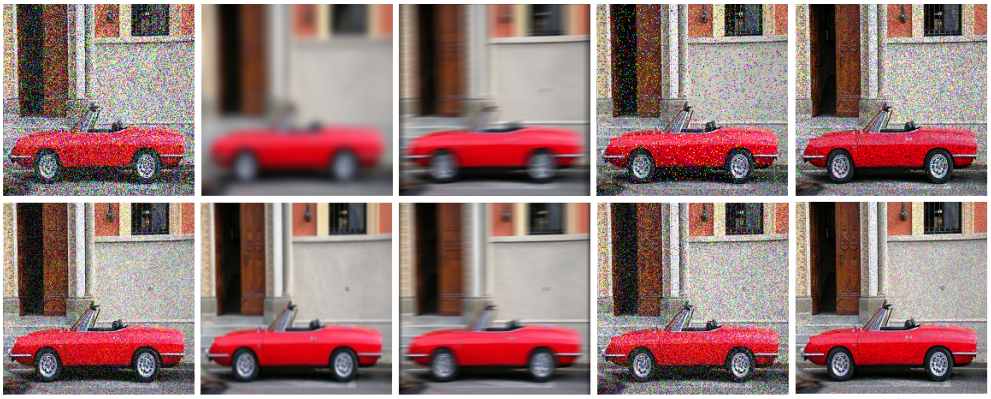

Figure 3.9: The top row represents distorted images, and corresponding bottom row image is the Convolutional Autoencoder (CAE) reconstruction. (a) Gaussian noise, (b) Gaussian blur, (c) Motion blur, (d) Salt and Pepper noise, and (e) Speckle noise. CAE is trained to reconstruct the original image from the corresponding DCT transformed image. Once trained, CAE reconstructs cleaner images from distorted inputs. The reconstructions are not completely distortion-free, but substantially cleaner. Intriguingly, all the distorted images (top row) are misclassified by conventionally trained VGG16 and WRN, but the reconstructed ones (bottom row) are classified correctly.

- X^{CAE} appears to be much cleaner compared to its distorted counterpart (X'_d) as depicted in Figure 3.9, i.e., $f_{DCT}(X'_d) \rightarrow X \approx f_{DCT}(X'_d) \rightarrow X^{\text{CAE}}$.

Interestingly, the distorted automobile images taken from (X'_d) in the first row of Figure 3.9 are misclassified by a conventional CNN, but the corresponding reconstructions (X^{CAE}) are correctly classified by the same CNN. This implies that (X^{DCT}) was able to represent all the distortion types present in (X'_d). This is the reason DCT-Net performs well on distorted images without knowing in advance the explicit attributes and type of distortions.

Peak Signal-to-Noise Ratio. Peak Signal-to-Noise Ratio or PSNR has long been used as a metric to determine the quality of an image. For a given modified or transformed image, PSNR is computed against a reference or original image. A higher value means greater similarity for the image pair. To further explain our results aided by CAEs, we resort to a PSNR-based comparison. PSNR is calculated for the following two pairs:

(a) The distorted image X'_d and the original image X.

(b) The CAE (f_{DCT}) reconstructed image X^{CAE} from X'_d and the original image X.

Table 3.2: PSNR comparison for the Original-CAE reconstructed pair and the Original-Distorted pair across all three datasets. 1,000 images were randomly sampled for this experiment.

Dataset	**PSNR**	
	Original-CAE Reconstructed	Original-Distorted
CIFAR10	21.65	15.08
CIFAR100	22.87	16.42
ImageNet	20.28	16.15

Ideally, *(b)* should have higher PSNR than *(a)* if the reconstructed image is indeed more similar to the clean image. This, in fact, is what we have observed from experimental results. For this test, we randomly sample 1,000 images from each of the CIFAR-10, CIFAR-100, and ImageNet datasets. We use these as the clean set X. A CAE is trained following the steps in Section 3.4.2 with X^{DCT} as input and X as the ground truth. Once trained, PSNR is calculated for both (a) and (b) as mentioned above.

It is visible from Table 3.2 that the PSNR for the original-reconstructed (X-X_{CAE}) pair is consistently higher than the original-distorted (X-X'_d) pair. This justifies our Autoencoder analogy and the effectiveness of X^{DCT} in mimicking X'_d.

3.5 Conclusion

We proposed DCT-Net with an adaptive dropout and showed that discarding part of the input signal based on DCT adds diversity to each of the training data. Since the threshold used for discarding information is random for each image, every epoch is likely to produce a different version of a training image. This way, the network gets to learn a wide range of features from all the variants of an image. Our proposed DCT-Net is capable of correctly classifying images even with substantial distortion or degradation. DCT-Net is 'blindly' trained only once and shows impressive accuracy on unseen distortions on several benchmark datasets. To unveil the reasons behind DCT-Net's robustness against distortion, we provided a theoretical justification with the help of data manifold, CAE, and PSNR.

While data augmentation and network regularisation help in distortion robustness, improving the fundamental functional blocks of CNNs can provide further gains. The

next chapter, i.e., Chapter 4 presents a novel AF that complements DCT augmentation to make CNNs even more robust to distortions.

Chapter 4

Improved Distortion Robustness with a Novel Activation Function

In the previous chapter, we showed how the proposed DCT augmentation and adaptive drop-out improve CNN's performance on distorted datasets. In this chapter, we first discuss what role AFs can play to keep the impact of distortions in check. Later, we shed light on the deficiencies of existing AFs and argue that a built-in filtering mechanism can largely solve the problem. To this end, we propose an AF with built-in signal filtering called Low-Pass ReLU (LP-ReLU). We also show that LP-ReLU coupled with DCT augmentation proposed in the last chapter can provide substantial performance gain. The core contribution of this work has already been published in the IEEE ACCESS [137].

The rest of this chapter is organised as follows. Section 5.1 provides an introduction of this chapter. The proposed AFs and the recommended augmentation method are discussed in Section 4.2. Section 4.3 provides details on the benchmark datasets, extensively evaluates the performance of different networks, and outlines the implementation details. Section 4.4 presents the proposed decision space visualisation process to deepen our understanding of CNN's lack of robustness on distorted input. Finally, Section 4.5 concludes the chapter.

4.1 Introduction

CNNs achieve high classification and recognition accuracy on clean benchmark datasets. However, their performance deteriorates on corrupted datasets even when the cor-

ruption[1] is visually imperceptible. This is particularly concerning for real-world safety-critical applications where the data may not always remain clean. For example, a self-driving car should not presume to have ideal driving conditions at all times. Optical sensors are expected to encounter a wide range of corruptions such as signal noise, motion, and defocus blur. Variable weather conditions including sudden change of brightness, snow, frost, and fog, can also pose challenges. Digital artefacts due to transmission and compression can compound the challenges even further. Therefore, accounting for corruptions at inference time and closing the performance gap between clean and corrupted datasets are important.

Different data augmentation techniques [36, 55, 39, 38, 138] are widely used to address this issue. In addition to data augmentation, adversarial training [31, 139, 34, 140, 141], self-supervised learning [46, 47], domain adaptation [48, 49], and loss function optimisation [142, 46] are some of the other ways explored to enhance CNNs' robustness against such corruptions.

In this chapter, we take a different approach and investigate corruptions from the frequency domain. We emphasise on the role of AFs and argue that if designed properly, AFs can substantially enhance CNNs' robustness against corruptions. Currently, ReLU [14] is the most widely used AF because of its computational efficiency and convergence ability compared to other AFs [20]. ReLU does not suffer from the vanishing gradient problem [11] and it allows very deep networks to be optimised through backpropagation. However, ReLU and a number of its variants lack robustness against common corruptions (as shown later in Section 4.3).

By definition, ReLU blocks out any negative input but does not alter positive input at all. Therefore, the intermediate layer features inside ReLU-based networks become sparse. Sparse features maximise the inter-class distance to gain high classification accuracy but could compromise the classification performance with their higher intra-class dispersion (see Figure 4.1(a)). This compromise does not harm CNN's overall performance so long as the evaluation dataset is clean. However, the compromise (i.e., feature sparsity) gets exposed in the presence of input corruptions - especially High Frequency corruptions (HFc), e.g., Gaussian noise. HFc, even when visually

[1] We use the word distortion and corruption interchangeably in this work.

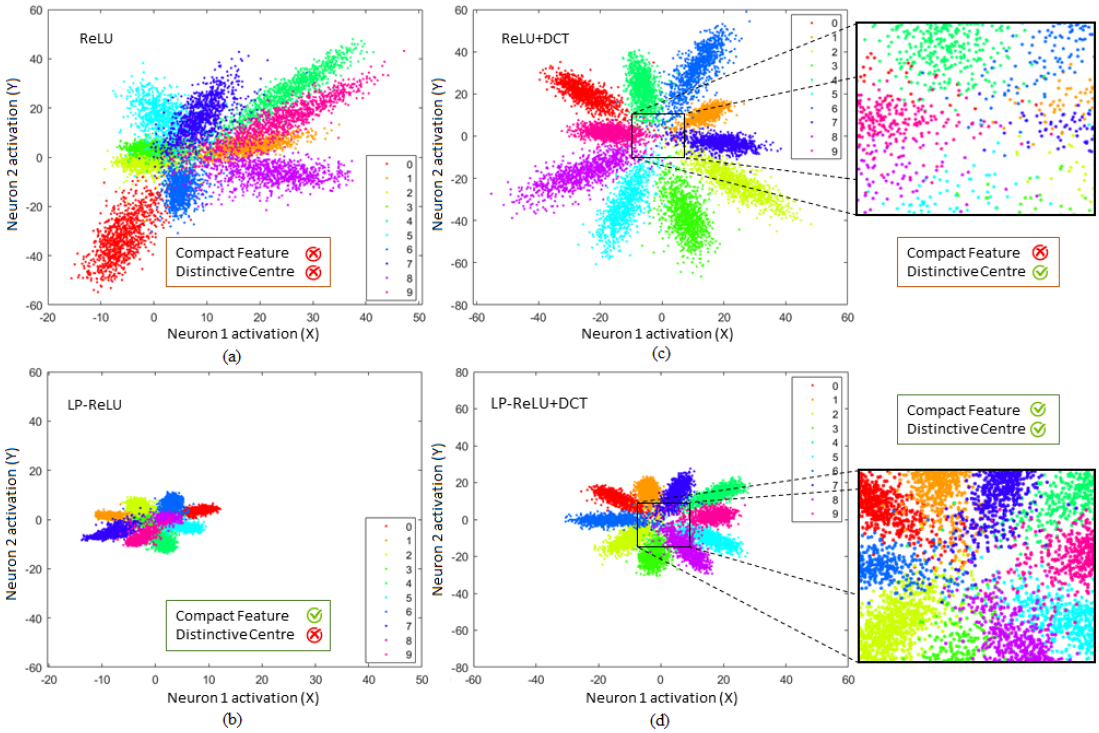

Figure 4.1: Visualising deep feature characteristics for different network setting on the MNIST dataset. **(a)** ReLU allows activations or features to flow through network layers without any filtering. This introduces sparsity in the feature space. Such sparsity can cause shift in the feature space in the presence of data corruption, even when corruption is visually imperceptible. **(b)** Our proposed AF has low-pass filter built-in, which enforces feature compactness. This, in effect, limits the internal feature shift and resists corrupted features from drifting away. **(c)-(d)** As discussed later in Section 4.2.1, robustness against LFc demand better distinction for weak features (weak features reside in the centre of the above plots). DCT data augmentation [55] improves robustness for both AFs, i.e., ReLU **(c)**, and LP-ReLU **(d)**- especially against LFc by increasing the inter-class distance (distance among weak features) in the centre. See the zoomed insets for comparison. Despite overall improvement, feature sparsity still exists in ReLU **(c)**, which makes it vulnerable to corruptions. All plots in this figure use features taken from a CNN proposed to visualise deep features and decision boundary (details in Figure 4.10, Section 4.4).

imperceptible, can cause the corrupted data to drift away to a different part of the feature space [50] in the absence of a low-pass filter. This leads to misclassification.

The classic signal processing fix to a HFc is low-pass filtering. For example, voice recorders use low-pass filtering to cancel the 'hiss' noise originating from electromagnetic interference. As shown by Campbell et al., [143], human visual system also has limited sensitivity to abrupt or high frequency changes owing to an inherent low-pass

filtering process. This particular property of human visual system is later exploited in JPEG image compression as well [118]. Inspired by these observations, we incorporate a low-pass filtering mechanism inside the proposed AF, namely, Low-Pass ReLU (LP-ReLU, see Figure 4.2) and show its efficacy against corruptions – especially HFc. LP-ReLU resists features from drifting, even when there is HFc by enforcing a compact feature space (see Figure 4.1(b)).

As for Low Frequency corruptions (LFc), one might think of AFs with high pass filtering as a potential fix. However, corrupted features from LFc substantially overlap with meaningful image features (details in Section 4.2). Using a high pass filter can exacerbate the problem by discarding meaningful features along with the corrupted ones. Therefore, LFc demand another way around. We find that a data augmentation method based on DCT [55] can further boost overall robustness, especially against LFc. DCT augmentation provides greater distinction for weak features (at the centre region of the feature space) with LFc (see Figures 4.1(c) and 4.1(d)). Feature characteristics for both HFc and LFc are discussed in details in Section 4.2.

We evaluate our method's efficacy on benchmark datasets (CIFAR-10-C and Tiny ImageNet-C [31]) containing common corruptions. We further conduct performance stability tests on perturbed datasets (CIFAR-10-P and Tiny ImageNet-P [31]). Our experimental results show that LP-ReLU provides better robustness and stability against corruptions compared to contemporary works.

We also stress the importance of visualisation of the learned decision space to better understand the dynamics of CNNs' robustness against common corruptions. To this end, we propose a way of visualising the learned decision space and how a corrupted input triggers misclassification. We show that the learned decision boundaries extend to infinity and extrapolate predictions in the space beyond the scope of the data manifold. This observation reiterates the importance of a compact feature space to suppress the detrimental effects of corruptions.

To summarise, we make the following contributions in this chapter:

- We analyse image corruptions from a frequency perspective and show that CNN's weakness against corrupted data can be addressed by using appropriate AFs and augmentation techniques.

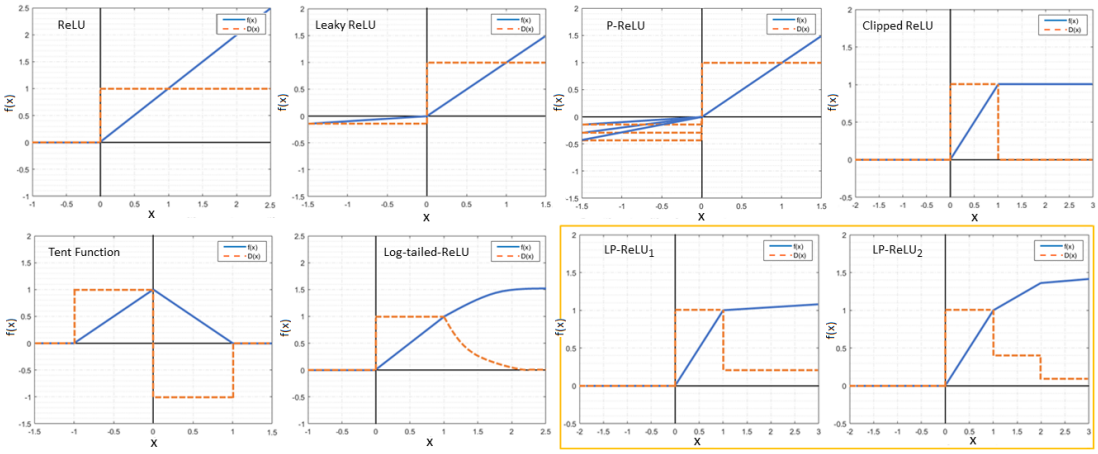

Figure 4.2: Activation functions $f(x)$ and their corresponding 1^{st} derivatives $D(x)$. The proposed variants of AFs, i.e., LP-ReLU$_1$ and LP-ReLU$_2$ are highlighted in the lower right panel.

- We propose novel AFs (LP-ReLU) by embedding low-pass filtering properties into ReLU.
- To complement LP-ReLU in tackling LFc, we use DCT based augmentation [55].
- To better understand CNNs' overall robustness against data corruptions, we propose a method to visually illustrate CNNs' decision boundaries and their intermediate feature space.
- Finally, we extensively evaluate the accuracy and stability of our proposed method on benchmark datasets and demonstrate the efficacy of our method in improving CNNs' robustness.

4.2 Proposed Approach

Achieving robustness against corruptions requires an in-depth understanding of their properties. In this section, we set the foundation for our proposed approach by analysing different corruptions from a Fourier perspective. For all our experiments presented in Figures 4.3, 4.4, and 4.5, WRN-40-2 is used.

4.2.1 Frequency Domain Analysis

Commonly found image corruptions can be broadly categorised into two types: HFc and LFc [37]. We argue that successfully handling the entire corruption spectrum, i.e., both HFc and LFc, is at the core of achieving a desired level of robustness.

Noise introduces sudden spikes in the signal and thereby falls in the HFc category [37]. Blur, on the other hand, is another common form of corruption that removes high frequency information and thereby falls in the LFc category (e.g. Gaussian blur) [37].

High Frequency Corruptions. In signal processing applications, low-pass filtering is a common operator to remove nuisance frequencies beyond a cut-off threshold. From experimental analysis, we observe that an image with HFc induces activations with a higher magnitude compared to its clean counterpart (see Figure 4.3). In other words, convolution kernels fire strongly on 'noisy' parts of an input with HFc. These hyperactive features coming out of the noise rather than the signal are also fed to ReLU. As ReLU does not filter input in the positive domain, i.e., for $x > 0$, such features are allowed to flow through the network layers. Consequently, these corrupted features exploit the sparsity offered by ReLU and trigger heavy shift in the feature space. This way, the features easily end up on the wrong side of the decision boundary, causing misclassification (details in Section 4.3.5).

Low Frequency Corruptions. As stated earlier, LFc do not have an easy filtering fix. The underlying reasons can be better understood if we move from spatial to frequency domain.

As observed in Figure 4.3, convolution kernels fire weakly on images with LFc as such corruptions usually blunt the visual features. Contrary to HFc, this implies that the LFc induce relatively weaker features[2] that congregate in the centre region of the feature space (as shown in Figure 4.1). For example, a vertical edge detector kernel would produce strong output features each time it convolves over a sharp vertical edge present in an image. However, the same kernel would produce weaker features if the vertical edge is not sharp enough (e.g., loss of sharpness due to LFc).

[2]By weak activation or feature, we refer to the smaller magnitude of absolute values produced from each convolution operation.

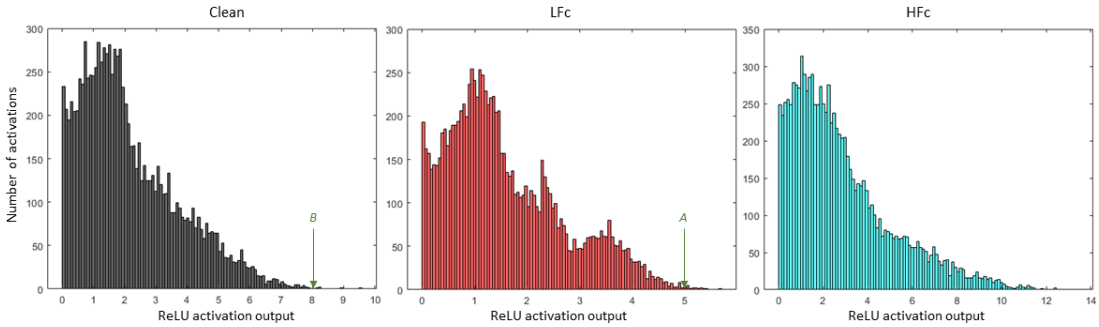

Figure 4.3: Histograms of ReLU output, averaged across all the layers in WRN-40-2, from clean images in CIFAR-10 and Tiny ImageNet (**Left**), images with LFc (**Middle**), and images with HFc (**Right**) from CIFAR-10-C and Tiny ImageNet-C. Horizontal axis denotes the magnitude of a feature or activation and the vertical axis denotes the number of activations having a certain magnitude. These plots show that the HFc indeed induce hyperactive features with large output magnitude (**Right**) and LFc induce relatively weak features (**Middle**). The histogram with weak features (LFc) overlaps with the clean one (**Left**) and hence LFc are harder to distinguish. These plots on clean, LFc and HFc also provide a heuristic on the potential cut-off point initialisation values (A and B) in the proposed LP-ReLU as discussed in Section 4.2.5.

It is known from DCT analysis that most of the energy (of a natural image) is concentrated on the low frequency spectrum [118]. Unlike HFc, features from LFc substantially overlap with meaningful features. Consequently, the classic fix, i.e., using a high-pass filter, is not a viable option as such a filtering might eliminate legitimate features as well as corrupted ones. This loss of information would not be recoverable and result in performance deterioration.

We argue that a data augmentation technique capable of mimicking LFc can significantly boost robustness not only against LFc but also against HFc.

4.2.2 Proposed Activation Function: LP-ReLU

Inspired by the human visual system and conventional signal processing fix for HFc, we design two variants of low pass filters for our proposed AFs. Before explaining further, there are a couple of key differences between the proposed and conventional low-pass filtering we would like to highlight:

- Unlike the conventional fix, where everything beyond a cut off frequency is completely ignored, we use a soft filtering technique with a signal attenuation

factor that we call $Filtering\ Factor$. Note that completely cutting off the signal beyond a threshold would resurface the vanishing gradient problem (similar to C-ReLU).

- In a conventional low-pass filtering operator, a cut-off frequency is chosen based on the maximum frequency available in the signal of interest (following Nyquist Theorem [144]). However, computing such a cut-off frequency is complicated for visual datasets like images. Hence, we design two low pass filter variants in our AFs namely, LP-ReLU$_1$ (one cut-off point) and LP-ReLU$_2$ (two cut-off points) with different cut-off point selection strategies.

4.2.3 LP-ReLU$_1$

Equation 4.1 represents the first variant of our proposed AF, i.e., LP-ReLU$_1$. A closer look into Equation 4.1 reveals that LP-ReLU$_1$, upto a threshold A, is equivalent to ReLU. Beyond this threshold, the input features get attenuated by a $Filtering\ Factor$ α ($\alpha \in [0, 1]$). Note that $\alpha = 0$ will make LP-ReLU$_1$ equivalent to C-ReLU. $\alpha = 1$, on the other hand, would make LP-ReLU$_1$ equivalent to ReLU.

$$F(x) = \begin{cases} 0, & x \leq 0, \\ x, & x \in (0, A], \\ A + \alpha(x - A), & x > A \end{cases} \tag{4.1}$$

Equation 4.2 denotes the derivative of LP-ReLU$_1$ at each piece-wise linear stage. It is evident that LP-ReLU$_1$ has a slope at every point beyond the origin, which is a much-desired property for training any neural network. See the graphical representations of LP-ReLU$_1$ and its derivative in Figure 4.2.

$$D(x) = \begin{cases} 0, & x \leq 0, \\ 1, & x \in (0, A], \\ \alpha, & x > A \end{cases} \tag{4.2}$$

4.2.4 LP-ReLU$_2$

In contrast to LP-ReLU$_1$, LP-ReLU$_2$ has two cut-off points A, and B (where $A < B$) in series and two corresponding $Filtering\ Factors$ α, and β (where $\alpha > \beta$) as can be seen from Equation 4.3. The graphical representations of LP-ReLU$_2$ and its derivative are shown in Figure 4.2.

LP-ReLU$_2$'s effectiveness against corruptions can be attributed to the following reasons:

- **Phase 1 (soft) filtering.** A relatively larger α, i.e., $\alpha \to 1$ (for $x \in (A, B]$) allows greater feature sparsity in the centre, i.e., sparsity for weak features in the centre region where LFc congregate (as shown in Figure 4.1). This initial soft filtering gives our network enough sparsity in the centre to accommodate weak features in distinct regions.
- **Phase 2 (hard) filtering.** A relatively smaller β, i.e., $\beta \to 0$ (for $x > B$) means hard suppression for large x, i.e., towards the perimeter of the feature space. This hard filtering stage suppresses noise in the signal, and limits feature shift by ensuring a compact feature space.

$$F(x) = \begin{cases} f_1(x) = 0, & x \leq 0, \\ f_2(x) = x, & x \in (0, A], \\ f_3(x) = A + \alpha(x - A), & x \in (A, B], \\ f_4(x) = max(f_3(x)) + \\ \qquad \beta(x - max(f_3(x)), & x > B \end{cases} \tag{4.3}$$

Equation 4.4 represents the derivative of LP-ReLU$_2$. Much like LP-ReLU$_1$, LP-ReLU$_2$ also has a slope at every piece-wise linear stage.

$$D(x) = \begin{cases} 0, & x \leq 0, \\ 1, & x \in (0, A], \\ \alpha, & x \in (A, B], \\ \beta, & x > B \end{cases} \tag{4.4}$$

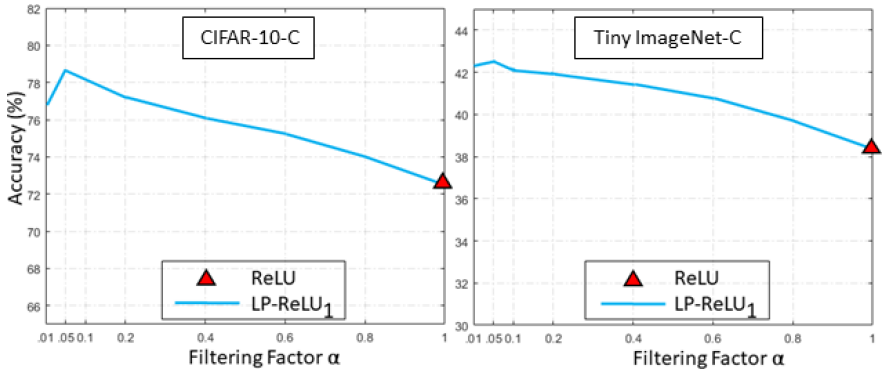

Figure 4.4: Empirical analysis for a suitable $Filtering\ Factor\ \alpha$ in LP-ReLU$_1$. Notice that $\alpha \approx 0.05$ yields the best results and $\alpha = 1$ simply replicates the results of ReLU. WRN-40-2 is used as the backbone for these experiments.

Overall, LP-ReLU$_2$ shows us that successfully handling the entire corruption spectrum requires separate approaches, i.e., sparsity for LFc and compactness for HFc. As shown later in this chapter, LP-ReLU$_2$ has a slight advantage over LP-ReLU$_1$ in accuracy but LP-ReLU$_1$ is faster to train.

4.2.5 Cut-off Point and Filtering Factor

For LP-ReLU$_1$, the cut-off value A is set as a learnable parameter initialised with the value 6 based on heuristic [69]. $Filtering\ Factor\ \alpha$ can be a constant or cast as a trainable parameter. In this chapter, based on the analysis in Figure 4.4, we use $\alpha = 0.05$ as it yields the best performance as a constant parameter. When cast as a trainable parameter, we encourage using 0.05 as the initial value while making sure $\alpha < 1$ at all times.

For LP-ReLU$_2$, we cast the cut-off values A and B as learnable parameters and ensure at all times during training they maintain a buffer in between, i.e., $A < B$. We investigate a histogram-based approach to initialise these hyperparameters (with $A = 5$ and $B = 8.1$). In Figure 4.3, we plot the ReLU output (averaged over all layers) histograms for the clean, LFc, and HFc. It can be derived that ReLU output beyond B is likely to be noise. On the other hand, output under the threshold A could either be the true signal or LFc. This is why to initialise, we choose A as the first cut-off point (with a soft $Filtering\ Factor\ \alpha$) and B as the second cut-off point (with a hard $Filtering$

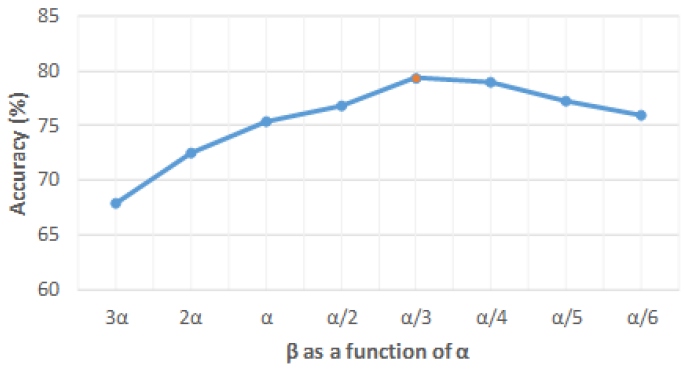

Figure 4.5: Empirical analysis of β as a function of decreasing α in LP-ReLU$_2$. Here, α is the phase 1 and β is the phase 2 $Filtering\ Factor$. In this experiment, WRN-40-2 is used as the backbone and CIFAR-10-C test fold as the dataset. This experiment serves as a heuristic to initialise β regardless of the dataset to avoid data dependency.

$Factor\ \beta$). For the experiments in Figures 4.3, 4.4, and 4.5, 15% of the training set is used as the validation subset on which the statistics are derived.

Both $Filtering\ Factor$s, α and β are learnable parameters. α is initialised in the same way as described in LP-ReLU$_1$, and β is cast as a function of α. At all times during training, the $\alpha > \beta$ relation is maintained. From empirical analysis (see Figure 4.5), we initialise with $\beta = \frac{\alpha}{3}$. During training, we observed that the performance variance with different initialisations is marginal as long as these hyperparameters oscillate between optimal and near-optimal values. This allows the hyperparameter initialisation to be flexible and transferable to other datasets.

4.2.6 DCT Augmentation

To further strengthen CNN's robustness against corruptions, especially LFc, we incorporate DCT data augmentation [55] alongside LP-ReLU. DCT data augmentation randomly drops low impact high frequency information based on DCT coefficients. Because of the randomness, some low frequency information gets dropped as well, which is found to be more effective as it introduces data diversity [55]. Unlike JPEG compression [118], a block size equal to the input's spatial resolution is used. DCT data augmentation is corruption agnostic but generalises well to a wide array of common corruptions, especially LFc. The impact of DCT data augmentation coupled with

Figure 4.6: Sample images from DCT augmentation [55]. The first image in each row is the clean one and drop of information based on DC coefficient intensifies from left to right.

LP-ReLU, can be visually observed in Figure 4.1 as the centre region (LFc) becomes much more distinct. Samples derived from DCT augmentation can be seen in Figure 4.6.

4.3 Performance Analysis

In this section, we introduce the datasets, provide details on implementation and evaluation metrics and then discuss how the proposed network fares against the existing ones.

4.3.1 Datasets

To evaluate robustness of a deep classifier, the following datasets are used:

CIFAR-10 and Tiny ImageNet[3]**.** CIFAR-10 [127] has 50,000 $32 \times 32 \times 3$ clean images equally distributed in 10 classes. The split is 50,000 training images and 10,000 test images. Tiny ImageNet [145] is an ImageNet [1] subset comprising of 100,000 clean training images with 200 classes. Each image has a spatial resolution of $64 \times 64 \times 3$. Each of the classes has 500 training and 50 test images (total 10,000 test images).

[3]From here on, whenever we mention clean data set or clean accuracy, we refer to the original dataset contained in either CIFAR-10 or Tiny ImageNet.

CIFAR-10-C and Tiny ImageNet-C. CIFAR-10-C [31] is effectively the corrupted variant of the CIFAR-10 test set. There are a total of 19 corruption types categorised into noise, blur, weather, and digital corruptions. Each image within a particular corruption type undergoes five increasing levels of severity to ensure a thorough evaluation. Thereby, CIFAR-10-C has $19 \times 10,000 \times 5$ or $950,000$ test images in total.

The corruption induction process for Tiny ImageNet-C [31] is identical to that of CIFAR-10-C. In total, Tiny ImageNet also has $19 \times 10,000 \times 5$ or $950,000$ test images.

CIFAR-10-P and Tiny ImageNet-P. Both these datasets contain perturbed images of the clean datasets to test the performance stability. Each example here is a short video of about 30 frames containing 10 perturbations, including noise (Gaussian and shot), blur (motion and zoom), weather conditions (snow and brightness), and affine transformations (translate, rotation, scaling, and tilt). Frames within a video clip contain progressively increasing perturbations to challenge network stability. Unlike corruptions, stability is not measured with classification accuracy. Rather, the mean Flip Probability (mFP) is calculated over all perturbation categories. A flip event occurs when two consecutive frames' predictions mismatch. A lower mFP score indicates a more robust network.

4.3.2 Implementation and Training Details

For an in-depth evaluation of the proposed method, we use ResNet-based [26] WRN [27] as the base model. Unlike ResNet, where performance gain comes with increasing depth, WRN's architecture pivots around a central argument that shallower networks with greater width lead to higher performance gain. WRNs are denoted as *WRN-d-k* where d is the number of convolutional units and k is the widening factor considering ResNet has a widening factor of 1, i.e., $k = 1$. We use *WRN-40-2* with ReLU for the base model and replace ReLU with LP-ReLU for the proposed network. We use SGD with $Momentum$ 0.9. We train the network for 160 epochs with initial $learning\ rate$ set to 0.1. The initial learning rate is dropped by a factor of 0.2 after 50, 100, and 140 epochs.

$Batch - size$ is set to 128 with an $L2Regularisation$ factor of 0.0005. $Zero - centre$ input image normalisation is used for all experiments.

For training variants with DCT data augmentation, we follow the training protocol described in [55]. In DCT augmentation, each image is first transformed into the frequency domain. Later, frequency components with relatively less impact on the image are dropped based on a chosen threshold. Inverse DCT is performed afterwards with the remaining DCT coefficients for reconstructing the image in spatial domain. These reconstructed images are fed to train the deep network.

4.3.3 Evaluation Metrics

We follow the protocol outlined in [31] for evaluating robustness against corruption and performance stability on perturbations. The protocol is briefly described below:

Corruption Robustness on CIFAR-10-C and Tiny ImageNet-C. To evaluate robustness against corruption, we calculate the Top-1 classification accuracy as in [31]. This involves validating the class prediction with the highest probability (for each test sample) against the ground truth. The ratio of correct predictions and total test samples is used as the Top-1 accuracy.

Perturbation Stability on CIFAR-10-P and Tiny ImageNet-P. Instead of Top-1 accuracy, as proposed in [31] stability is measured against Flip Probability (FP) which provides us a quantitative insight of a network's tendency to flip prediction with increasing perturbation. To calculate FP, let us denote k perturbation sequences (each with v number of frames) with $S = \left\{\left(x_1^{(i)}, x_2^{(i)}, \ldots, x_v^{(i)}\right)\right\}_{i=1}^{k}$. For a fixed i, i.e., perturbation type k, $x_1^{(i)}$ denotes the clean image (no perturbation) and $x_v^{(i)}$ denotes a frame with maximum (k type) perturbation. Considering our CNN classifier as a function F, the Flip Probability of Network $F : x \rightarrow \{class_1, class_2, \ldots, class_n\}$ on perturbation sequence S is:

$$FP_p^F = \frac{1}{k(l-1)} \sum_{i=1}^{k} \sum_{j=2}^{v} 1\left(F\left(x_j^{(i)}\right) \neq F\left(x_{j-1}^{(i)}\right)\right) \tag{4.5}$$

Taking the mFP across all perturbations denotes the stability metric used in this chapter.

Clean Accuracy on CIFAR-10 and Tiny ImageNet. Besides evaluating on distortions and perturbations, we evaluate the networks on the clean dataset as well. To compare the performance on clean datasets, we calculate the Top-1 accuracy.

Table 4.1: Top-1 classification accuracy (%) for different network configurations.

Network	AF	DCT	CIFAR-10-C	Tiny Imagenet-C
VGG-19	ReLU	✓	76.6	40.1
ResNet-101	ReLU	✓	78.5	42.9
WRN-40-2	ReLU	✓	79.1	43.2
VGG-19	ReLU	×	68.5	35.9
ResNet-101	ReLU	×	70.2	37.8
WRN-40-2	ReLU	×	72.7	38.6
VGG-19	LP-ReLU$_2$	✓	84.2	45.5
ResNet-101	LP-ReLU$_2$	✓	86.5	47.7
WRN-40-2	LP-ReLU$_2$	✓	89.2	51.9

Table 4.2: Top-1 classification accuracy (%) for different hyperparameter configurations (Network: WRN-40-2 with LP-ReLU$_2$ + DCT).

A, α	B, β	CIFAR-10-C	Tiny Imagenet-C
Learnable	Learnable	89.2	51.9
Learnable	Frozen	88.1	50.3
Frozen	Learnable	88.4	50.1
Frozen	Frozen	87.9	49.8

4.3.4 Comparative Performance Evaluation

WRN-40-2 architecture performs better than VGG-19 and ResNet-101 on both of the corrupted datasets (Table 4.1), and therefore we use it as the backbone network for our proposed method. As Table 4.2 shows, learnable hyperparameters perform better as opposed to freezing them at the initialisation values. For all our experiments, we cast the hyperparameters as learnable unless mentioned otherwise.

Table 4.3: Top-1 classification accuracy (%) on 19 individual corruptions from CIFAR-10-C and Tiny ImageNet-C with different network configurations. *WRN-40-2* architecture is used in all these networks as it performs best as shown in Table 4.1. LP-ReLU$_2$ coupled with DCT augmentation performs best. Accuracy is averaged over all five severity levels and reported in the last row. The penultimate row (Clean) reports accuracy on the respective distortion-free dataset. Best in each row is highlighted in bold.

	CIFAR-10-C					Tiny ImageNet-C				
Corruption	ReLU	LP-ReLU$_1$	LP-ReLU$_1$+DCT	LP-ReLU$_2$	LP-ReLU$_2$+DCT	ReLU	LP-ReLU$_1$	LP-ReLU$_1$+DCT	LP-ReLU$_2$	LP-ReLU$_2$+DCT
Brightness	93.8	90.3	89.2	**95.4**	94.0	58.3	58.1	57.3	**59.3**	58.6
Contrast	74.1	74.2	75.1	75.3	**80.7**	33.4	29.3	**34.1**	30.2	33.2
Defocus_blur	88.1	85.3	86.3	80.3	**94.5**	44.2	45.9	52.6	48.6	**56.2**
Elastic	81.3	79.6	82.5	79.9	**90.5**	45.9	45.1	48.3	45.3	**51.5**
Fog	81.7	84.5	85.2	85.6	**91.2**	48.5	49.1	52.4	49.1	**53.7**
Frost	64.4	74.8	84.6	75.9	**90.4**	35.2	39.4	48.1	40.2	**49.8**
Gauss_blur	84.1	85.9	91.3	88.4	**93.2**	35.6	38.3	51.6	40.8	**56.7**
Gauss_noise	51.1	69.5	**88.4**	69.4	85.1	15.1	23	38.9	22.9	**42.5**
Glass	47.8	53.7	73.8	53.4	**78.6**	21.5	23.2	37.4	23.4	**40.6**
Impulse_noise	53.6	78.8	**88.3**	77.6	86.3	27.3	48.2	52.8	48.2	**54.3**
Jpeg	75.8	82.6	86.2	84.5	**94.2**	41.2	48.5	52.9	50.9	**56.1**
Motion_blur	80.2	78.3	82.6	81.2	**90.2**	46.8	47.9	55.5	50.8	**57.9**
Pixelate	73.6	72.9	84.1	73.4	**91.7**	41.7	43.2	48.9	44.6	**51.2**
Saturate	88.1	91.7	89.9	**91.9**	91.6	55.4	56.1	56.4	56.5	**57.7**
Shot_noise	59.2	76.8	86.3	77.2	**88.9**	33.5	45	52.3	45.2	**55.1**
Snow	73.3	78.3	81.1	77.9	**89.3**	45.2	46.7	52.9	45.3	**55.2**
Spatter	73.5	84.9	84.6	85.6	**86.9**	44.9	46.2	46.8	46.8	**47.6**
Speckle	56.7	77.8	85.3	76.9	**88.3**	21.5	33.4	45.2	33.6	**49.9**
Zoom_blur	80.6	76.2	**92.6**	78.5	91.6	37.7	38.3	54.3	41.6	**58.1**
Clean	96.0	96.1	96.3	96.2	**96.4**	61.1	61.3	61.2	61.3	**61.5**
Average Accuracy	72.7	78.7	85.1	79.5	**89.2**	38.6	42.4	49.4	43.3	**51.9**

CIFAR-10-C and Tiny ImageNet-C. According to Table 4.3, WRN-40-2 with ReLU as the AF achieves an average accuracy of 72.7% over all corruptions on CIFAR-10-C. When LP-ReLU$_1$ replaces ReLU, overall corruption accuracy stands at 78.7% (an improvement by 6%). A closer inspection reveals that LP-ReLU$_1$ invokes greater gains on HFc compared to the baseline (the baseline has 77.4% and 55.1% while LP-ReLU$_1$ has 79.5% and 75.7% accuracy on LFc and HFc, respectively). This complements the role low-pass filtering plays in limiting the misclassification against HFc.

LP-ReLU$_2$ which is specifically designed to allow greater sparsity for weaker activations, achieves 80.5% LFc and 75.15% HFc accuracy.

DCT augmentation complements both LP-ReLU variants by boosting overall robustness- especially against LFc. LP-ReLU$_1$ coupled with DCT augmentation achieves 84.6% accuracy on LFc and 87% accuracy on HFc. LP-ReLU$_2$ coupled with DCT augmentation achieves 89.9% accuracy on LFc and 87.1% accuracy on HFc.

It is worth noting that the soft phase 1 filtering in LP-ReLU$_2$ actually allows DCT augmentation to take full effect and improve performance, especially against LFc. To be specific, the soft phase 1 filtering provides weak features (in the centre, as shown in

Table 4.4: Average Top-1 classification accuracy (%) comparison. LP-ReLU variants along with DCT augmentation demonstrate better robustness on both CIFAR-10-C and Tiny Imagenet-C. LP-ReLU$_1$ + DCT obtains 0.9% and 4.8% improvement over AugMix on CIFAR-10-C and Tiny Imagenet-C respectively. LP-ReLU$_2$ + DCT obtains 5% and 7.3% improvement over AugMix on CIFAR-10-C and Tiny Imagenet-C, respectively. Best combination is highlighted in blue.

Network	CIFAR-10-C	Tiny Imagenet-C
ReLU	72.7 ± 0.82	38.6 ± 0.91
Leaky-ReLU	67.2 ± 0.66	32.9 ± 0.72
P-ReLU	68.8 ± 0.45	33.3 ± 0.53
Clipped-ReLU	73.2 ± 0.60	39.7 ± 0.66
Tanh	66.9 ± 0.79	32.3 ± 0.84
Tent	73.9 ± 0.29	40.5 ± 0.33
Cutout [42]	70.1 ± 0.39	40.1 ± 0.44
Mixup [41]	72.5 ± 0.45	40.8 ± 0.49
CutMix [40]	70.8 ± 0.36	39.7 ± 0.40
AutoAugment [36]	78.8 ± 0.34	42.5 ± 0.37
AugMix [38]	84.2 ± 0.28	44.6 ± 0.30
DeepAugment [138]	78.5 ± 0.11	42.9 ± 0.20
BNS [48]	78.4 ± 0.25	44.3 ± 0.39
LP-ReLU$_1$	78.7 ± 0.27	42.4 ± 0.29
LP-ReLU$_1$ + DCT	85.1 ± 0.25	49.4 ± 0.28
LP-ReLU$_2$	79.5 ± 0.28	43.3 ± 0.32
LP-ReLU$_2$ + DCT	$\mathbf{89.2 \pm 0.24}$	$\mathbf{51.9 \pm 0.30}$

Table 4.5: Comparison of average Top-1 classification accuracy (%). The combination of LP-ReLU$_2$ and DCT augmentation outperforms others. Best combination is highlighted in blue.

Network	CIFAR-10-C	Tiny Imagenet-C
ReLU + DCT	77.1 ± 0.55	41.2 ± 0.61
Clipped-ReLU + DCT	78.2 ± 0.37	42.2 ± 0.72
LP-ReLU$_2$ + AutoAugment	85.3 ± 0.36	45.2 ± 0.28
LP-ReLU$_2$ + AugMix	86.5 ± 0.46	47.9 ± 0.50
LP-ReLU$_2$ + DCT	$\mathbf{89.2 \pm 0.24}$	$\mathbf{51.9 \pm 0.28}$

Figure 4.1) room to be relatively sparser. This sparsity is vital for distinguishing weak features, which otherwise congregate together. On the other hand, the hard phase 2 filtering constrains hyperactive HFc feature space and limits features from drifting away.

For a more in-depth quantitative analysis, Figure 4.7 compiles performance on each type of corruption present in CIFAR-10-C at each available corruption severity level. LP-ReLU (LP-ReLU$_2$ + DCT augmentation) exhibits the best consistency across

Figure 4.7: Performance comparison between LP-ReLU (LP-ReLU$_2$ + DCT) and contemporary methods with progressively increasing corruption in CIFAR-10-C. 1 and 5 denote lowest and highest level of corruption respectively. LP-ReLU exhibits consistently better performance across all severity levels.

severity levels. The performance of ReLU network generally drops at a much faster rate with increasing severity compared to LP-ReLU.

As can be seen from Table 4.4, LP-ReLU$_2$ + DCT augmentation achieves SOTA Top-1 classification accuracy on both the corrupted datasets (89.2% on CIFAR-10-C and 51.9% on Tiny ImageNet-C). These are 5% and 7.3% better than AugMix [38] on CIFAR-10-C and Tiny ImageNet-C respectively. LP-ReLU$_2$ + DCT outperforms other combinations of AF and augmentation methods as shown in Table 4.5. We attribute the performance gain to the method's (LP-ReLU2 + DCT) ability to address both HFc and LFc.

CIFAR-10-P and Tiny ImageNet-P. While the corrupted datasets test the overall robustness, progressively increasing perturbations in CIFAR-10-P and Tiny ImageNet-P test the stability of network performance. Networks suffering from robustness deficiency provide erratic predictions with marginally varying perturbations. Robust networks, on the other hand, do not flip predictions unless the input corruption is substantial. As can be seen from Figure 4.8, our proposed method, i.e., LP-ReLU$_2$ + DCT augmentation, achieves SOTA mFP both on CIFAR-10-P (mFP 1.08%) and Tiny ImageNet-P (mFP 2.88%). This is lower than the baseline (mFP 4.09% on CIFAR-10-P and mFP 11.8% on Tiny ImageNet-P) and the next best AugMix [38] (mFP 1.49% on CIFAR-10-P and mFP 4.1% on Tiny ImageNet-P).

CIFAR-10 and Tiny ImageNet (clean). Although the proposed LP-ReLU has been designed to deal with corrupted images, both LP-ReLU variants maintain comparable results on the clean datasets with small but consistent improvements over ReLU. ReLU achieves an accuracy of 96% on clean CIFAR-10 whereas LP-ReLU$_1$ and LP-ReLU$_2$ achieve 96.1% and 96.2% clean accuracy respectively. DCT augmentation slightly improves clean accuracy for both LP-ReLU variants (see Table 4.3). As for Tiny ImageNet-C, similar performance gain trend is observed (see Table 4.3).

We attribute our method's better robustness and stability across corruptions and perturbations to our separate approaches in handling LFc and HFc. Contemporary data augmentation-based methods [38, 40, 42, 41, 138] do not account for the specific demands from corruptions residing in opposite ends of the frequency spectrum. To the best of our knowledge, only [37] stresses the importance of understanding the robustness issue from a Fourier perspective. Nonetheless, they resort to a pre-existing augmentation technique (AutoAugment[36]) that was originally proposed to improve clean accuracy – not robustness. AutoAugment does improve robustness as well (as reported in [37]), but the improvement does not catch up with ours. AugMix [38] further builds on AutoAugment and likewise – do not consider the corruptions through the lens of Fourier transform. DeepAugment [138] resorts to auto-encoder models for augmentation, but the synthetic data alone is not enough for significant performance gain.

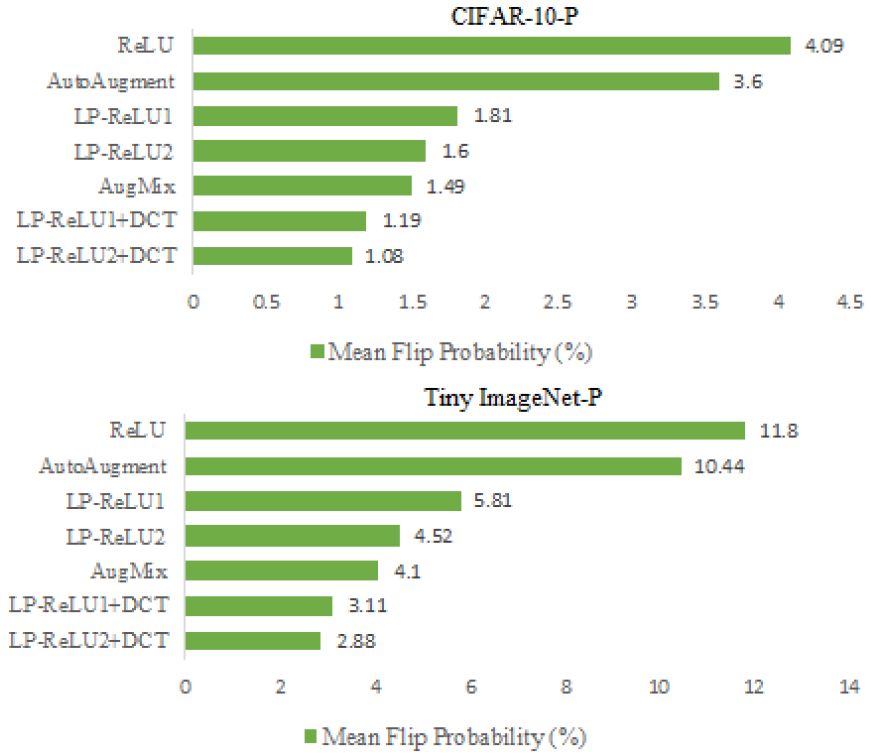

Figure 4.8: LP-ReLU$_2$ + DCT has the lowest mFP on both CIFAR-10-P (top) and Tiny Imagenet-P (bottom,) which is a testament to its performance stability across 10 perturbations present in these benchmark datasets.

4.3.5 Comparing Shift in Feature Space

In this section, we analyse the impact of input corruption in the intermediate feature space of a CNN. To be more specific, we demonstrate how a ReLU network fares against an LP-ReLU network in terms of feature shift due to corruption.

Let us assume a CNN as a feature extractor function $\mathcal{F}(x) \rightarrow \mathbb{C}_L$ where $\mathbb{C}_L$ denotes the feature set $\mathbb{C}$ at layer L. The input to $\mathcal{F}$, i.e., x can either be an image from the clean set X ($x \in X$) or from a corrupted set $\hat{X}$. $\hat{X}$ consists of elements from X that has undergone a certain corruption operation, i.e., $Corruption_{\Delta m}(X) \rightarrow \hat{X}$ ($x \in \hat{X}$) where Δm is the corruption magnitude. For a particular CNN, i.e., $\mathcal{F}$ to be robust against input corruption, $\mathcal{F}$ has to produce similar $\mathbb{C}_L$ for an input from X and its corrupted counterpart $\hat{X}$ (ideally exactly same) as shown in Equation 4.6.

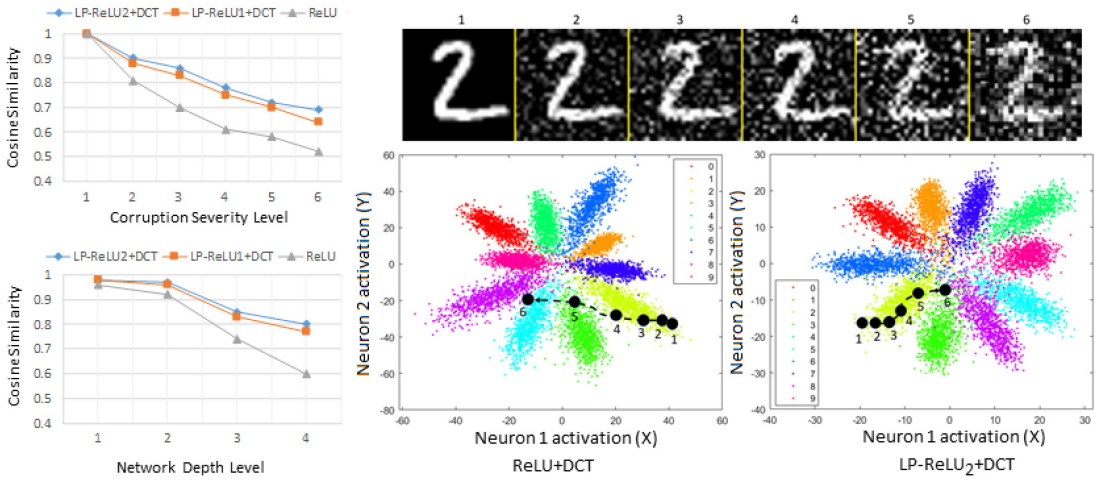

Figure 4.9: (Top Left) With increasing corruption severity, similarity between the clean (level 1) and corrupted (level $[2, 6]$) features fall sharply in ReLU network suggesting heavy shift in the intermediate feature space (lower similarity score refers to higher shift in feature space). Similarity score is calculated in the following severity order: $(1 \rightarrow 1), (1 \rightarrow 2), (1 \rightarrow 3), (1 \rightarrow 4), (1 \rightarrow 5), (1 \rightarrow 6)$. Note that similarity between the clean image and itself $(1 \rightarrow 1)$ is 1. **(Bottom Left)** The impact of corruption magnifies as similarity falls sharply with increasing network depth in ReLU network. Networks are divided into four depth levels ($[1, 2, 3, 4]$) and similarity is then calculated between the corresponding depth levels, i.e., $(1 \rightarrow 1), (2 \rightarrow 2), (3 \rightarrow 3), (4 \rightarrow 4)$ on clean and corrupted datasets. **(Right)** With increasing input corruption (Gaussian noise in this case), features drift far away in ReLU network resulting in misclassification of the digit '2'. Because of a compact feature representation in LP-ReLU networks, shift in feature space is constrained and features stay close to where they belong. It is worth noticing the scale in X and Y axes to perceive the relative feature compactness.

$$\begin{aligned} &\mathcal{F}(X) \approx \mathcal{F}(Corruption_{\Delta m}(X)), \quad \Delta m \rightarrow \text{small} \\ &\mathcal{F}(X) \approx \mathcal{F}(\hat{X}), \end{aligned} \tag{4.6}$$

We use the Cosine Similarity (CS) [146] to evaluate the effects of shift in feature space on increasing corruption severity. CS is widely used to calculate distance in high dimensional space where Euclidean or $L2$ distance does not work well [146]. CS gives a measure of the angular distance between two high dimensional vectors. We measure CS on six severity levels of corruptions as annotated in CIFAR-10-C (see Figure 4.9). CS is measured between clean and corrupted features, one severity level at a time (average CS across layers is reported in Figure 4.9). Level 1 represents zero corruption, i.e., clean image and level 6 represents maximum corruption. Compared to the baseline,

both LP-ReLU$_1$ and LP-ReLU$_2$ maintain higher CS across severity levels. To put it in simple words, even in the presence of input corruption, LP-ReLU coupled with DCT augmentation produces features similar to those produced for corresponding clean images.

In addition to different severity levels, we also measure shift at four increasing network depth levels as well. Figure 4.9 shows that our LP-ReLU based networks maintain the lowest average shift at all levels. It is also evident from Figure 4.9 that the shift is benign in the initial layers for ReLU network and becomes malignant in the deeper layers to a point where misclassification is inevitable.

As illustrated in the qualitative example in Figure 4.9, features from a noisy digit ('2') remain within the true class perimeter in LP-ReLU networks even with increasing magnitude of corruption. This is because of the compactness enforced by the low-pass filtering property inside LP-ReLU. However, sparsity in ReLU networks means features start drifting away even with negligible corruption resulting in misclassification.

4.3.6 Training Time

Unlike ReLU, both variants of the proposed LP-ReLU execute conditional statements during training and hence require marginally greater time per training epoch. As can be seen from Table 4.6, ReLU is the fastest with 50.3 s/epoch. LP-ReLU$_1$ turns out to be slightly faster than LP-ReLU$_2$ because of lesser parameters. However, we found both LP-ReLU networks to converge to training with fewer epochs compared to ReLU networks. For example, ReLU requires $[180, 190]$ training epochs to reach optimal performance, whereas both LP-ReLU variants perform optimally at 150 epochs. We observed that the overall training time for ReLU and LP-ReLU variants are quite similar. We argue that the sparsely represented data in ReLU requires additional training iterations to reach the global minima (with respect to the cost function). On the other hand, LP-ReLU's data representation is compact by nature and, therefore, requires fewer iterations to reach the global minima.

Table 4.6: Comparison of training time with different AFs (on CIFAR-10). Average time per epoch is reported in seconds. Number of epochs required to reach convergence is provided in a range. Training protocol is same as described in Section 4.3.2.

Network	Time/epoch (s)	No. of epochs
ReLU	50.3	$[180, 190]$
Leaky-ReLU	55.0	$[185, 190]$
P-ReLU	54.5	$[180, 190]$
C-ReLU	55.9	$[140, 145]$
LP-ReLU$_1$	58.1	$[145, 150]$
LP-ReLU$_2$	59.6	$[140, 145]$

4.4 Visualising Features and Decision Space

Visualising features and learned decision boundaries could deepen our understanding of CNN's robustness issues. Unfortunately, high dimensional features are hard to visualise. Dimensionality reduction algorithms based on statistical properties of features, e.g., PCA [135] and t-SNE [136], are often used in such cases. However, they do not provide an absolute feature space and the true decision space remains unknown. MLPs could be used as binary classifiers for a visual explanation, but the notion of data corruption is hard to replicate in such setups. In this chapter, rather than using statistical properties from a high dimensional feature vector extracted from a CNN's FC layer, we use a two dimensional FC layer to reduce the dimensions during training. This way, the network itself provides us with a lower-dimensional feature set (without using PCA or t-SNE) that is absolute in nature. In the following section, we provide details on this feature extraction process and how we can approximate the learned decision boundary.

To train a CNN on the MNIST dataset, first, we design a simple three-convolution layer CNN that achieves $\approx 99\%$ accuracy on MNIST (see Figure 4.10(a)). Keeping visualisation in mind, we augment another FC layer (FC_2) right before the already existing FC_1 layer with only two neurons (see Figure 4.10(b)). Such a low dimensional FC layer is rarely used in complex tasks. However, our FC_2 augmented network maintains the same classification accuracy ($\approx 99\%$) on the MNIST test set which confirms the network does not overfit or underfit because of the additional layer. This

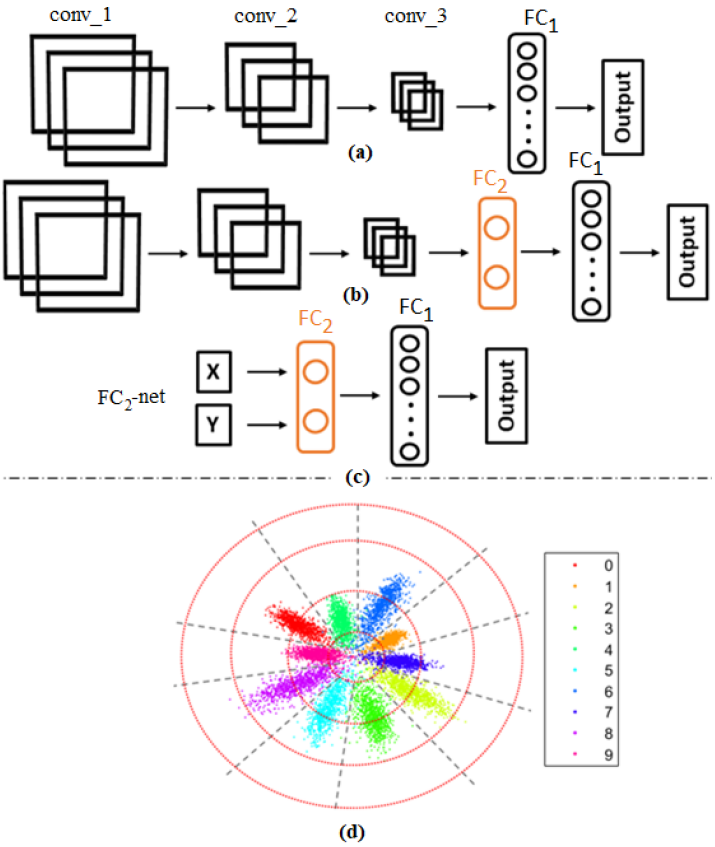

Figure 4.10: (a)-(c) To visualise a CNN's learned decision space on MNIST, a FC layer with two neurons is added as the penultimate FC layer (we call it FC_2). Once this network is trained, all the layers upto FC_2 are pruned and a tiny three-layer network is formed (FC_2-net). **(d)** Following Algorithm 4.1, (X, Y) values (generated image features) are systematically fed to FC_2-net and a decision space map is created from the class response.

can be attributed to MNIST's low complexity as a dataset. We argue that it is still a high dimensional dataset (28×28 images) and it presents an opportunity to understand the decision landscape learned by CNNs trained on any image dataset. Plots in Figures 4.1, 4.9, and 4.10 are produced from the FC_2 layer features from their respective CNNs (ReLU is used in one network (ReLU-net) and LP-ReLU variants in two other networks). No conventional dimensionality reduction is used as the features themselves are only two dimensional.

Algorithm 4.1 : Decision Space Mapping
Input: Generated Features (X_i, Y_i)
Output: Learned Decision Landscape.

Init $Polar(r, \theta)$ and $i = 1$
where, $r \in [1 : 1 : N], \theta \in [0 : .01 : 2\pi]$
for $r = 1$ to N **do**
  for $\theta = 0$ to 2π **do**
    $(X_i, Y_i) = Cartesian(Polar(r, \theta))$
    $[class_i scores_i] = classify(2FCnet, (X_i, Y_i))$
    if $scores_i > 50\%$ **then**
      $Flag_i = 1$
    end if
    $i = i + 1$
  end for
end for
Connect $\forall(X_i)\exists(Y_i)$ $(Flag == 1)$

4.4.1 Decision Space Mapping

From the trained ReLU-net, we prune all the layers up to the FC_2 layer and call this tiny network FC$_2$-net (see Figure 4.10(c)). FC$_2$-net gives us the opportunity to directly mimic absolute image (digits from 0 to 9) features with simple two dimensional values (X, Y).

To map the decision boundary in the feature space, the following steps are followed:

1. First, we plot FC_2 features from ReLU-net, compute the origin, and fix an initial point in the Polar co-ordinate $P(r, \theta)$ with $r = 1$ and $\theta = 0$.

2. Next, we keep r unchanged but increase θ by 0.01 up to 2π and complete a full circle traversal. We start over this process by incrementing r by 1.

3. While traversing, we move back to Cartesian co-ordinate and feed the values (FC_2 features) to the ReLU-net. We keep a note of the points where classification score trips over from one class to another adjacent class ($< 50\%$).

4. Intriguingly, the class label does not alter regardless of increasing r value as long as θ stays fixed. This means the decision boundaries are linear and extend to infinity.

5. Interestingly, all inter-class trip over points from Step 2 fall in a straight line (with negligible deviation). This complements the finding in the last step, and joining these trip over points reveals the true decision boundary (see Figure 4.10 for a visual representation).

The entire decision space mapping process is summarised in Algorithm 4.1, and visually depicted in Figure 4.10(d).

While experimenting, we found that classification scores become increasingly extreme with features moving further away from the manifold. This explains why corrupted instances often induce unusually high misclassification score in ReLU-based networks as corrupted features often drift away from true class. LP-ReLU effectively reduces the open space risk by enforcing compactness and limiting sensitivity to corruptions.

4.5 Conclusion

In this chapter, we investigated why unbounded AFs such as ReLU is a weak link in modern CNNs, especially against corruptions and perturbations. To this end, we analysed common corruptions from the frequency domain and proposed an AF with built-in low-pass filtering called LP-ReLU. LP-ReLU's design correlates with the human visual system and complies with the classic signal processing fix for HFc. LP-ReLU complements DCT augmentation and improves CNNs' robustness to corrupted input.

While the proposed AF improves distortion robustness by suppressing noise, aliasing in conventional DS layers such as max-pool can create additional forms of vulnerabilities in CNNs. For example, signal aliasing leaves CNNs vulnerable to benign image transformations such as shift and scale. In the next chapter, we propose an anti-aliasing AF and DS method to ensure further robustness against such vulnerabilities.

Chapter 5

Robustness to Shift and Perturbation with Anti-Aliasing CNN

In the previous chapter, we discussed the role of AFs in suppressing spurious signal and proposed LP-ReLU for improved robustness against corruption. In this chapter, we first discuss why conventional down-sampling (DS) suffers from signal aliasing and how it impacts performance on transformed images, such as shift and common corruptions, discussed in the previous chapters. To this end, we have proposed a novel DS method to mitigate the problem. However, fixing only the DS does not suffice since aliasing reappears through conventional AFs. As a solution, we propose a novel anti-aliasing AF in this chapter. Our proposed methods not only provide robustness against image transformation but also distortion. This work is currently under review in the IEEE Transactions on Multimedia.

The rest of this chapter is organised as follows. Section 5.1 provides an introduction. In Section 5.2, we introduce and formulate the proposed DS method and AF. In Section 5.3, we outline the datasets used for training and testing and then provide a comparative performance evaluation. Section 5.4 provides an ablation study for different design choices of our proposed network. Finally, Section 5.5 concludes this chapter.

5.1 Introduction

CNNs have achieved SOTA results in a wide range of vision tasks, including large-scale image classification [1]. To understand the extent of CNNs' robustness, researchers

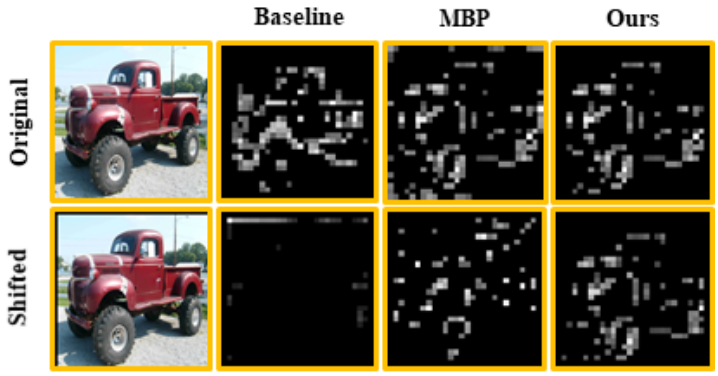

Figure 5.1: Features from an original image (top), and its shifted variant (bottom). The baseline (ResNet-101) – without any low-pass filtering, and MaxBlurPool (MBP) [59] – with monotonic blurring, both experience signal degeneration due to aliasing. Our method retains bulk of the expected signal. Here, 28×28 features are taken from the res_3 block.

have started looking beyond performance on i.i.d. (independent and identically distributed) datasets only. Recent studies reveal that CNNs are vulnerable to subtle changes in input, e.g., simple one pixel shift [60, 59, 147], quasi-imperceptible noise [43, 29, 31, 55, 33], blur [148, 38], and adversarial attacks [149, 150, 151]. This is concerning for real-world multimedia applications, where the i.i.d. assumption does not always hold. For instance, an autonomous car should not flip predictions for the same object between consecutive video frames due to marginal spatial shift or image noise [152, 153]. Recently, aliasing – a process whereby signal degenerates while DS before adequate blurring, has been identified as one of the main reasons behind CNNs' lack of robustness, especially against small image transformations such as shift [60, 59].

In CNNs, aliasing occurs when DS operations do not satisfy the Nyquist sampling rate [144, 60]. The theorem states that if the DS factor is not at least double of the maximum signal frequency, the output will be corrupted. In the context of CNNs, the problem aggravates with increasing network depth, leading to loss of the feature structure, and in effect, loss of shift-invariance. Blurring before subsampling, a well-known signal processing fix, could be effective here [59], but naively using the same blur kernel across the network, as used in MBP [59] or using separate ones for each spatial location [65], does not yield satisfactory performance. In Figure 5.1, we compare feature maps from three different networks, and find that the baseline (ResNet-101), as well as MBP suffer from signal degeneration.

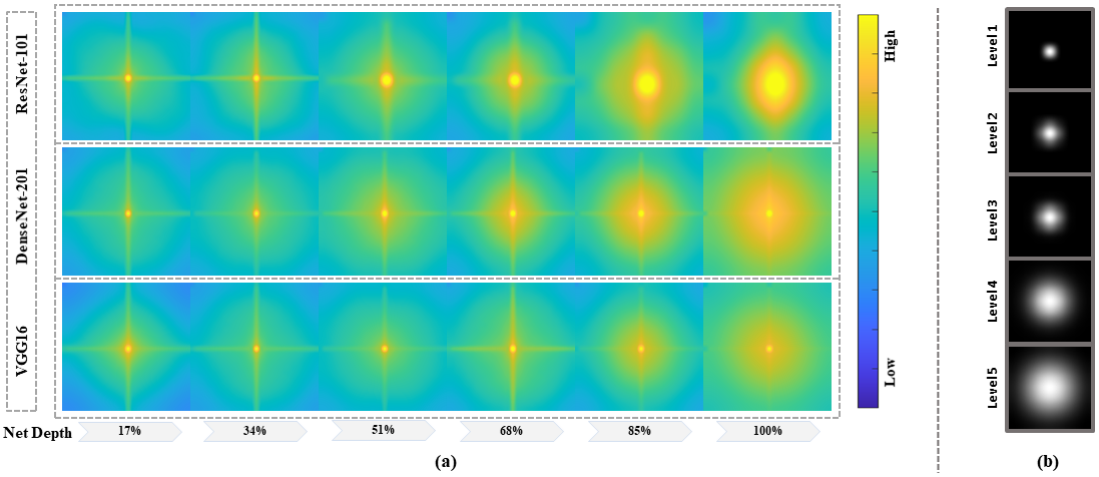

Figure 5.2: (a) Fourier energy at different network depths (lower frequencies are in the centre). Heat maps are generated using feature maps, at different network depths, belonging to the ImageNet validation set. For all three networks, deeper feature maps have greater high frequency energy, suggesting the importance of depth adaptive blurring. **(b)** Our proposed method ensures better anti-aliasing by learning stronger blur filters for deeper layers, in this instance, using a ResNet-101 backbone.

In this chapter, we propose a Depth Adaptive Blur-pool (DAB-pool) module and recommend replacing common DS methods with it (details in Section 5.2.1). DAB is inspired by the Fourier analysis, where we find that the feature maps – belonging to deeper CNN layers become increasingly higher frequency in nature. This can be visualised in Figure 5.2(a) which plots the Fourier energy heat maps at different network depths. Lower frequencies from the feature maps are in the centre of each square block. This heuristic motivates us to let the network learn progressively stronger blur kernels, and Figure 5.2(b) shows such a set of learned kernels.

Recently, Azulay et al. [60] have pointed out that aliasing persists even after blurring, as AFs often allow high frequency noise to alias back into the signal, and yet, this remains largely unaddressed. As a solution, in this chapter, we propose a novel AF that not only serves as an effective non-linearity unit, but also acts as a secondary low-pass filter (see Figure 5.3). Similar to DAB-pool, this can also be readily used in existing CNNs for further performance gain.

To assess shift-invariance, we follow benchmark evaluation protocols in [60, 66, 59]. We also test our defence against recent (translation-based) adversarial attacks [150],

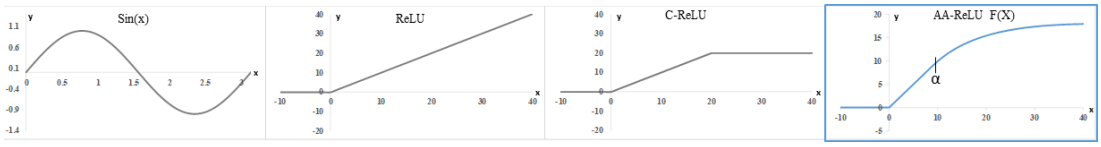

Figure 5.3: Our proposed AF (AA-ReLU) gates between ReLU and C-ReLU by exploiting the roll-off phase in $\sin x$.

covering both black-box and white-box settings. Although primarily pursuing shift-invariance, we assess performance generalisation on a variety of corruptions and spatial perturbations on ImageNet-C and ImageNet-P respectively [31]. In summary, we make the following contributions:

- Inspired by the sampling theorem [144], we investigate ways of instilling anti-aliasing properties in CNNs. Since CNNs host a variety of layers, simply inserting a low-pass filter does not suffice. To this end, we propose to redefine the conventional DS and AFs as potential anti-aliasing units.
- We analyse the spectral properties of deep features and propose to use DAB-pool for DS as the primary anti-aliasing unit.
- We propose a novel AF with a built-in low-pass filter as a secondary anti-aliasing unit.
- We use the proposed DS method and AF in VGG16, ResNet-101, and DenseNet-121 to demonstrate their effectiveness across architectures.
- In addition to shift and shift-based perturbations, we also evaluate on adversarial attacks, data corruptions, and image transformations. Our method consistently produces results better than existing methods.

5.2 Proposed Method

In this section, we first define the problem setup. Then we describe DAB-pool and how to integrate it into modern CNNs, followed by the details of our anti-aliasing AF.

Problem Formulation. Let a CNN classifier be $\mathcal{G}$ – a function that takes X as the input and maps it to an output class $\mathcal{O}_c$. Considering X has a spatial resolution of $H \times W \times C$, $\mathcal{G}$ can be expressed as $\mathcal{G}(X^{H\times W\times C}) \rightarrow \mathcal{O}_c$. For translation invariance to hold, $\mathcal{G}$ should predict the same class label for X, and its translated variant as Equation 5.1 shows.

$$\mathcal{G}(X^{H\times W\times C}) = \mathcal{G}(X^{H\times W\times C} + \Delta T) = \mathcal{O}_c \tag{5.1}$$

where, ΔT represents a marginal input translation.

5.2.1 Depth Adaptive Blur-pool (DAB-pool)

For effective anti-aliasing, a blur kernel has to obey the sampling theorem, i.e., sufficient high frequencies should be removed before sub-sampling. A signal, containing mostly high frequencies, needs stronger blurring, as opposed to a low frequency signal. This is because a weaker than needed filter might leave frequencies beyond the limit imposed by the Nyquist theorem. On the other hand, using over-aggressive blurring might remove the meaningful part of the signal.

Our spectral analysis, according to Figure 5.2(a), reveals that the feature maps in deeper layers have greater energy quotient for high frequencies. A similar trend is observed in all three networks, which reinforces the consistency of this observation. It also suggests that a 'one size fits all' approach, i.e., using the same blur kernel for all layers would be sub-optimal, hence, we propose DAB-pool.

To put these in formal notation, let the feature map for an input X at layer L be x^L, a Gaussian blur kernel at depth level D be G^σ_D, and the output of the convolutional DAB be Y_{DAB}. Here, σ is the standard deviation of the kernel, and a higher σ denotes stronger blurring. Depth level denotes the number of DS layers, e.g., a network with four DS layers has four depth levels, i.e., $D \in \{1, 2, 3, 4\}$. Also let the $(L+1)^{th}$ layer be a DS layer and hence we aim to blur x^L. Now, we can define our convolutional DAB operation as Equation 5.2:

$$Y_{DAB}(i, j) = \sum_{p=1}^{m-1} \sum_{q=1}^{n-1} x^L_{i+p,j+q} \cdot (G^\sigma_D)_{p,q} \tag{5.2}$$

where $Y_{DAB}(i,j)$ is the output at location (i,j) and the Gaussian kernel resolution is $m \times n$.

Rather than using pre-fixed filters [59], we let the network learn them by casting σ as a learnable parameter. To ensure progressively stronger blurring with increasing depth D, we enforce the following learning constraints: $G_1^\sigma < G_2^\sigma < G_3^\sigma < G_4^\sigma \ldots < G_M^\sigma$, where $M = max(D)$. It is worth mentioning that Equation 5.2 closely resembles conventional convolution operation. The only difference is in the filter. G_D^σ in Equation 5.2 is the filter that performs a blur operation and the only learnable parameter is σ. However, conventional convolution involves a fully parameterised filter that learns to recognise object level features.

Replacing DS Layers. We replace max-pool, strided convolution (S-conv), and avg-pool as follows:

$$\begin{aligned} \text{Max-pool}_{k,s} &\rightarrow \text{Subsample}_{k,s} \circ \text{DAB}_{k,s} \circ \text{DenseMax}_{k,s=1} \\ &= \text{DAB-pool}_{k,s} \circ \text{DenseMax}_{k,s=1} \end{aligned} \tag{5.3}$$

$$\begin{aligned} \text{Conv}_{k,s>1} &\rightarrow \text{Subsample}_{k,s} \circ \text{DAB}_{k,s} \circ \text{Conv}_{k,s=1} \\ &= \text{DAB-pool}_{k,s} \circ \text{Conv}_{k,s=1} \end{aligned} \tag{5.4}$$

$$\begin{aligned} \text{AvgPool}_{k,s} &\rightarrow \text{Subsample}_{k,s} \circ \text{DAB}_{k,s} \\ &= \text{DAB-pool}_{k,s} \end{aligned} \tag{5.5}$$

Here, k and s refer to kernel dimensions and stride respectively. DenseMax is simply stride one max-pooling, found effective in improving task performance [59]. Each sub-sampling operation, in Equations 5.3, 5.4, and 5.5, can be easily combined with DAB as a single operation for efficiency, and hence, jointly represented as DAB-pool in the equations.

Gradient Backpropagation. In a CNN, partial derivatives of the final cost function are propagated backwards and the learnable parameters are updated accordingly. Here, we demonstrate backpropagation through our proposed DAB layer.

As shown in Equation 5.2, x^L is the input, G_D^σ is the kernel, and Y_{DAB} is the output. Assume dy is the error that has already backpropagated up to Y_{DAB}. From Equation 5.2, we can further derive the following as Equation 5.6.

$$\begin{aligned} Y_{DAB}(i,j) &= \sum_{p=1}^{m-1}\sum_{q=1}^{n-1} x^L_{i+p,j+q} \cdot (G_D^\sigma)_{p,q} \\ \text{Where, } (G_D^\sigma)_{p,q} &= \frac{H_g\,(p,q)}{\sum_p \sum_q H_g} \\ \text{and } H_g\,(p,q) &= e^{\frac{-\left(p^2+q^2\right)}{2\sigma^2}} \end{aligned} \tag{5.6}$$

As shown later in Section IV, we use a fixed 3×3 blur kernel for all DAB layers and hence, p and q always have the same size, i.e., $m = 3$ and $n = 3$. Standard deviation σ is the only learnable parameter here that gets updated during training.

During backpropagation, partial derivatives can be calculated by applying the chain rule [154]. However, replicating the chain rule via convolution is a more efficient way [155]. Partial derivatives of the weight matrix G_D^σ are calculated by convolving the input x^L with the error matrix dy [155] and represented as dG_D^σ. Equation 5.7 shows the calculation of dG_D^σ. Similarly, error dy is dispersed backwards by convolving it with G_D^σ [155] as shown in Equation 5.8.

$$dG_D^\sigma(p,q) = \sum_{p=1}^{m-1}\sum_{q=1}^{n-1} x^L_{i+p,j+q} \cdot (dy)_{i,j} \tag{5.7}$$

$$dx^L(i+p,j+q) = \sum_{p=1}^{m-1}\sum_{q=1}^{n-1} (dy)_{i,j} \cdot (G_D^\sigma)_{p,q} \tag{5.8}$$

5.2.2 Anti-Aliasing ReLU (AA-ReLU)

In common signal processing tasks, low pass filtering is theoretically sufficient to avoid aliasing. For example, an analogue to digital sound converter inside a voice recorder can guarantee alias-free output, as long as an optimal low-pass filter is used. In contrast, blurring before sub-sampling cannot provide such guarantee in CNNs due to non-linearities in the form of AFs [60]. We argue that an AF can largely address this

by acting as a secondary AA unit that complements DAB-pool by resisting noise. Such an AF should: allow the signal to go through and have a low-pass filtering or function roll-off mechanism, as well as continuity described as below:

I. **Early feature sparsity:** [69] Small positive features are mostly true signal [137] and hence should be allowed to propagate forward – similar to what C-ReLU offers prior to the clipping point.

II. **Smooth roll-off:** This is required to complement DAB-pool by high frequency noise suppression. Contrary to a hard-clipping used in C-ReLU, a smooth roll-off is a better choice as the buffer between the signal and noise is unknown. A hard early clipping might affect the original signal, as well as deny early feature sparsity.

III. **Function continuity:** For any AF to be effective, it has to be continuous so that there is no point in the function without a derivative.

Considering these, we propose an AF in Equation 5.9 and subsequently describe how it satisfies all three properties.

$$F(x) = \begin{cases} 0, & x \in (-\infty, 0] \\ x, & x \in (0, \alpha) \\ f_1^x = \alpha \sin\left(\ln \frac{x}{\alpha}\right) + \alpha, & x \in \left[\alpha, \alpha \exp\left(\frac{\pi}{2}\right)\right] \\ f_2^x = max(f_1^x) = 2\alpha, & x \in \left(\alpha \exp\left(\frac{\pi}{2}\right), +\infty\right) \end{cases} \tag{5.9}$$

where α is a learnable parameter representing the roll-off start point, as well as the amplitude, frequency, and phase shift parameter of our sinusoidal roll-off function f_1^x.

To ensure **early feature sparsity (I)**, the proposed AA-ReLU, as in Equation 5.9, is identical to ReLU for $x \in [-\infty, \alpha)$, however, for $x \geq \alpha$, the signal starts to **smoothly roll-off (II)**. f_1^x dictates the roll-off severity and span, whereas $f_2^x = 2\alpha$ denotes the capping value as shown in Equation 5.9. For further analysis, let f_1^x in Equation 5.9 be represented as Equation 5.10:

$$\begin{aligned} f_1^x &= \alpha \sin\left(\ln \frac{x}{\alpha}\right) + \alpha \\ &= \alpha \sin(g(x)) + \alpha, \quad \text{here, } g(x) = \ln \frac{x}{\alpha} \end{aligned} \tag{5.10}$$

Unlike the linear growth in ReLU, we aim to roll-off after a certain α. To ensure that happens, f_1^x starts by marginally suppressing the output for inputs slightly greater than α, i.e., $x > \alpha$. Harder suppression takes effect for sufficiently large x, i.e., $x \gg \alpha$. Contrary to the oscillating property of a typical $\sin$ function, $F(x)$, for $x \gg \alpha$, should plateau, i.e., it should reach and stay at the function maximum such that $F(x) = max(F(x))\{\forall x | x \gg \alpha\}$. A $\sin x$ function in the interval of $(0, \frac{\pi}{2}]$ has these desirable roll-off properties and our proposed AF in Equation 5.9 exploits it.

It is worth noting that the input to f_1^x can be large real numbers, i.e., $x \in \mathbb{R}_{>0}$. Therefore, plugging x directly in a $\sin x$ function would be impractical, as we do not want our sinusoidal output to oscillate periodically. Rather, we want a substantially lower $\sin$ frequency f, so that we have access to a wider and smoother roll-off region. This is why we use a natural logarithmic function $g(x)$ in Equation 5.10, and in effect, inside our proposed AF in Equation 5.9.

In addition to denoting the roll-off start point, α also increases the amplitude of f_1^x (since $\alpha \in \mathbb{Z}_{>0}$), which allows early sparsity in $F(x)$, as Figure 4.2 shows. α also vertically shifts f_1^x, and further modulates the frequency (by dividing x inside the $\log$ function) such that $F(x)$ remains continuous at each point.

To show AA-ReLU is **continuous (III)**, we prove that $F(x)$ in Equation 5.9 is continuous everywhere as follows:

- $F(x) = 0$ and $F(x) = x$ are continuous since constant and identity functions are continuous everywhere [156]. At the joining point, i.e., for $x = 0$, both function pieces produce 0 and hence they are jointly continuous as well.
- f_1^x is continuous in the given domain as it is a function of $\sin$ [156]. At the joining point with $F(x) = x$, i.e., for $x = \alpha$, both functions produce the same output since $f_1^x = \alpha \sin(\ln \frac{\alpha}{\alpha}) + \alpha = \alpha$ and $F(x) = x$ is an identity function.
- f_2^x is another (constant) continuous function and at the joining point with f_1^x, i.e., for $x = \alpha \exp\left(\frac{\pi}{2}\right)$, $f_1^x = \alpha \sin(\ln \frac{\alpha \exp\left(\frac{\pi}{2}\right)}{\alpha}) + \alpha = 2\alpha = f_2^x$, and hence, they are also jointly continuous.

A case study is presented below for further understanding.

Case Study. Let a 1D sample signal be x = [-9, 3, 8.9, 9, 10, 43.3, 81]. Assuming x as the input to Equation 5.9, and considering $\alpha = 9$, $F(x)$ is evaluated as follows:

- $F(-9)$, $F(3)$, and $F(8.9)$ output $0, 3$, and 8.9 respectively. By definition, these outputs are identical to ReLU as $x < \alpha$.
- $F(9) = f_1^9 = 9 \cdot \sin\left(\ln \frac{9}{9}\right) + 9 = 9 \cdot 0 + 9 = 9$ (note $x == \alpha$). From here on, the roll-off function smoothly takes over from ReLU. Notice that using the same α as the amplitude, frequency, and phase shift parameter ensures function continuity.
- $F(10) = f_1^{10} = 9.95$. x has been slightly suppressed since $x > \alpha$.
- $F(43.3) = f_1^{43.3} = 9 \cdot \sin\left(\ln \frac{43.30}{9}\right) + 9 = 9 \cdot \sin\left(\ln e^{1.57}\right) + 9 = 9 \cdot \sin \frac{\pi}{2} + 9 = 18$. x has been heavily suppressed since $x >> \alpha$.

  By calculating the second order derivative of f_1^x, we can calculate the function maximum. For this example, solving Equation 5.11 yields $max(F(x)) = 18$.

$$f_1^{x''} = -\frac{\alpha\left(\sin\left(\ln\left(\frac{x}{\alpha}\right)\right) + \cos\left(\ln\left(\frac{x}{\alpha}\right)\right)\right)}{x^2} \tag{5.11}$$

- $F(81) = f_2^{81} = max(f_1^x) = max(F(x)) = 18$. As shown in Equation 5.11, $x = 43.3$ induces F(x) to reach its maximum value, and therefore, according to Equation 5.9, $F(81) = F(x > 43.3) = 18$.

Gradient. Equation 5.12 shows the derivatives of $F(x)$.

$$\frac{d}{dx}F(x) = \begin{cases} 0, & x \in (-\infty, 0], \\ 1, & x \in (0, \alpha), \\ \frac{\alpha}{x}\cos\left(\ln\frac{x}{\alpha}\right), & x \in \left[\alpha, \alpha\exp\left(\frac{\pi}{2}\right)\right] \\ 0, & x \in \left(\alpha\exp\left(\frac{\pi}{2}\right), +\infty\right) \end{cases} \tag{5.12}$$

Here, beyond α, saturated points exist, but an optimally learned α accounts for this issue.

Backbone	Methods	Clean Accuracy		Shift Consistency (higher is better)					
				Diagonal Shift		Rescale Shift		Double Rescaling	
		Abs	Δ	Abs	Δ	Abs	Δ	Abs	Δ
VGG16	Baseline (VGG16)	71.66	-	88.52	-	83.89	-	85.04	-
	MBP [59] (ICML '19)	72.29	+0.63	90.11	+1.59	83.99	+0.10	86.98	+1.94
	SABP [65] (BMVC '20)	72.95	+1.29	90.36	+1.84	84.20	+0.31	87.10	+2.06
	BNS [48] (WACV '21)	71.75	+0.09	90.36	+1.84	84.06	+0.17	85.24	+0.20
	WaveCNet [67] (CVPR '20)	71.96	+0.30	90.42	+1.90	85.35	+1.46	86.05	+1.01
	F-Conv [66] (CVPR '20)	70.65	-1.01	90.01	+1.49	81.15	-2.74	84.78	+0.74
	DABP + AA-ReLU (ours)	**73.91**	**+2.25**	**91.85**	**+3.33**	**87.47**	**+3.58**	**90.25**	**+5.21**
ResNet-101	Baseline (ResNet-101)	77.37	-	89.95	-	80.20	-	83.54	-
	MBP ([59])	77.40	+0.03	91.31	+1.36	86.10	+5.90	83.97	+0.43
	SABP ([65])	79.32	+1.95	92.19	+2.24	85.94	+5.74	83.81	+0.27
	BNS [48]	77.25	-0.12	90.05	+0.10	84.11	+3.91	82.89	-0.65
	WaveCNet [67]	78.21	+0.84	90.89	+0.94	85.95	+5.75	83.15	-0.39
	F-Conv ([66])	78.31	+0.94	90.05	+0.10	84.66	+4.46	82.23	-1.31
	DABP + AA-ReLU (ours)	**81.45**	**+4.08**	**94.11**	**+4.16**	**91.36**	**+11.16**	**86.45**	**+2.91**
DenseNet-121	Baseline (DenseNet-121)	74.44	-	88.81	-	82.35	-	84.25	-
	MBP ([59])	75.03	+0.59	90.39	+1.58	84.21	+1.86	85.74	+1.49
	SABP ([65])	75.36	+0.92	88.85	+0.04	83.55	+1.20	85.77	+1.52
	BNS [48]	74.65	+0.21	88.96	+0.15	82.46	+0.11	85.09	+0.16
	WaveCNet [67]	74.75	+0.31	89.02	+0.21	83.50	+1.15	85.16	+0.91
	F-Conv ([66])	74.09	-0.35	88.84	+0.03	80.88	-1.47	84.55	+0.03
	DABP + AA-ReLU (ours)	**77.44**	**+3.00**	**92.85**	**+4.04**	**88.25**	**+5.90**	**88.08**	**+3.83**

Table 5.1: Top-1 clean accuracy (%) and shift *consistency* (%) on ImageNet. Our method outperforms others under different types of shift. Here, Δ represents the improvements over baseline and the best result is highlighted in bold.

5.3 Experiments and Results

In this section, we first outline the training details and hyper-parameter settings. Later, for evaluation, we test using (1) three shift-based perturbations, (2) three recent translation-based adversarial attacks [150, 151], and (3) a range of corruptions and perturbations. In each set-ups, we refer to the following works for comparison: MBP [59], Spatially Adaptive Blur Pooling (SABP) [65], Wavelet Integrated CNN (WaveCNet) [67], Batch Normalisation Statistics (BNS) [48], and Full-Convolution (F-Conv) [66].

5.3.1 Implementation and Training Details

We train and test with three baseline networks – VGG16 [25], ResNet-101 [26], and DenseNet-121 [28], following the original network architectures presented in respective

papers. We use SGD with $Momentum = 0.9$, and train for 100 epochs, while using an initial learning rate $L_r = 0.1$. L_r is dropped by a factor of 0.2 after 30, 60, and 90 epochs and a batch-size of 128 is maintained. Zero-centre input normalisation is used for all experiments, with an L2 Regularisation factor of 0.0005.

For our learnable parameter σ, i.e., the standard deviation of Gaussian blur filters, we initialise based on their depth and set $\sigma_D = \frac{D}{2}$. Take for example ResNet-101: it has five DS layers and hence, has five depth levels, yielding the following initialisations: $\sigma_1 = 0.5$, $\sigma_2 = 1$, $\sigma_3 = 1.5$, $\sigma_4 = 2$, and $\sigma_5 = 2.5$. For all the depth levels, we use 3×3 blur kernels (see ablation study in Section 5.4).

To initialise the other learnable parameter α, placed inside AA-ReLU, we start off with 6 based on heuristic [69].

5.3.2 Shift Invariance on ImageNet

ImageNet has 1,000 classes with 1.2 million training and 50,000 test images. We follow three shift evaluation protocols from [60, 59]. A network $\mathcal{G}$'s *consistency* (evaluation metric) is the measure of stable Top-1 predictions for an input X, and a translated $X_{\Delta T}$, formally put in Equation 5.13.

$$\textit{consistency} = \mathbb{E}_X[\![\{\mathcal{G}(X) == \mathcal{G}(X_{\Delta T})\}]\!] \tag{5.13}$$

where $[\![\cdot]\!]$ denotes Iverson Bracket that outputs 1 if the proposition inside is true and 0 otherwise. A summary of all three shift perturbation protocols is provided below:

Diagonal Shift. Here, each X from the validation set is diagonally shifted by N pixels, where $N \in [1, 64]$, and N is random.

Rescale Shift. Here, each X is first down-scaled to a 100×100 image, and later, embedded in a 224×224 canvas, filling rest of the blank space with black pixels. Finally, the resultant image is diagonally shifted by a single pixel to obtain the perturbed variant, i.e., $X_{\Delta T}$.

Double Rescaling. Here, X is first down-scaled and embedded, similar to Rescale Shift. However, rather than shifting the rescaled embedding itself, it is spatially altered

by ± 1 pixel to obtain $X_{\Delta T}$ (e.g., from an embedding of 100×100 to 101×101).

Results. Compared to the baselines VGG16, ResNet-101 and DenseNet-121 (pictorial depiction in Figure 2.6), DABP+AA-ReLU improves shift-invariance across all three deep networks in Table 5.1 despite substantial architectural difference. For instance, VGG16 has four intermediate max-pool layers as opposed to three average pooling layers in DenseNet-121. On the other hand, ResNet-101 and DenseNet-121 use residual block and depth concatenation, respectively, whereas VGG16's architecture is much simpler with sequential input and output layers. Interestingly, our method improves the clean Top-1 accuracy as well exceeding the baseline by $+2.25\%$ in VGG16, by $+4.08\%$ in ResNet-101, and by $+3\%$ in DenseNet-121. Compared to a fixed blur kernel in MBP [59] and a spatially adaptive blur kernel in SABP [65], DABP+AA-ReLU shows better consistency in each of the evaluation protocols in Table 5.1. Unlike other methods, it addresses the role of AFs as well, which complements DAB-pool in achieving strong shift-invariance.

5.3.3 Robustness Against Adversarial Attacks

By definition, an image X' is an adversarial variant of some valid image X, if both appear visually similar to a human, and yet, a network misclassifies X'. Although visual perception is subjective and similarity between images is hard to quantify, the relevant literature presumes that X' is adversarial if and only if $\|X - X'\|_p \leq \varepsilon$, where $p \in [1, \infty)$ and ε is small. Conventional first-order attacks, e.g., Fast Gradient Sign Method (FGSM [50]), operate in pixel space and generate adversaries from a bounded ℓ_p ball. As recently pointed out in [150], spatial adversarial vulnerability, e.g., due to translation, do not obey the bounded protocol. This is because a translated image $X_{\Delta T}$ can still appear visually similar, and yet, have a substantially large norm $\|X - X'\|_p$, compared to the norms allowed in ℓ_p. Hence, finding adversaries within a confined ℓ_p space is difficult with first-order methods. In this chapter, we adhere to the spatial attacks outlined in [150] and summarised below:

First Order Method (FO). FO is a white-box attack, meaning it requires access to

the target network architecture, and in effect, to its gradients. In ℓ_p based adversaries, each input pixel is cast as a learnable parameter in a bid to maximise loss for a particular class (to inflict misclassification). However, here, FO has to perturb the spatial tuning of an image X to generate a translated adversary X'. This is why input translation is parameterised in FO, and the loss for the correct class is maximised by iteratively updating these parameters. Following [150], the maximum perturbation, i.e., translation, allowed in this attack is 25 pixels, and in terms of implementation, it is 200 steps of projected gradient descent with step size 0.24.

Grid Search Attack (GSA). This is a black-box attack and requires only a few queries to the target model. GSA attacks the network by exhaustively translating X in each spatial direction until X becomes an adversary X' (if such an X' exists in the search grid). Here, maximum translation allowed in each direction is five pixels.

Worst-of-k. Here, to attack a target, X is translated randomly k times by keeping the translation parameters within the allowed limit, i.e., 25 pixels. An X', within this limit, inducing the worst classification performance is the one chosen as the adversary. For example, in worst-of-10 attack, it performs 10 random translations per X and feeds the corresponding X's to find the most effective adversary. It is a black-box attack as well.

Results. As Table 5.2 shows, our method exhibits better resistance against all three attacks in terms of Top-1 accuracy. Interestingly, DABP+AA-ReLU performs best against FO – the most effective attack in pixel-level ℓ_p bounded space, performing 1.5% over baseline (from here on, we refer to ResNet-101 as the baseline unless mentioned otherwise). This can be attributed to different loss landscapes in natural image transformations versus typical ℓ_p settings [150]. In ℓ_p loss landscape, the maxima are consistent and concentrate well in a locality, making it easier to find adversaries by taking small iterative steps towards it [51, 149]. However, the loss landscape of translation is non-convex and tend to have multiple scattered local maxima [150], meaning there is no guarantee that taking small steps would lead to translated adversaries, and hence, it is sub-optimal. GSA, on the other hand, turns out to be more challenging, and the baseline accuracy drops from 77.4% to 61.5%. DABP+AA-ReLU performs better than

Top-1 Accuracy (higher is better)				
		Adversarial Attack		
Methods	**Clean**	**First Order**	**Grid Search**	**Worst-of-10**
Baseline	77.37	74.78	61.50	73.39
MBP([59])	77.40	74.90	63.29	73.85
SABP([65])	79.32	74.96	63.55	73.44
BNS([48])	77.89	72.70	61.65	73.45
WaveCNet([67])	78.05	75.05	62.30	73.66
F-Conv([66])	78.31	73.85	62.60	72.88
DABP+AA-ReLU	**81.45**	**76.28**	**65.05**	**74.70**

Table 5.2: Top-1 classification accuracy (%) against different adversarial attacks (Baseline: ResNet-101).

MBP by 1.76%, SABP by 1.50%, BNS by 3.40%, WaveCNet by 2.75%, and F-Conv by 2.45% under GSA. A similar improvement trend is observed against Worst-of-10 attack as well.

5.3.4 Corruption and Perturbation Robustness

In this section, we test our corruption and perturbation stability on ImageNet-C and ImageNet-P respectively.

ImageNet-C. ImageNet-C [31] contains images from the ImageNet validation set with 17 different corruptions, broadly falling into the following four categories: noise, blur, weather, and digital. Each of the 17 corruptions has five severity levels and mean Corruption Error (mCE) is used as the evaluation metric.

Results. Table 5.3 shows that DABP+AA-ReLU achieves SOTA robustness against a variety of corruptions in ImageNet-C with an mCE 4.71% lower than the Baseline, 2.12% lower than MBP [59], 1.68% lower than SABP [65], 5.79% lower than BNS [48], 4.58% lower than WaveCNet, and 2.53% lower than F-Conv. Our method exhibits marked robustness in the noise category consisting of Gaussian, shot, impulse, and speckle noise, achieving an overall 7.83% lower mCE than the baseline. Compared to competing methods, our overall noise mCE is 3.28% lower than MBP [59] and 3.47% lower than SABP [65]. This is an expected by-product of blurred DS as most of the high frequency noise gets neutralised in this step. However, signal noise – much like

aliasing, can reappear in the non-linear AFs, which we believe is the case in the baseline and in [59, 65]. Our method is better suited to counter this challenge as AA-ReLU itself acts as a secondary low-pass filter. The baseline often performs well under level-1 corruption severity, but struggles beyond level 2 and 3 (see Figure 5.4).

Methods	ImageNet-C (Corruption Error CE, lower is better)																	
	Noise				Blur					Weather				Digital				
	Gauss	Shot	Impulse	Speckle	Gauss	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contr	Elastic	Pixel	Jpeg	mCE
Baseline	68.91	72.12	74.66	73.55	64.96	61.45	74.65	60.20	61.75	66.45	60.55	54.10	30.87	60.87	54.28	54.34	44.61	61.08
MBP ([59])	64.45	66.05	68.98	71.56	62.34	61.06	70.30	61.25	60.35	66.01	58.21	50.24	29.91	59.87	55.36	48.75	39.54	58.48
SABP ([65])	65.32	65.87	69.50	71.10	63.80	61.11	68.32	60.05	60.14	64.98	58.33	49.87	29.87	59.70	53.88	46.68	38.22	58.04
BNS ([48])	67.11	69.89	70.98	74.52	67.23	64.90	73.85	67.21	62.92	71.66	63.24	54.87	33.87	60.45	57.32	52.82	43.75	62.15
WaveCNet ([67])	66.32	69.15	70.08	74.14	66.14	63.36	71.98	65.05	62.44	69.70	61.34	54.01	31.22	60.14	56.99	52.32	41.66	60.94
F-Conv ([66])	66.98	65.21	69.37	72.88	62.77	60.81	72.66	60.32	61.45	66.02	60.24	52.36	29.56	59.86	54.44	46.87	39.35	58.89
DABP+AA-ReLU	**61.45**	**62.19**	**65.32**	**68.95**	**59.87**	**60.77**	**66.25**	**60.01**	**59.98**	**64.21**	**58.12**	**47.87**	**29.41**	**59.35**	**53.21**	**44.24**	**36.97**	**56.36**

Table 5.3: Corruption Error rate (%) on ImageNet-C [31] (Baseline: ResNet-101). Results in each category represent the average across all five severity levels in the dataset. The rightmost column (mCE) denotes the mean Corruption Error.

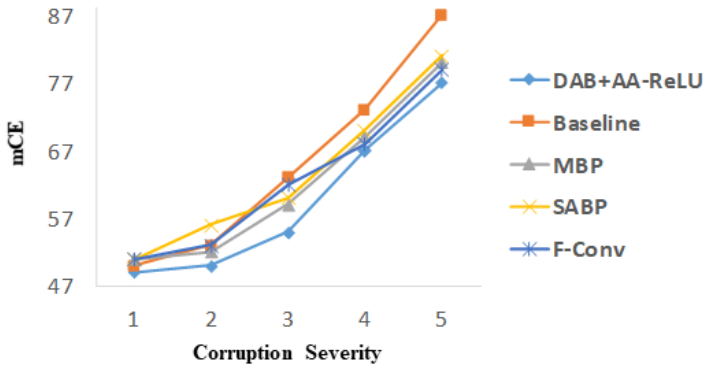

Figure 5.4: Mean Corruption Error (%) on increasing corruption severity levels in ImageNet-C (lower is better).

Our method shows better consistency, even with increasing corruptions.

ImageNet-P. ImageNet-P [31] introduces a range of perturbations in the validation set to test performance stability. In this chapter, we evaluate on four affine transformation-based perturbations involving translation, rotation, scaling, and tilt. In ImageNet-P, each test instance is a short video of about 30 frames with increasing perturbation severity. Flip Probability FP is used as the stability evaluation metric. A flip event occurs when two consecutive frames' predictions mismatch. To put it formally, let us denote k perturbation sequences (each with v number of frames) with $S = \left\{\left(x_1^{(i)}, x_2^{(i)}, \ldots, x_v^{(i)}\right)\right\}_{i=1}^{k}$. For a fixed i, i.e., perturbation type k, $x_1^{(i)}$ denotes the clean image (no perturbation) and $x_v^{(i)}$ denotes a frame with maximum (k type) perturbation. The Flip Probability (FP) of a deep classifier $\mathcal{G}$ on perturbation sequence S is:

$$FP_p^{\mathcal{G}} = \frac{1}{k(l-1)} \sum_{i=1}^{k} \sum_{j=2}^{v} 1\left(\mathcal{G}\left(x_j^{(i)}\right) \neq \mathcal{G}\left(x_{j-1}^{(i)}\right)\right) \tag{5.14}$$

Results. As Table 5.4 shows, our method shows better stability (0.87% lower mFP than the next best SABP), and interestingly, it generalises to perturbations other than translation as well, e.g., in rotation perturbation, our FP is 2.27% lower than the baseline, 1.34% lower than MBP, and 1.17% lower than SABP; gain is observed for tilt and scale as well.

ImageNet-P Flip Probability FP (lower is better)					
	Affine Transform Perturbation				
Methods	**Translate**	**Rotate**	**Tilt**	**Scale**	**mFP**
Baseline	4.15	6.39	3.96	10.85	6.34
MBP ([59])	3.70	5.46	3.37	8.41	5.24
SABP ([65])	3.42	5.29	3.32	8.06	5.02
BNS ([48])	4.46	5.98	4.20	9.35	6.00
WaveCNet ([67])	3.58	5.87	3.68	8.89	5.51
F-Conv ([66])	4.15	6.74	4.25	9.01	6.04
DABP+AA-ReLU	**2.26**	**4.12**	**3.22**	**6.98**	**4.15**

Table 5.4: Flip Probability rate (%) on transformation-based perturbations in ImageNet-P [31] (Baseline: ResNet-101). mFP denotes mean Flip Probability.

Network	Time/epoch (mins)	No. of epochs
ReLU+(S-conv, Max-pool) (baseline)	16.3	$[105, 115]$
C-ReLU+(S-conv, Max-pool)	18.2	$[85, 95]$
ReLU+SABP [65]	24.9	$[100, 110]$
AA-ReLU+DAB-pool (Ours)	22.6	$[90, 100]$

Table 5.5: Comparison of training time (on ImageNet). Average time required to complete an epoch is presented in minutes and the corresponding number of epochs for convergence is shown within a range. ResNet-101 is used as the baseline in all the networks.

5.3.5 Training Time

By definition, AA-ReLU executes more conditional statements than ReLU during training. Similarly, our proposed DAB-pool layer has an additional blurring operation compared to S-conv or max-pool with a single operational stage. Therefore, our CNN requires more training time compared to the baseline, i.e., vanilla ResNet-101 with ReLU + (S-conv and max-pool). As can be seen from Table 5.5, this baseline is the fastest with 16.3 mins/epoch. ResNet-101 with C-ReLU is slightly slower with 18.2 mins/epoch. Our network takes 22.6 mins/epoch which is 2.3 minutes lesser than the SABP network. It is worth noticing that our method requires $[90, 100]$ training epochs to converge, whereas SABP and the baseline require more epochs to converge. We argue that the unbounded nature of ReLU results in additional training iterations for convergence. In contrast, AA-ReLU has a bounded output and hence, the search space for the global minima is much smaller. This means our network requires fewer iterations of weight updating to reach this minima resulting in a reduced convergence time. Similar to AA-ReLU, C-ReLU also has faster convergence due to its bounded

Ablation Study (ImageNet)				
		Shift Consistency (higher is better)		
Methods	**Clean Accuracy**	**Diagonal shift**	**Rescale shift**	**Double Rescaling**
$DABP_{3*3}$+ReLU	79.95	91.19	88.41	84.15
$DABP_{5*5}$+ReLU	78.85	88.54	87.37	83.15
$DABP_{7*7}$+ReLU	77.03	85.21	83.87	81.29
$DABP_{3*3}$+C-ReLU	79.03	91.58	88.80	84.33
$DABP_{5*5}$+C-ReLU	78.25	88.75	87.54	83.36
$DABP_{7*7}$+C-ReLU	74.87	87.25	86.66	81.89
$DABP_{3*3}$+AA-ReLU	**81.45**	**94.11**	**91.36**	**86.45**
$DABP_{5*5}$+AA-ReLU	78.94	90.66	88.96	84.90
$DABP_{7*7}$+AA-ReLU	76.15	88.96	87.32	83.08
Max-pool+ReLU	76.37	88.95	79.20	82.54
Max-pool+C-ReLU	75.01	88.98	79.89	83.04
Max-pool+AA-ReLU	76.90	89.06	89.23	83.85

Table 5.6: Ablation study under varying architectural settings.

nature.

All the training hyper-parameters are the same as discussed earlier in Section 5.3.1.

5.4 Ablation Study

To justify our design choices, we conduct an ablation study on ImageNet using ResNet-101 backbone. All the other settings remain identical to that of Section 5.3.

As Table 5.6 shows, a 3×3 Gaussian blur filter performs better than 5×5 and 7×7 filters. We hypothesise that too big of a blur kernel spreads the features across an unnecessarily large region. This is particularly concerning for deeper layers where filters have large receptive fields, e.g., in ResNet-101, a 224×224 input reduces down to 14×14 feature maps in the last DS layer. Using a 7×7 filter which is half as big as the feature map dimension turns out to be sub-optimal.

We also test the compatibility of different AFs with DAB-pool and max-pool. Although C-ReLU performs better than ReLU under all three shifts, it lags behind in clean accuracy. We argue that because of C-ReLU's bounded nature, it has some anti-aliasing property, but a large saturated region prevents it from achieving higher clean

accuracy. Overall, DABP$_{3*3}$+AA-ReLU is the best configuration achieving 2.41% and 2.42% higher average accuracy than DABP$_{3*3}$+C-ReLU and DAB$_{3*3}$+ReLU respectively.

5.5 Conclusion

To make CNNs robust to small image transformations and corruptions, we have shown that better anti-aliasing is achievable if depth is considered for blurred DS. To this end, we have introduced DAB-pool, and outlined ways to replace common DS operations with it. To stop reappearance of aliasing, we have proposed a new AF, i.e., AA-ReLU.

We have evaluated our method against a broad array of shift, adversarial attack, corruption, and perturbation, and demonstrated the efficacy of our method under challenging conditions. In doing so, we have also integrated our modules in three different networks and shown that the benefits generalise well despite architectural differences. Since classification network backbones are an integral part of many more vision tasks such as object detection and semantic segmentation, we believe our proposed methods are easily transferable to these tasks as well.

Leading to this point, we have discussed ways of making CNNs robust to challenges such as image distortions and transformations. However, none of these paradigms consider the possibility of an unknown class's presence in the test environment. In the next chapter, we will discuss why it is important to consider such unknown or open set images in the test set-up and how we improve the existing methods for OSR.

Chapter 6

A Novel Training Paradigm for Open Set Recognition

In the previous chapter, we discussed how AA-ReLU coupled with DAB-pool can improve CNN's robustness against different image transformations, perturbations, and distortions. In addition to the challenges covered in the previous chapters, CNNs are vulnerable to open set or unknown class images also known as the Open Set Recognition problem (OSR). Open set images are samples belonging to classes a CNN has not seen during training.

CNNs are commonly trained and tested in closed set arrangements. Test instances are unseen samples but belong to one of the 'Known Known' (KK) classes used in training. In OSR, an input may belong to a completely 'Unknown Unknown' (UU) class. However, a closed set CNN will classify such an UU sample as one of the KK classes. In simple words, conventional CNNs cannot detect classes it is not trained on.

In this chapter, we propose a novel OSR network (OSRNet) that relies on a hard known unknown data mining method to recognise unknowns. We show in Section 6.4 that the proposed methods substantially improve the robustness of CNNs in OSR. The core contributions of this chapter are currently under review in the journal of Computer Vision and Image Understanding.

The rest of this chapter is organised as follows. Section 6.1 presents an introduction of this chapter. In Section 6.2, we present the architecture of the proposed OSRNet, a systematic process for mining a training set for OSRNet, and then explain why it works so well. In Section 6.3, we outline the datasets used for training and testing and implementation details. Section 6.4 provides an experimental evaluation of the

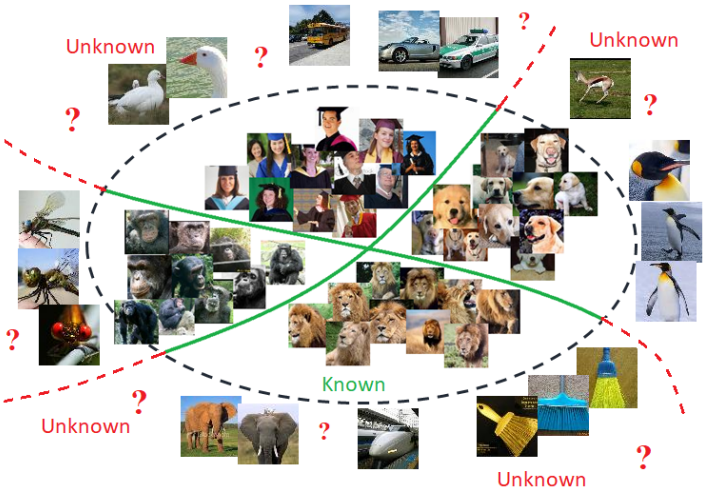

Figure 6.1: Conceptual illustration of a 4-class classifier decision boundary. A CNN can produce high-precision classification results as long as a novel test instance belongs to one of the KK classes. However, when the CNN is exposed to UU instances, it misclassifies such images as one of the KK classes because of its lack of precision outside the KK distribution.

proposed and existing methods. Section 6.5 provides a discussion on the Known Unknown Trainer (KUT) image set. Finally, Section 6.6 concludes this chapter.

6.1 Introduction

An image classifier is expected to correctly classify images belonging to the KK[1] classes seen during training. However, during inference, UU[2] instances might trigger incorrect classification. As the training set comprises a finite number of classes, identifying a UU instance is a challenge (see Figure 6.1). This is referred to as the OSR problem. To better understand the importance of OSR, let us consider an example. Imagine an autonomous car is trained to recognise 10 different street signs, and the car responds according to a pre-defined set of rules. The on-board sensors perceive the environment and feed data to the trained classifier (e.g., CNN). In an environment full of objects beyond the closed set training classes, it is likely for a CNN to classify a non-street sign

[1]KK refers to a standard training dataset of known classes.
KU refers to the Known Unknown classes used to represent the unknown world during training.
UU is the unknown dataset used only for testing.

[2]Note that we refer to the collection of Known Unknowns (KUs) and Unknown Unknowns (UUs) as unknowns.

object as one of the 10 street signs (e.g., a commercial billboard might be incorrectly perceived as a stop sign). Such an event and the consequent pre-programmed response of the autonomous car can lead to an undesired situation, even to a fatal accident.

In conventional classification tasks, a deep network is usually trained only on a KK dataset D_{KK} without considering the unknowns. For an effective OSR, training a network with 'everything else' in the world as the unknown is unrealistic. Therefore, a KUT dataset (referred to as D_{KUT} hereafter) is required for this task. Generative Adversarial Network (GAN)-based artificial images have often been used as a data augmentation-based solution in a number of contemporary works [157, 78, 80]. However, being trained on only KK samples, the distribution of the augmented data is not always compatible with the UU distribution present in the test set. Moreover, GANs suffer from a number of issues stemming from the generation techniques themselves, such as unwanted artefacts, and mode collapse [158]. These deteriorate the network's performance and increase training time. Another way to address the OSR problem is to find an effective SoftMax threshold to reject UUs [79, 82, 80].

In this chapter, we argue that drawing a boundary between the KK and the unknown is the key to effectively handling the OSR task and aim to approximate such a decision boundary by collecting a D_{KUT}. To accomplish this, we propose a way of mining the hard KU negatives into D_{KUT} and design a deep network to be trained on this dataset. D_{KUT} is mined from publicly available benchmark datasets (D_x) (where, $D_{KUT} \subset D_x$). Images from D_x inducing high probability ($P >$ some threshold T) for one of the KK classes are admitted into D_{KUT} (calculation of T is explained in Section 3). We demonstrate that OSRNet, which is trained to distinguish D_{KUT} from D_{KK} can identify novel UU instances at test time. D_{KUT} does not include any of the classes present in the UU test fold to ensure a fair evaluation.

The proposed OSRNet has two parts: a conventional CNN as the base and a Confidence Subnetwork (CS). The CNN is trained conventionally to classify a given D_{KK}. The CS, which effectively is an ANN, is separately trained to identify UUs. Once the CS is trained, it is augmented to the trained CNN, and OSRNet is formed (details in Section 6.2). At inference time, the newly formed network works as one single end-to-end unit. Inside OSRNet, the CNN produces class predictions and at the same

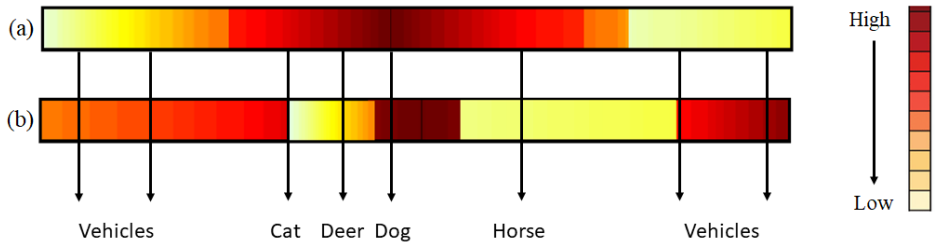

Figure 6.2: Behavioural difference of a trained CNN on classes it has and has not seen during training. For a KK test instance, usually, the correct class receives maximum probability, and the leftover energy is distributed to visually similar classes. In contrast, no such pattern is found for an unknown test instance. **(a)** denotes the probability heat map for CIFAR-10 dog class (test set). Most of the energy is distributed to either dog or among classes visually similar to dog, e.g., cat, deer, and horse. In **(b)**, when unknown instances (taken from CIFAR-100) are misclassified as dog, the leftover energy is distributed almost randomly to other classes.

time, the CS outputs a single confidence score $S = [0, 1]$ to indicate whether an instance belongs to one of the KKs or not ($S \rightarrow 1$ denotes a high chance of the input being UU). A cut-off value δ is used on S to determine the final outcome, i.e., whether to accept or reject the class label produced by the base CNN. OSRNet does not require any additional computation module (e.g., EVT) outside the deep network's perimeter. The entire inference process is end-to-end without any bottleneck.

The OSRNet architecture is inspired from the observation by Bendale et al. [79] depicted in Figure 6.2. It is reported that when classifying an image from D_{KK}, conventional CNNs generally produce a high probability score for the correct class while the leftover probability is usually distributed across visually similar classes. Even when a CNN is unsure or misclassifies a KK instance, the probability scores mostly remain concentrated to visually similar classes. In contrast, when an unknown instance gets classified as one of the KK classes, the leftover probability distribution does not usually follow such a pattern [79]. For illustration (in Figure 6.2), we train a simple CNN on CIFAR-10 and generate a probability heat map for the dog class test set. It is evident in Figure 6.2(a) that either dog or classes visually similar to dog receives most of the energy. In Figure 6.2(b), we feed the same CNN instances from unseen CIFAR-100 classes and produce the heat map with 1,000 samples misclassified as dog. The leftover probability distribution has marked contrast to Figure 6.2(a). This is the pattern we aim to exploit in this chapter for OSR. Because of the above-mentioned behavioural

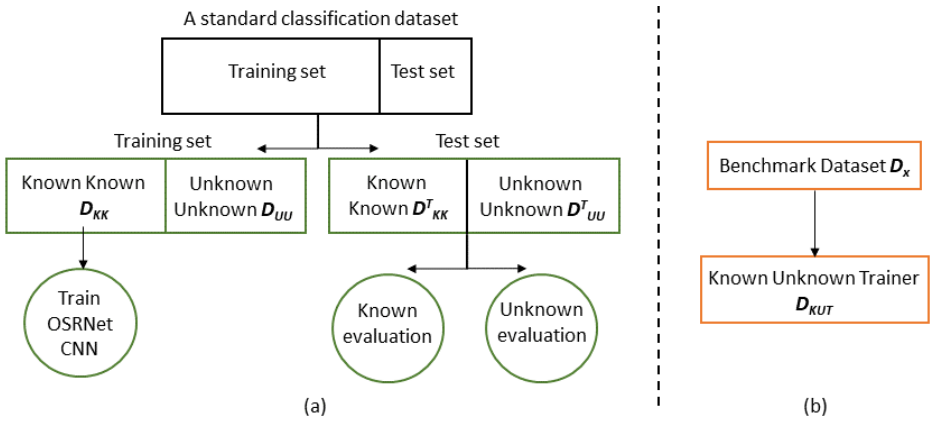

Figure 6.3: (a) An overview of the dataset splits in this chapter. **(b)** The Known Unknown Trainer dataset D_{KUT} is a subset of D_x mined following our proposed method. D_{KUT} does not contain any of the $D^T{}_{UU}/D_{UU}$ classes. As a result, OSRNet remains blind regarding the classes it is going to encounter during evaluation.

difference on KKs and any unknown instances, we expect the intermediate FC layer features of a CNN (trained on D_{KK}) to reflect this difference. Therefore, FC features should be distinct for KK and unknown samples. Collecting such FC features for D_{KK} is straightforward from our base CNN. However, mining an appropriate D_{KUT} set to collect FC features is challenging. Our proposed method addresses both issues.

In this chapter, we make the following contributions:

- We propose a way of building the KUT dataset D_{KUT} for OSR.
- To effectively distinguish between the KKs from the UUs at test time, a novel deep network (OSRNet) is proposed.
- We extensively evaluate OSRNet and compare with contemporary OSR methods on six benchmark datasets. OSRNet not only detects UUs with higher precision but also exhibits impressive discriminative power within D_{KK}.
- Finally, we discuss the underlying reasons behind the effectiveness of the proposed D_{KUT} and OSRNet in OSR.

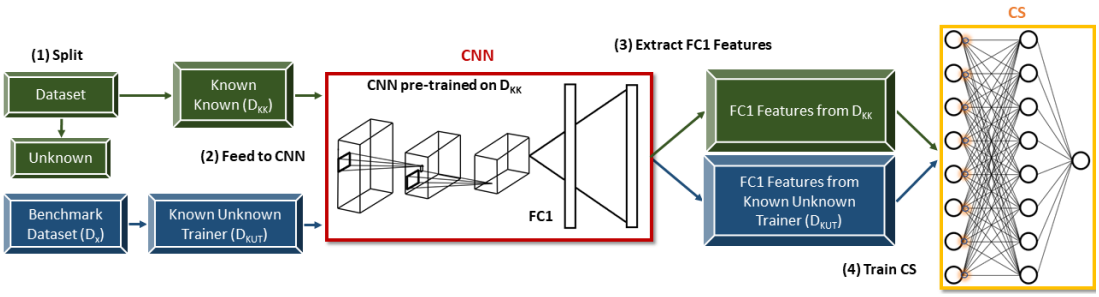

Figure 6.4: An overview of OSRNet training process. A CNN ($ResNet_{2FC}$) is trained on the KK split (D_{KK}) of a dataset. Images from D_{KK} and a D_{KUT} mined from another benchmark dataset D_x are fed to this already trained CNN. FC_1 features are extracted for both D_{KK} and D_{KUT} and the CS is trained with these features and binary labels. Once trained, this CS is augmented to the corresponding CNN layer (FC_1) and the proposed OSRNet is formed. At inference time, OSRNet works as one single end-to-end unit. The augmented CS receives FC_1 features from the CNN and detects whether the input belongs to one of the D_{KK} classes or not. The CNN, on the other hand, simultaneously classifies an instance without any interference from the CS.

6.2 Proposed Method

In this section, we first present the architecture of the proposed OSRNet. Then we provide details on how D_{KUT} is mined. Lastly, we provide training details for OSRNet and outline why it works better.

6.2.1 OSRNet

Before elaborating on the proposed method, we outline the notations of the dataset splits:

- (**Training**) One KK set D_{KK}.
- (**Training**) One KUT set D_{KUT} mined from a D_x following our proposed method (details in Section 6.2.2).
- (**Test**) One KK test set $D^T{}_{KK}$.
- (**Test**) One UU test set $D^T{}_{UU}$.

Following the evaluation process defined in [78, 82], we split a standard classification dataset (e.g., CIFAR10) into a KK part D_{KK} and a UU part D_{UU}. D_{UU} is left aside

and only its test set counterpart $D^T{}_{UU}$ is used for evaluation along with $D^T{}_{KK}$. $D^T{}_{UU}$ is used only for testing and our network remains unaware of the classes contained in $D^T{}_{UU}$. A KUT set D_{KUT} is mined from another publicly available benchmark dataset D_x. Classes contained in either D_{KK} or $D^T{}_{UU}$ are removed from D_x so that D_{KUT} does not contain any overlapping classes ($D_{\text{KK}} \cap D^T{}_{UU} \cap D_{\text{KUT}} = \varnothing$). Figure 6.3 provides an overview of the dataset splits.

OSRNet has two parts: a base CNN responsible for classifying an instance (x) regardless of its distribution, i.e., KK or UU and the CS augmented to the base CNN outputing a score indicating whether $x \in D_{KK}$ or $x \notin D_{KK}$ (see Figure 6.4). In other words, even if an input belongs to a class that our network has not seen during training, the CNN will output a label. It is up to the CS to decide whether to accept or reject it. Formally, an input image x triggers the N-way CNN in OSRNet (N is the number of classes) to produce a probability y_{i} ($i \in \{1...N\}$) for each of the classes. The CS outputs a score S for each x, indicating the confidence of the input being a UU instance. The final OSRNet output OP can be expressed using Equation 6.1.

$$OP = \begin{cases} \max P\left(y_i|x\right) & \text{if } S < \delta, \\ x \notin D_{KK} & \text{else} \end{cases} \quad \text{where, } i \in \{1...N\} \tag{6.1}$$

As shown later, an optimally chosen cut-off value δ can be applied on S to reject UU instances.

As discussed in Section 6.1 and depicted in Figure 6.2, the probability scores of a CNN are confined within the visually similar classes when the input belongs to D_{KK}. However, the probability distribution does not follow the same trend for unknown instances. Therefore, the intermediate layer features of a CNN should exhibit differences for the KK and unknown classes. Since we aim to exploit this behavioural distinction, the choice of the feature layer is important. In contrast to a single FC layer (FC_{Soft}) used in conventional ResNet (we call it $\text{ResNet}_{\text{1FC}}$) [26], we add one additional FC layer (FC_1) prior to FC_{Soft} and train this ResNet (we call it $\text{ResNet}_{\text{2FC}}$) on D_{KK} from scratch. A detailed analysis on the impact of different ResNet variants and OSRNet's performance is provided in Section 6.4.

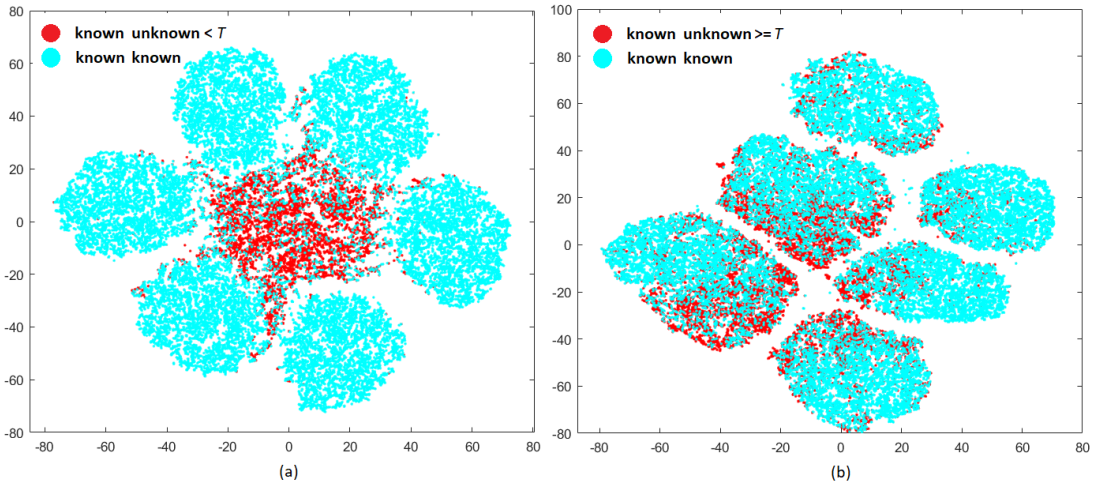

Figure 6.5: t-SNE plots of **(a)** FC_1 features from all images in D_{KK} (cyan) and those in D_x (red) inducing probability $P <$ some threshold T and **(b)** $P \geq T$. It is visually evident that separating KKs from KUs in (b) is more difficult. OSRNet trained to separate KKs from KUs in (b) can identify relatively easier KUs in (a) as well. An effective probability threshold T is chosen based on finding the maximum point on a cubic polynomial curve (see Figure 6.6).

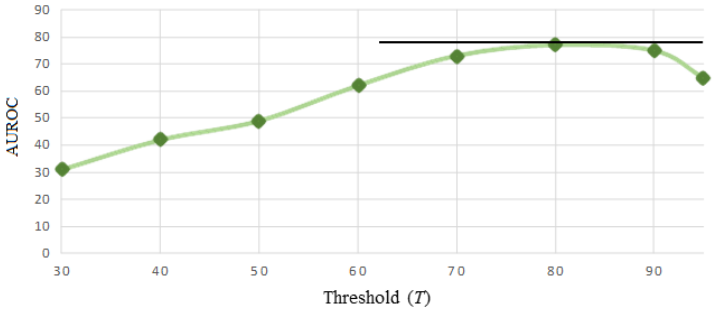

Figure 6.6: AUROC vs T curve based on sample points for Tiny ImageNet as D_{KK} and Caltech256 as D_x. A probability threshold T is calculated from the maximum point on the curve. This threshold T calculation is repeated for each D_{KK} reported in this chapter.

Table 6.1: An empirical analysis of OSRNet's performance on different D_{KUT} selection criteria or probability threshold T. It is evident that images from D_x with $P \geq 80\%$ perform best as D_{KUT}. The average entropy is also the lowest around this point.

KK	D_x (AUROC %)						
	Probability Threshold T (%)	Entropy $H(X)$	CIFAR100	Indoor67	Caltech256	T-ImageNet	F-MNIST + ADBase
CIFAR10	60	1.65	75.60	69.25	78.65	78.23	-
	70	1.41	78.85	75.02	84.30	82.30	-
	80	1.10	89.25	83.82	93.15	88.40	-
	90	1.21	86.32	81.40	89.66	85.49	-
CIFAR+10	60	1.71	-	72.69	80.20	80.60	-
	70	1.45	-	79.05	88.54	84.25	-
	80	0.98	-	85.76	95.69	91.40	-
	90	1.16	-	82.85	92.36	88.71	-
CIFAR+50	60	1.55	-	71.15	83.69	77.06	-
	70	1.51	-	79.30	90.04	84.55	-
	80	1.18	-	83.60	94.60	89.78	-
	90	1.24	-	80.95	92.69	87.45	-
T-ImageNet	60	1.92	60.88	55.01	62.22	-	-
	70	1.64	69.35	65.25	73.41	-	-
	80	1.36	74.55	70.95	77.10	-	-
	90	1.48	72.98	68.07	75.94	-	-
MNIST	60	1.33	-	-	-	-	93.42
	70	1.18	-	-	-	-	95.36
	80	0.91	-	-	-	-	98.96
	90	1.08	-	-	-	-	96.25
SVHN	60	1.47	-	-	-	-	88.40
	70	1.21	-	-	-	-	91.13
	80	1.02	-	-	-	-	92.66
	90	1.14	-	-	-	-	90.69

6.2.2 KUT Dataset Mining

After splitting a standard classification dataset into D_{KK} and D_{UU}, ResNet$_{2FC}$ is trained on D_{KK} and remains completely unaware of the classes in D_{UU}. To mine D_{KUT}, another publicly available benchmark dataset D_x is fed to a ResNet$_{2FC}$ trained on D_{KK}. From all the images in D_x, only the ones inducing probability greater than a threshold are mined for D_{KUT}, i.e., $\forall x \; \exists(x) \; (P(x) > T)$ (see Figure 6.5). Choosing an effective T is vital for OSRNet's performance. To select such a T, it is important to analyse how different T values fare against OSRNet's performance. Table 6.1 has a compilation of increasing T against Area Under ROC curve ($AUROC$) score of OSRNet. It is evident that for the given sample points, $T \approx 80\%$ is a good choice across datasets. Calculating OSRNet's performance on every possible T is tedious. Therefore, to pick a T, we fit a cubic polynomial curve for all the collected sample points across datasets (e.g., $(70, 78.85)$ is a data point for CIFAR10 (D_{KK})-CIFAR100 (D_{KU}) combination). The curve is represented by Equation 6.2 (A, B, C, and D are coefficients).

$$F(T) = A + BT + CT^2 + DT^3 \tag{6.2}$$

A degree two quadratic function has a higher error rate than a cubic one and hence we fit a cubic curve approximated by Equation 6.2 (see Figure 6.6). With the help of first and second-order derivatives, the maximum point on the curve is found where the gradient of a tangent to the curve $F(T)$ is 0. We fit an $AUROC$ vs T curve for all the D_{KK}s with Caltech256 as the D_x (as will be explained in the next section, Caltech256 is the best D_x). For each D_{KK} and D_x, a threshold T is calculated from the curve.

Key characteristics of this curve (Figure 6.6) are explained as follows:

- A smaller T allows a large number of images from D_x to qualify for D_{KUT}. A considerably small T might allow the entire D_x set to qualify for D_{KUT} (i.e., $D_x \approx D_{KUT}$ if $T \to 0$). As a result, D_{KUT} gets populated with sub-optimal imagery along with high probability inducing ones. OSRNet's performance (AUROC), upto a certain point, improves with increasing T.
- OSRNet's performance experiences a downward trend followed by the peak ($T \approx 80\%$). This is because too high of a T value leaves very few images from D_x

to be eligible for D_{KUT}. Such a D_{KUT} is inadequate as training dataset for OSRNet. Hence, OSRNet's performance drops as the meagre training data in D_{KUT} leads to overfitting.

- OSRNet's performance peaks around $T \approx 80\%$ for all the datasets. Such a T offers the best trade-off between the number and quality of images mined for D_{KUT}.

Algorithm 6.1 : D_{KUT} Construction Process
Input: Images (x_n) from D_x, $n \in N$, N is the total number of images in D_x.
Output: x_n either selected or discarded for D_{KUT} .

```
Choose a benchmark dataset as the base D_x
for all x_n ∈ D_x, where n = {1, 2, ...N} do
    Feed x_n to ResNet_2FC
    p = max(Probability(x_n))
    if p > T, where T is an optimal threshold found from the curve in Equ. 6.2 then
        D_KUT ← x_n
    else
        Discard x_n
    end if
end for
```

Algorithm 6.1 summarises the overall D_{KUT} mining process, which is used to teach OSRNet the essence of the UU world. For numeric datasets such as MNIST and SVHN as D_{KK}, the options for an ideal D_x are limited because natural object classes do not work well as D_x for such numeric D_{KK}s. In this chapter, Fashion MNIST [159] and ADBase [160] are used in conjunction as the base D_x.

6.2.3 Training OSRNet

Training OSRNet involves a series of steps. The overall training workflow is depicted in Figure 6.4, and the steps are explained as follows.

- ResNet$_{2FC}$ is used to extract FC_1 features from the same D_{KK} it was initially trained on. All these features are labelled as 0.
- D_{KUT} images (mined following the process in Section 6.2.2) are fed to the same ResNet$_{2FC}$ and FC_1 features are extracted. These features are labelled as 1.

- The CS (an ANN with one output node) is trained with these two sets of FC_1 features (from D_{KK} and D_{KUT}) to teach the difference between KKs and UUs. As the CS is capable of distinguishing relatively difficult borderline D_{KUT} features from D_{KK} features, it can identify rest of the relatively easier KUs in D_x.
- Finally, this trained CS is augmented to the corresponding (FC_1) layer of the pre-trained ResNet$_{2FC}$. OSRNet functions as a single unit at inference time.

Algorithm 6.2 summarises the OSRNet training process. As the CS takes care of UU detection, ResNet$_{2FC}$ within OSRNet maintains the original classification accuracy on the test fold of D_{KK}. At test time, OSRNet not only produces a class label but also provides a confidence score suggesting how likely an input is to be UU.

Algorithm 6.2 : OSRNet Training
Input: Individual images from D_{KUT} (referred x_i) and D_{KK} (referred x_j).
Output: Trained $OSRNet$.

for all $x_i \in D_{KUT}$, where $i = \{1, 2, ...N\}$ **do**
  Feed x_i to $ResNet_{2FC}$
  $Feature_{KUT}[i] \leftarrow FC_1$
  $Feature_{KUT\_label}[i] \leftarrow 0$
end for
for all $x_j \in D_{KK}$, where $j = \{1, 2, ...N\}$ **do**
  Feed x_j to $ResNet_{2FC}$
  $Feature_{KK}[j] \leftarrow FC_1$
  $Feature_{KK\_label}[j] \leftarrow 1$
end for
Train Confidence Subnetwork (CS) with $\{Feature_{KUT}, Feature_{KUT\_label}\}$ and $\{Feature_{KK}, Feature_{KK\_label}\}$
Augment CS to $ResNet_{2FC}$

6.2.4 Why OSRNet works?

Since perceiving high dimensional space is difficult for humans, conceptual illustrations are widely used for better understanding [131, 161]. Here, with the help of Figure 6.7, we explain why the proposed D_{KUT} and OSRNet work well in OSR.

The asterisks within the dotted oval resemble D_{KK}, and classification within this oval is quite accurate. The negative space or the region beyond the encapsulating

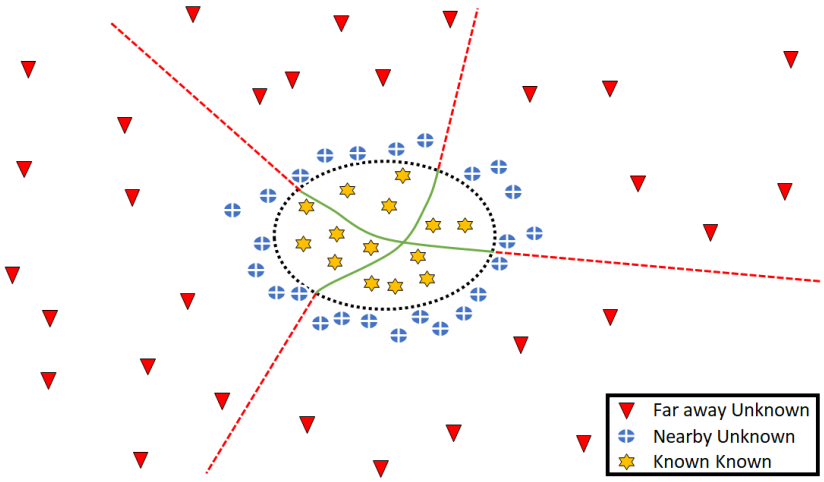

Figure 6.7: A conceptual illustration of the D_{KUT} image characteristics. D_{KUT} consists of only the nearby unknowns and OSRNet is trained to detect these difficult to identify unknowns. This way, the far-away unknowns are automatically detected without being used as part of the training. For OSRNet, less training data is required and better accuracy is achieved.

oval accommodates all the unknown images (as mentioned earlier, we refer to KUs and UUs as unknowns), i.e., nearby unknowns (circle), and far away unknowns (triangle). Conventional CNN's performance lack robustness in the negative space since the decision boundaries (dotted red) extrapolate to infinity [50] without any precision. In this chapter, we only use high probability ($P > T$) inducing images in D_{KUT} to represent the UU. Such images (blue circles) in D_{KUT} reside close to the KK distribution (the dotted oval). We argue that a deep classifier capable of treating the borderline D_{KUT} instances (blue circles) as unknown can easily identify relatively easier far-away unknowns (triangles). It is understandably a challenging task to collect all the borderline unknowns perfectly encapsulating the KK distribution.

In this chapter, we strive to maximise the participation of such borderline images in D_{KUT}. Two follow-up questions remain:

- (**Q1**) For a certain D_{KK}, which D_x should we choose to mine D_{KUT}?
- (**Q2**) Can a D_{KUT} with fewer data outperform some D_x it was mined from (although $D_{KUT} \subset D_x$)?

Answer to Q1. We experimented with four public datasets: CIFAR100, Indoor67, Clatech256, and Tiny ImageNet as D_x. It is evident from Table 6.1 that Caltech256

performs best as D_x across datasets and Indoor67 performs worst. One intriguing aspect is comparing the visual characteristics of the best performing D_x against others. Is it visually similar to D_{KK} ($D_x \approx D_{KK}$) or is it quite different (we show quantitative distance between different data distributions in Section 6.5)? For example, bus is a visually similar class to truck compared to some indoor images. It turns out, Indoor67 is not a good candidate for D_x as it consists of only interior images (e.g., office room, bedroom, and auditorium) with little to no visual similarity to the D_{KK}s. On the other hand, Caltech256 performs better as it consists of classes ranging from a variety of animals to different vehicles having greater visual similarity to the D_{KK}s.

Our suggestion is to use a D_x that has classes similar to D_{KK}. For example, if a certain D_{KK} consists of different dog breeds, a dataset containing different cat breeds could be a better option as D_x rather than using some indoor images like Indoor67 as the D_x. We argue that training to separate an apparently more difficult D_x, like Caltech256, from D_{KK} enables OSRNet to automatically distinguish relatively easier instances at test time.

Answer to Q2. Experimental analysis shows that the type of images in D_{KUT} is more important than the number of images. It is intriguing that despite being a subset of D_x, D_{KUT} teaches OSRNet the essence of the unknown world better than a much larger in size D_x. A detailed explanation of this observation is provided in Section 6.5.

To further understand the conceptual explanation and why mining only high probability inducing KUs for D_{KUT} makes sense, we resort to entropy comparison. In multi-class classification tasks such as ours, a trained CNN provides a class probability for each of the output nodes. The probability distribution (D_P) generated by the CNN portrays how certain the network is about the classification. When a test instance is classified confidently, one output node exhibits higher (≈ 1) probability than the rest. The uncertainty factor is low here as is the entropy. Conversely, when the classifier is susceptible to the input and does not provide a confident probability distribution (e.g., a Uniform distribution), the uncertainty is high and hence the entropy as well. This suggests that for an input, lower the D_P uncertainty, closer the input is to D_{KK}. This uncertainty can be quantified from D_P entropy using Equation 6.3.

$$H(X) = H\left(p_1, \ldots, p_n\right) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{6.3}$$

In a number of cases, when a trained CNN faces UU instances, the entropy or $H(X)$ stays low. This generally means those images are adjacent to the KK distribution despite actually being UU. On the other hand, the rest of the images induce high $H(X)$ which implies such images are generally not adjacent to D_{KK} and rightly so. As can be seen from Table 6.1, the lowest entropy co-occurs with the best performing T. This supports the usage of only high probability inducing images ($P > T$) in D_{KUT} since such a D_{KUT} would be closer to D_{KK} and harder to distinguish resulting in a tight but efficient decision boundary.

6.3 Experiments

6.3.1 Datasets and Splits

MNIST, SVHN, and CIFAR10. MNIST [162] is a digits dataset containing 60,000 training and 10,000 test images. Each digit from 0 to 9 denotes a class. All the images are grayscale and have a resolution of 28×28. SVHN [163] is also a digits dataset, but the images are collected from Google Street View cameras capturing house numbers (from 0 to 9 as well). This dataset is considered harder than MNIST and contains 32×32 colour images. It has 73,257 training and 26,032 test instances. CIFAR10 [127] also has 10 classes with 32×32 colour images. It has 50,000 training and 10,000 test images of different objects (e.g., cat and dog). To train OSRNet individually, each of these three datasets is split into D_{KK} with six classes and D_{UU} with four classes. The split for corresponding test set is same as the training set, i.e., ${D^T}_{KK}$ and ${D^T}_{UU}$ contain same classes as their training split. For example, while training on MNIST, six randomly selected classes are used as D_{KK} and the other four classes are considered as D_{UU}. For each dataset, the 'openness' is estimated by Equation 6.4 [72, 78]. Greater openness value implies higher difference in the number of classes between $D_{UU}/{D^T}_{UU}$ and $D_{KK}/{D^T}_{KK}$, i.e., the UU to KK class ratio is higher. It is worth reinstating that no images from D_{UU} are used in training OSRNet. Only the test fold ${D^T}_{UU}$ is used for testing.

$$\text{openness } = 1 - \sqrt{\frac{|\,\mathrm{N}\,|}{|\,\mathrm{Q}|}} \tag{6.4}$$

Here, $|N|$ and $|Q|$ denote the number of D_{KK} classes and the number of total test classes ($D^T{}_{UU} + D^T{}_{KK}$) respectively. The term $openness$ is measured in percentage where higher value represents greater $openness$. According to Equation 6.4, $openness$ of CIFAR10, MNIST and SVHN is 22.50%.

CIFAR+10. Four classes from CIFAR10 are selected as D_{KK} and 10 non-overlapping classes from CIFAR100 are selected as D_{UU}. $openness$ is 46.50%.

CIFAR+50. Four classes from CIFAR10 are selected as D_{KK} and 50 non-overlapping classes from CIFAR100 are selected as D_{UU}. $openness$ is 72.78%.

Tiny ImageNet. Tiny ImageNet is a subset of ImageNet [1] comprising 200 classes with 64×64 colour images. Each of the classes has 500 training and 50 test images. We randomly select 20 classes as D_{KK} and 180 classes as D_{UU}. $openness$ is 68.38% The class selection process is random, and for each benchmark dataset, we experiment with five iterations and the average is reported (Section 6.4).

6.3.2 KUT Datasets

We use publicly available benchmark datasets (D_x) to mine D_{KUT}.

Caltech256. As explained earlier, Caltech256 performs best as D_x for all non-digit D_{KK}s. It has 256 object categories containing 30,607 images. All the overlapping classes among D_x, D_{KK}, D_{UU} are removed to achieve a fair (no prior knowledge about $D^T{}_{UU}$) and sane (no D_{KK} class is used later in training as UU) D_{KUT}.

Fashion MNIST, ADBase. Fashion MNIST [159] has exactly the same attributes as MNIST but contains 10 classes of fashion products. ADBase [160] is the Arabic version of MNIST. Images in both of these datasets are of resolution 28×28 and grayscale. These two datasets are used collectively as the D_x for both of the numeric datasets, i.e., when D_{KK} belongs to either MNIST or SVHN.

6.3.3 Training CNN

CIFAR10, CIFAR+10, CIFAR+50, SVHN. Each of these datasets has 32×32 colour images. After splitting the datasets into D_{KK} and D_{UU} following the process described

in the previous section, a CNN classifier is trained on D_{KK}. ResNet$_{2FC}$ (with depth 20 [26]) is used as the CNN in OSRNet. The first 3×3 convolution layer is followed by six $3 \times 3 \times 16$, six $3 \times 3 \times 32$ and six $3 \times 3 \times 64$ convolution layers where the first two dimensions represent the filter size and the third dimension stands for the number of filters. The SoftMax output is preceded by an N-way (N is the number of classes in D_{KK}) FC_{Soft} layer. FC_{Soft} layer in turn is preceded by the additional FC_1 layer.

Tiny ImageNet. ResNet$_{2FC}$ (with depth 32 [26]) is used to train on D_{KK} of Tiny ImageNet. It has six $3 \times 3 \times 16$, $3 \times 3 \times 32$, six $3 \times 3 \times 64$, six $3 \times 3 \times 128$, and $3 \times 3 \times 256$ convolution layers. The FC_{Soft} layer here as well, is preceded by an additional FC_1 layer. Since images in this dataset have a higher spatial resolution, a deeper variant of ResNet$_{2FC}$ is used.

The greater distinctiveness of the additional FC_1 features from ResNet$_{2FC}$ compared to solitaire FC_{Soft} features in ResNet aids OSRNet's OSR performance. The benefits come at a negligible increase in the total number of network parameters ($\approx 3.5\%$). Detailed results are provided in Section 6.4.

MNIST. For MNIST, we train a plain CNN on D_{KK} consisting of three 3×3 convolution blocks with respective filter numbers of 8, 16, and 32. These layers are followed by two FC layers (128 unit FC_1 and N-way FC_{Soft}) and the SoftMax output layer. ResNet architecture is not adopted for MNIST as a plain CNN network works just fine with competitive classification accuracy (99.66%).

We adopted SGD, data shuffling before every epoch while training and data augmentation (horizontal flip and translation). Minibatch size of 128 is used for all the datasets except MNIST (8,192). Adaptive dropout [55] is followed to avoid overfitting. A multi-class cross-entropy loss ($\mathcal{L}_m$) is used as the objective function (see Equation 6.5).

$$\mathcal{L}_{\mathrm{m}} = -\sum_{i=1}^{N}\sum_{j=1}^{K} \mathrm{t}_{ij} \ln y_{ij} \tag{6.5}$$

where N is the number of samples, K is the number of classes, t_{ij} denotes that the i^{th} sample belongs to the j^{th} class, and y_{ij} is the output for sample i for class j, which

Table 6.2: AUROC performance (%) comparison of different OSR and OOD methods.

Method	CIFAR10	CIFAR+10	CIFAR+50	MNIST	SVHN	Tiny ImageNet
SoftMax	67.70	81.60	80.50	97.80	88.60	57.70
OpenMax (Bendale et al. [79])	69.50	81.70	79.60	98.10	89.40	57.60
G-OpenMax (Ge et al. [80])	67.50	82.70	81.90	98.40	89.60	58.00
OSRCI (Neal et al. [78])	69.90	83.80	82.70	98.80	91.00	58.60
C2AE (Oza et al. [82])	89.50	95.50	93.70	98.90	92.20	74.80
OE ([88])	63.50	73.40	71.20	93.11	81.40	44.90
ODIN ([84])	64.10	75.10	72.20	96.10	81.30	47.10
G-ODIN ([164])	68.80	79.10	73.20	97.40	84.70	47.30
Proposed Method (D_x)	90.40 ± 0.08	94.88 ± 0.12	93.50 ± 0.11	98.35 ± 0.09	91.22 ± 0.14	$75.70 \pm 0..20$
Proposed Method (D_{KUT}-ResNet1FC)	91.66 ± 0.05	94.95 ± 0.08	94.03 ± 0.10	98.44 ± 0.05	91.45 ± 0.10	76.80 ± 0.16
Proposed Method (D_{KUT}-ResNet2FC)	**93.15** ± 0.04	**95.69** ± 0.06	**94.60** ± 0.07	**98.96** ± 0.04	**92.66** ± 0.09	**77.10** ± 0.13

effectively is the value from the SoftMax function, i.e., it is the probability that the network associates the i^{th} input with class j.

6.3.4 Training CS

As depicted earlier in Figure 6.4, CS is an ANN subnetwork within OSRNet and is responsible for detecting UUs while ResNet$_{2FC}$ classifies the instance. Features (for D_{KK} and D_{KUT}) extracted from ResNet$_{2FC}$'s FC_1 layer are fed to CS as the training data. Binary class labels are supplied as the ground truth and variable learning rate gradient descent (GDX) with momentum is used as the training method. We train CS with two hidden layers and the number of hidden units (H) per layer is between the number of inputs and outputs [165]. For 128 dimensional FC_1 features, H is set to 64, 128, and 256 while for 256 dimensional FC_1 features, H is set to 128, 256, and 512. A binary cross-entropy loss function ($\mathcal{L}_b$) is used following Equation 6.6.

$$\mathcal{L}_b = -\frac{1}{N}\sum_{i=1}^{M} T_i \log\left(Y_i\right) \tag{6.6}$$

where M is the total number of responses in Y, N is the total number of observations in Y, Y_i is the network output, and T_i is the target value.

For each configuration, 10-fold training is conducted. Later, the best average score yielding CS is augmented to the corresponding ResNet$_{2FC}$ to form OSRNet. It is observed that, CS with H as 64 performs best for 20-depth ResNet$_{2FC}$ and CS with H as 128 performs best for 32-depth ResNet$_{2FC}$. Hence the reported results in Table 6.2 are derived using these two ensembles.

6.3.5 Optimum Confidence Cut-Off δ Estimation

AUROC provides an overview of a binary classifier's performance. A greater AUROC magnitude indicates better performance. However, when the classifier is deployed in a real-life application, it is expected to rule out negative samples based on a cut-off value δ. A classifier's ultimate accuracy largely depends on the choice of δ. An optimal δ ensures maximum correct classification with minimum error. Such a δ can be estimated from the optimal operative point on the ROC curve. There are multiple ways of finding such a point on ROC curve that will provide a good compromise between FPR (1 – specificity) and TPR (sensitivity). However, in this chapter, we follow the optimal slope intersection method [166, 167] where an initial slope S_{op} is calculated in the ROC curve from the misclassification cost using Equation 6.7.

$$S_{\text{op}} = \frac{\text{C}(P|N) - \text{C}(N|N)}{\text{C}(N|P) - \text{C}(P|P)} \times \frac{N}{P} \tag{6.7}$$

$C(N|P)$ is the cost of misclassifying a positive class as a negative class. $C(P|N)$ is the cost of misclassifying a negative class as a positive class. In our case, the cost of any form of misclassification is equally penalised ($C(N|P) = C(P|N) = 0.5$). On the other hand, there is no penalty for a correct classification ($C(N|N) = C(P|P) = 0$).

$P = TruePositive + FalseNegative$

$N = TrueNegative + FalsePositive.$

A line with the slope S_{op} is dragged from the upper-left corner of the ROC plot (where $FPR = 0, TPR = 1$) down and to the right. The first intersection point of the slope line and the ROC curve is the optimal operating point. The value which ensures the optimal cut-off point is selected as the δ when OSRNet is deployed.

6.4 Performance Comparison

An open set recogniser performs two tasks simultaneously: it has to identify an input either as KK or UU and classify it correctly if KK. While accuracy is a good metric to gauge classification performance, UU detection performance should be evaluated on a metric that takes True Positive Rate (TPR) and False Positive Rate (FPR) into account.

Table 6.3: Classification accuracy (%) comparison of contemporary OSR methods. OSRNet_1 is the variant of OSRNet with ResNet_{1FC}.

CIFAR10 D_{KK}				
OSRNet_1	OSRNet	OpenMax	G-OpenMax	OSRCI
80.20	82.91	80.10	81.60	82.10
MNIST D_{KK}				
99.50	99.72	99.50	99.60	99.60
SVHN D_{KK}				
94.61	95.95	94.70	94.80	95.10

Table 6.4: Classification accuracy (%) comparison among conventional ResNet_{1FC}, ResNet_{2FC}, and ResNet_{3FC}. ResNet_{2FC} performs better on the benchmark classification task (without KK-UU split). Tiny ImageNet does not have labelled test set hence not shown here.

	CIFAR10		
	ResNet_{1FC} 90.24	ResNet_{2FC} 90.27	ResNet_{3FC} 88.95
Classification Accuracy (%)	**MNIST**		
	PlainCNN_{1FC} 98.87	PlainCNN_{2FC} 99.66	ResNet_{3FC} 98.66
	SVHN		
	ResNet_{1FC} 94.77	ResNet_{2FC} 95.29	ResNet_{3FC} 93.05

Therefore, we provide evaluation results separately with AUROC as the detection metric and TOP-1 accuracy as the classification metric. OSRNet outperforms other works in the literature under both metrics- AUROC (see Table 6.2 and Figure 6.8) and classification accuracy (see Table 6.3).

Class Conditioned Auto-Encoder (C2AE) [82] comes close to OSRNet in terms of AUROC performance on a number of datasets. C2AE follows a three-stage training scheme. First, a dataset is randomly divided into D_{KK} and D_{UU}. An encoder (a CNN minus the SoftMax output) is trained on D_{KK}. A decoder is augmented to the encoder output and is trained to reconstruct any given input. The entire network (when the encoder-decoder ensemble works as one unit) is designed to reconstruct any D_{KK} instance as precisely as possible. However, it is designed to poorly reconstruct D_{UU} instances so that the reconstruction error is high for UUs at inference time. Finally, at inference time, reconstruction error is calculated for each input. D_{KK} inputs are expected to be correctly reconstructed. Therefore, the reconstruction error for D_{KK} instances should be close to zero. For D_{UU} instances, however, the reconstruction will

have a significant error compared to its D_{KK} counterpart. This way, the magnitude of the error is exploited to determine whether an instance belongs to the UU or not. As advocated in [88], we attribute OSRNet's better task performance to the use of an effective, diverse, and real KUT dataset over encoder-decoder-based generative models.

Table 6.5: AUROC performance (%) comparison of different benchmark datasets as the base D_x for choosing D_{KUT}. The number of images inside D_x is listed in the column header and the number of images ultimately qualifying for D_{KUT} is provided inside the parenthesis in each row. Interestingly, D_{KUT} - although a subset of D_x and much smaller in size, works consistently better than using entire D_x as D_{KUT} across all four datasets.

KK	**UU**							
	CIFAR100		Indoor67		Caltech256		Tiny ImageNet	
	D_x (all 46k)	D_{KUT} ($>T$)	D_x (all 16k)	D_{KUT} ($>T$)	D_x (all 28k)	D_{KUT} ($>T$)	D_x (all 100k)	D_{KUT} ($>T$)
CIFAR10	86.10	89.25 (25k)	82.03	83.82 (9k)	90.40	93.15 (18k)	87.54	88.40 (64k)
CIFAR+10	-	-	83.65	85.76 (9.5k)	94.88	95.69 (19k)	88.45	91.40 (62k)
CIFAR+50	-	-	82.33	83.60 (9k)	93.50	94.60 (18k)	86.94	89.78 (60k)
Tiny ImageNet	72.06	74.55 (23k)	69.50	70.95 (7.5k)	75.70	77.10 (16k)	-	-

As mentioned earlier, detecting UUs is one side of OSR, while maintaining accuracy on classifying D_{KK} images is the other side of it. Compared to the conventional $\text{ResNet}_{1\text{FC}}$, our modified ResNet with two FC layers ($\text{ResNet}_{2\text{FC}}$) performs better on OSR task (see Table 6.2). $\text{ResNet}_{2\text{FC}}$ not only benefits our OSR performance but also ameliorates the classification accuracy (see Table 6.3). $\text{ResNet}_{2\text{FC}}$ performs better than $\text{ResNet}_{1\text{FC}}$ on the standard classification task (see Table 6.4). We argue that the increased depth of the FC layers in $\text{ResNet}_{2\text{FC}}$ provides richer features and boosts classification accuracy as well. However, using more than two FC layers does not work as well as using two (see our empirical analysis in Table 6.4).

Comparison with OOD methods. Outlier Exposure (OE) proposed in [88] emphasises on training with real and diverse KU datasets. To represent outliers (when $D_{\text{KK}} \subset$ **CIFAR10**), [84] trains on all the images in the '80 million tiny image' dataset [168] while we only use 18,000 images. In this case, our novelty lies in not naively using entire datasets as outliers. As for ODIN [84] and Generalised ODIN [164], both use temperature scaling while the latter not requiring any additional outlier training data. However, as shown in Table 6.2, OSRNet performs better than the above-mentioned

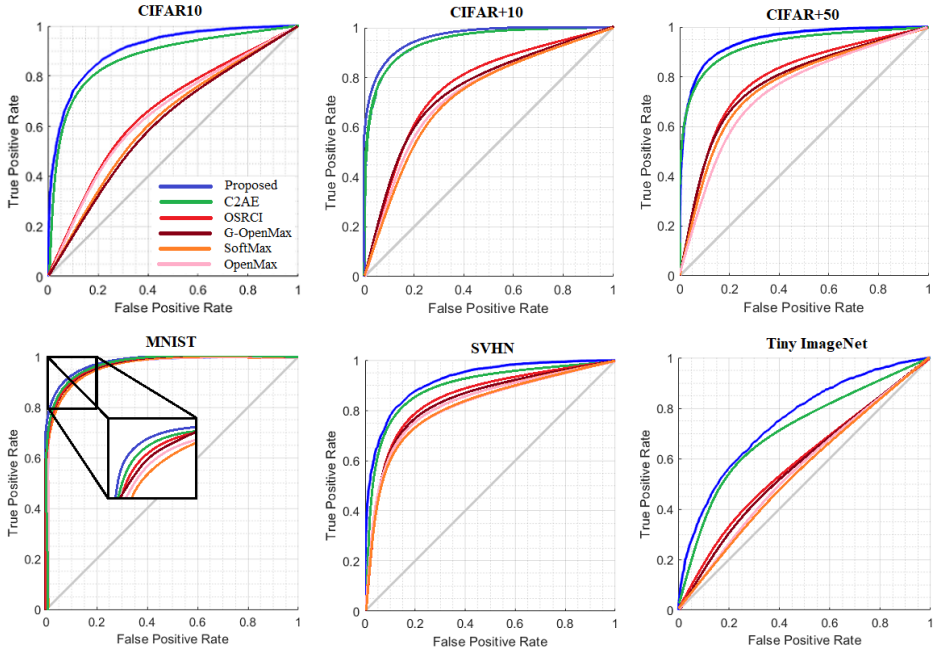

Figure 6.8: AUROC performance comparison between proposed OSRNet and other methods in the literature. OSRNet outperforms all the existing methods on all six benchmark datasets.

OOD methods under the OSR evaluation protocol. We attribute this to the different loss landscapes for these two paradigms which although similar, are not exactly same [150].

6.5 Discussion

To perform the OSR task, OSRNet is trained with a specially mined dataset D_{KUT}. We discussed in Section 6.2.2 the mining process of D_{KUT} and the type of images that work well as D_x. However, the significance of the size of D_{KUT} (or the number of images) is also worth discussing. Our experimental results show that the type of images is more important than the number of images. Our proposed method leaves only around half of the images (16,000) from Caltech256 (D_x) into D_{KUT}. However, it performs better than using the entire Caltech256 (28,000) as the D_{KUT} (Table 6.5). To understand the underlying reason behind this, we need to go back to the conceptual illustration provided in Figure 6.7. In order to detect UUs at test time, the best way is to collect a set of borderline KUs and train the classifier to draw a decision boundary between this

and D_{KK}. Anything beyond this decision line should be treated as unknown. Using an entire D_x set as D_{KUT} ([169]) is not a good idea since such a D_{KUT} would contain both the borderline (circles) and far-away (triangles) images. We argue that most of the KU instances from D_x that do not make it to the D_{KUT} can be automatically detected at test time if our classifier draws a line in between D_{KUT} (borderline ones) and D_{KK}. The following section elaborates on the experimental analysis.

CIFAR100. If we use all 46,000 images from CIFAR100 as the D_{KUT}, the AUROC scores come out as 86.10% and 72.06% for CIFAR10 and Tiny ImageNet respectively. Whereas, if the images inducing probability greater than 80% are used in D_{KUT}, the score becomes 89.25% (25,000) and 74.55% (23,000) for CIFAR10 and Tiny ImageNet respectively.

Indoor67. If we use all 16,000 images from Indoor67 as the D_{KUT}, the AUROC scores come out as 82.03%, 83.65%, 82.33%, and 69.50% for CIFAR10, CIFAR+10, CIFAR+50, and Tiny ImageNet respectively. Whereas, if the images inducing probability greater than 80% are used as D_{KUT}, the score becomes 83.82% (9,000), 85.76% (9,500), 83.60% (9,000), and 70.95% (7,500) for CIFAR10, CIFAR+10, CIFAR+50, and Tiny ImageNet respectively.

Caltech256. If we use all 28,000 images from Caltech256 as the D_{KUT}, the AUROC scores come out as 90.40%, 94.88%, 93.50%, and 75.70% for CIFAR10, CIFAR+10, CIFAR+50, and Tiny ImageNet respectively. Whereas, if the images inducing probability greater than 80% are used as the D_{KUT}, the score becomes 93.15% (18,000), 95.69% (19,000), 94.60% (18,000), and 77.10% (16,000) for CIFAR10, CIFAR+10, CIFAR+50, and Tiny ImageNet respectively.

Tiny ImageNet. If we use all 100,000 images from Tiny ImageNet as the D_{KUT}, the AUROC scores come out as 87.54%, 88.45%, and 86.94% for CIFAR10, CIFAR+10 and CIFAR+50 respectively. Whereas, if the images inducing probability greater than 80% are used as the D_{KUT}, the score becomes 88.40% (64,000), 91.40% (62,000), and 89.78% (60,000) for CIFAR10, CIFAR+10, and CIFAR+50 respectively.

Table 6.5 does not include digit-based MNIST and SVHN datasets in the KK column as such other datasets are scarcer. Non-digit datasets with natural images (e.g., Caltech256) do not work well at all as the D_x for MNIST and SVHN. It can be summarised that Caltech256 performs best as D_x for mining D_{KUT} because of its similarity with D_{KK} along with class diversity.

Train and Test Data Distributions. It is possible that even after explicitly removing all the overlapping classes, there could still be an implicit overlap between ${D^T}_{UU}$ and D_{KUT} because of the nature of these datasets. However, the core of our proposed method revolves around the fact that D_{KUT} lies close to the D_{KK} distribution. Therefore, our network's decision boundary around the D_{KK} distribution is fairly tight. Hence, OSRNet is capable of successfully identifying UUs regardless of the distribution of ${D^T}_{UU}$. A conceptual illustration is provided in Figure 6.7. To show OSRNet performs consistently with or without any implicit overlap between D_{KUT} and ${D^T}_{UU}$, we expand our test datasets for further experimentations. In addition to the default ${D^T}_{UU}$, we test on two separate ${D^T}_{UU}$s according to the following setup:

- $D_{KK} \subset CIFAR10$ (default train split).
- ${D^T}_{UU}\_S =$ SVHN testset.
- ${D^T}_{UU}\_I =$ Indoor67 testset.
- ${D^T}_{UU}\_C \subset CIFAR10$ testset (default unknown test split).
- ${D^T}_{UU}\_C \cap D_{KK} = \emptyset$.

CIFAR10 contains 10 different classes (such as bird and dog) and six randomly selected classes are used as the D_{KK}. Indoor67 dataset contains 67 interior classes, e.g., bedroom, office, and garage, while SVHN contains 10 classes belonging to digits (0-9). These two datasets do not visually or semantically overlap with the ${D^T}_{UU}$_C. In fact, samples from both ${D^T}_{UU}$_S and ${D^T}_{UU}$_I are easier to identify as outliers compared to ${D^T}_{UU}$_C owing to their greater distance from D_{KK}. Table 6.6 supports our claim, i.e., OSRNet performs better on distant and non-overlapping unknowns. On ${D^T}_{UU}$_S and ${D^T}_{UU}$_I, OSRNet produces 5.15% and 1.80% greater AUROC score

Table 6.6: $D^T{}_{UU}$ from the left out set of CIFAR-10 is the most challenging testset compared to $D^T{}_{UU}$s drawn from non-overlapping SVHN and Indoor67 testsets.

	$D_{KK} \subset$ **CIFAR10**		
	AUROC (%)		
	$D^T{}_{UU}$_C	$D^T{}_{UU}$_S	$D^T{}_{UU}$_I
C2AE ([82])	89.50	92.88	88.59
OSRNet	93.15	98.30	94.95

Table 6.7: Both $D^T{}_{UU}$_S and $D^T{}_{UU}$_I have greater distance from D_{KK} compared to the default testset $D^T{}_{UU}$_C. This complements our findings in Table 6.6 where OSRNet performs better on distant testsets.

	$D_{KK} \subset$ **CIFAR10**			
	$D_{KK} \rightarrow D_{KUT}$	$D_{KK} \rightarrow D^T{}_{UU}$_C CIFAR10	$D_{KK} \rightarrow D^T{}_{UU}$_S SVHN	$D_{KK} \rightarrow D^T{}_{UU}$_I Indoor67
MMD	0.18	0.23	0.59	0.44

respectively compared to $D^T{}_{UU}$_C. Table 6.7 reinforces our argument regarding the inter distribution distance measured using Maximum Mean Discrepancy (MMD) [170].

MMD can approximate the distance between the underlying distribution of two image datasets based on the Reproducing Kernel Hilbert Space (RKHS) [171, 172]. We used the final FC layer features from OSRNet backbone to represent the dataset distributions. Let $X = \{x_1, \ldots, x_{n_1}\}$ and $Y = \{y_1, \ldots, y_{n_2}\}$ be two datasets with distributions $\mathcal{P}$ and $\mathcal{Q}$. The empirical distance between $\mathcal{P}$ and $\mathcal{Q}$, according to MMD, is

$$\text{Dist}(\text{X}, \text{Y}) = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(x_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(y_i) \right\|_{\mathcal{H}}$$

where $\mathcal{H}$ is a universal RKHS [173], and $\phi : \mathcal{X} \rightarrow \mathcal{H}$

It is evident from Table 6.7 that D_{KUT} indeed resides close to D_{KK}. It is also apparent that visually and semantically distant datasets are indeed far away from D_{KK} and hence, easier to detect. This is analogous to our findings in Table 6.6 where OSRNet performs better on both $D^T{}_{UU}$_S and $D^T{}_{UU}$_I than $D^T{}_{UU}$_C. Our experimental results also complement the conceptual visualisation in Figure 6.7.

6.6 Conclusion

OSR is one of the pressing issues deep image classifiers are faced with. In this chapter, we analysed CNNs' behaviour on both KK and UU datasets and proposed a network that is adept at the OSR task. Instead of using synthetic images, we have demonstrated that a trainer dataset mined from publicly available datasets can represent the unknown better. OSRNet functions as a single end-to-end unit at inference time. It outperforms contemporary OSR methods on a number of benchmark datasets. We have also discussed the reasons behind OSRNet's success in the given task through conceptual decision boundaries and comparative entropy analysis.

So far, we have discussed several vulnerabilities of modern CNNs and showed novel ways of mitigating these issues. In the next chapter, we address why CNNs often struggle in detecting small objects and propose a novel network architecture to address this problem.

Chapter 7

Detecting Small Objects with a Faster RCNN based Novel Backbone Network

In the previous chapter, we discussed the challenges posed by open set samples, developed a novel OSR network architecture, and proposed an unknown trainer data mining algorithm for a more accurate OSR compared to other benchmark methods. In this chapter, we discuss the role of a CNN classifier in deep detection networks and discuss why detecting objects with large variation in scale – especially small objects, still remains a challenging task. To solve this problem, we propose a novel backbone network for Faster RCNN that can successfully detect small scale objects with high accuracy while maintaining high performance on larger objects as well. The main contribution of this work was accepted and presented at the 2019 Pacific-Rim Symposium on Image and Video Technology (PSIVT) [174].

The rest of this chapter is organized as follows. Section 7.1 provides an introduction of this chapter. In Section 7.2, we introduce the proposed backbone network BackNet and the design rationales. In Section 7.3, we discuss the training and test datasets and then provide a performance comparison of existing works with our proposed approach. Section 7.4 concludes this chapter.

7.1 Introduction

Object detection has long been a fundamental task in the field of computer vision. With the emergence of self-driving cars, intelligent surveillance systems, autonomous

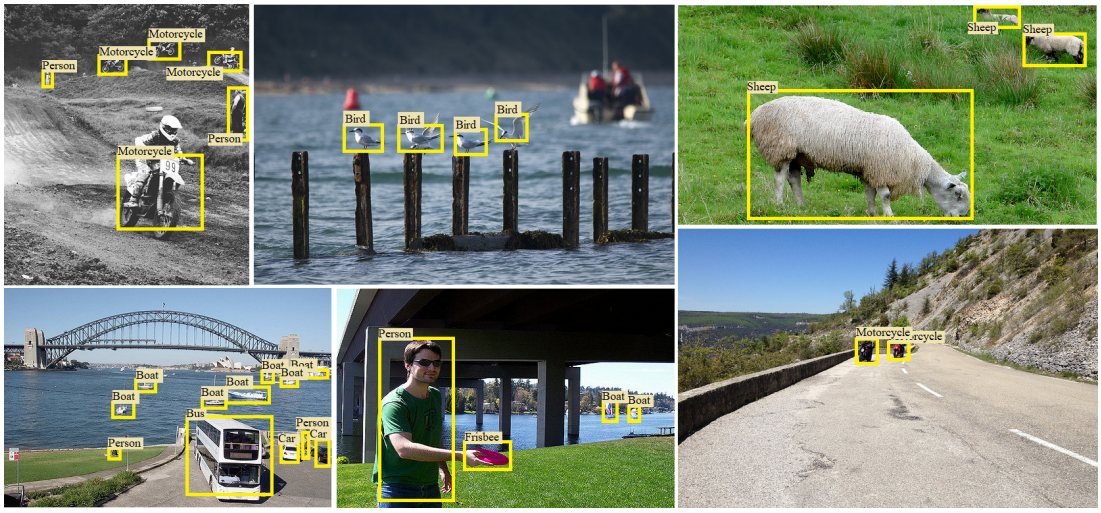

Figure 7.1: Proposed BackNet-based Faster RCNN is capable of detecting objects with large variations in scale within the same image.

traffic monitoring, and other smart applications, the demand for an accurate detection system is on the rise. Handcrafted features like SIFT [2, 175], SURF [3], HOG [176] and Deformable Part models [177, 178, 179] have long been used as conventional object detectors. The momentum has shifted towards CNN based detectors since the success of AlexNet [14] and other deep networks [10, 113, 26, 180] in a variety of tasks. SOTA detectors employ pretrained classification network as the detector backbone. Although classification networks have achieved superhuman level performance in a number of competitions, object detection still remains to be a much more difficult problem to solve [108, 105, 145]. There is a difference between the tasks of a detector and a classification network [108]. One distinct dissimilarity between the two is that classification datasets contain one prominent object per image, whereas multiple-class objects with large scale variations may appear within the same image in detection. An image with high resolution may contain extremely small objects which adds to the complexity.

Two-stage detectors like Faster RCNN host a pretrained classification network as the backbone and a Region Proposal Network (RPN). The last layer feature maps of the backbone network act as the input to the RPN. The spatial dimension of an input image is iteratively reduced in the intermediate convolution and pooling layers of a classification network. These layers are particularly helpful for keeping memory

requirement low and according to recent studies [181, 182], pooling also helps in ensuring the network's invariance to the small translation of the input. As a side effect of DS the original image, signals from small objects fades away in the intermediate layers and the deeper layer feature maps cannot retain enough information for these small objects. Consequently, detectors using such classification networks as the backbone struggle with small objects in the images.

One simple solution is to maintain high-resolution feature maps by avoiding several pooling/DS steps [108]. However, the loss of translation invariance has adverse effects on the detector's overall accuracy [181, 182]. Another solution is to incorporate shallower layer features with the deeper ones [102, 103, 104]. However, the shallow layer features lack in semantic meaning resulting in a weak set of feature maps when combined with the semantically strong deeper ones. In this chapter, we propose a novel backbone network that we call BackNet for Faster RCNN. We argue that it is extremely difficult for RPN to generate accurate object proposals for small objects if the feature maps fed to the RPN do not contain enough signals from those small objects. BackNet is based on the VGG16 network [113] with an additional subnetwork that maintains high-resolution feature maps up to the last layer. The final layer output of the VGG16 network is upsampled and merged with the corresponding subnetwork output to form a robust set of feature maps. These strong feature maps act as the RPN input and boost the overall detector performance (see Figure 7.1).

7.2 BackNet: Proposed Backbone Network

In this section, we first explain the impact of an effective backbone CNN in Faster RCNN architecture and then discuss the architecture and design rationales of BackNet.

7.2.1 Backbone Network

Starting from RCNN, it has taken a number of iterations to reach the upgraded variant Faster RCNN. Despite the evolutionary steps, the use of an ImageNet pretrained backbone network has been a constant. It highlights the important role a strong backbone network plays in the detection setup. The RPN takes the last layer feature maps

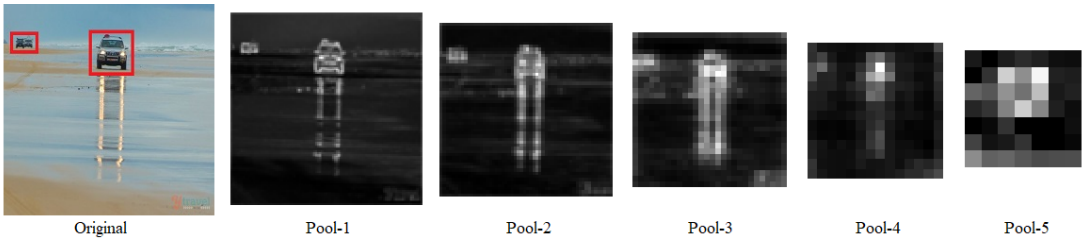

Figure 7.2: Feature maps are extracted after each of the five pooling layers in VGG16. The original image contains three instances of vehicle-one relatively large in the front and two smaller vehicles in the back (inside red box). It is intriguing to see that the large vehicle's activation signal can be found even in the deepest layer (Pool-5). However, activations from the small vehicles fade away with increasing depth. Pool-4 has very few weak activations and Pool-5 retains even lesser (almost nothing) information. Feature maps are enlarged from their original size for better visibility. Successive spatial dimension reductions from left to right is for illustration only (exact DS factor in VGG16 is 2.)

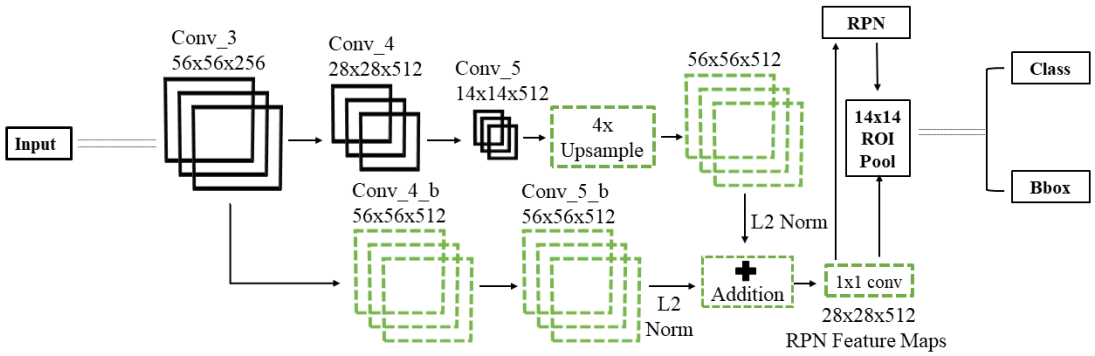

Figure 7.3: VGG16 is used as the base of our backbone network. BackNet is identical to VGG16 up to stage 3 (Conv_3), hence, stages 1-3 are not shown here. Dashed green boxes represent the additional operation introduced in BackNet. Solid boxes are the original components of a Faster RCNN network. Newly introduced subnetwork originating from Conv_3 maintains the same spatial resolution of 56×56 whereas the main network goes through multiple subsampling layers resulting in 14×14 feature maps. These Conv_5 feature maps go through a 4x upsampling layer and are merged with Conv_5_b feature maps. The merged feature map's spatial dimension is reduced to half and fed forward to the RPN. RPN yields object proposals and RoI pooling extracts feature from RPN input feature maps. Pooled features are warped to a fixed size (14×14) as they proceeds further inside the network.

produced by the backbone as input. The RPN then processes this input within its own network and outputs a set of object proposals. If the RPN is not supplied with strong feature maps containing ample signals from both small and large objects, it becomes extremely difficult for it to produce accurate object proposals. ImageNet classification networks are explicitly designed for optimum classification performance. Spatial dimension reduction through pooling and longer stride convolution are common characteristics for these networks. Unlike classification tasks where one prominent object is present per image, object detectors need to identify multiple object instances in the same image. The variety of object shape and scale adds to the complexity and the challenges here are far greater than classification. To design an effective backbone network, two factors should be considered.

1. Dimension reduction layers of a backbone network are important for translation invariance and efficient memory utilisation, but the loss of activation signals from small objects is a critical drawback for any detector.

2. A number of contemporary research works build feature pyramids by combining deep layer information with shallower ones. Even so, the semantic meaning of the shallower features is weak and combining these layers with deeper ones may not yield expected results.

7.2.2 BackNet Architecture

We consider VGG16 as the base of our backbone network with the introduction of an additional subnetwork originating from Conv_3 (see Figure 7.3). VGG16 generates a series of features at several stages (downsampled with a factor of 2 at each stage and layers producing feature maps of the same size are considered to be in the same network stage). VGG16 has five such stages where the original input of 224×224 is reduced to 14×14. We refer these five stages as Conv_1, Conv_2, Conv_3, Conv_4, and Conv_5.

We observed from our experimental studies that most of the small objects' ($\approx 30 \times 30$ pixels) signals start getting attenuated beyond Conv_3 stage (see Figure 7.2 for a visual interpretation). Therefore, a subnetwork originating from Conv_3 is introduced that

computes the convolution layers with the same configuration as the corresponding VGG16 layers (Figure 7.3). However, this subnetwork does not downsample the feature maps and maintains the same resolution throughout the rest of the network. On the other hand, Conv_3 feature maps in the VGG16 network are downsampled twice resulting in 14×14 feature maps in the Conv_5 stage. Adding Conv_5 feature maps with the output (Conv_5_b) of the subnetwork will form one set of merged features that contain activation signals from both small and large objects. Conv_5 feature maps are upsampled by a factor of 4 to match the spatial dimension of Conv_5_b before element-wise addition. Feature maps are upsampled using the two-dimensional cubic convolution interpolation function [183]. When (x, y) is a point in the rectangular subdivision $[x_j, x_{j+1}] \times [y_k, y_{k+1}]$, the two-dimensional cubic convolution interpolation function is expressed using Equation 7.1.

$$g(x,y) = \sum_{l=-1}^{2} \sum_{m=-1}^{2} c_{j+l,k+m} u(\frac{x - x_{j+l}}{h_x}) u(\frac{y - y_{k+m}}{h_y}) \tag{7.1}$$

where u is the one dimensional interpolation kernel and h_x, and h_y, are the x and y coordinate sampling increments. $C_{j,k}$'s are derived using Taylor's Expansion.

Both outputs from Conv_5 (upsampled) and Conv_5_b (subnetwork) are L2 Normalised before merging. Merged feature maps then undergo $1 \times 1 \times 512$ convolution with stride 2 resulting in $28 \times 28 \times 512$ feature maps; these act as the input to the RPN. We employ 14×14 RoI pooling rather than original 7×7 with a view to incorporating sufficient object features. The pooled features are propagated through to the regular Faster RCNN layers for a class label and a bounding box.

7.3 Performance Evaluation

In this section, we evaluate the performance of BackNet with five contemporary detectors based on a benchmark dataset. Table 7.1 compares different configurations of our proposed BackNet while Tables 7.2 and 7.3 illustrate how the BackNet-based Faster RCNN detector fares against other detectors for detecting objects with large scale difference. Table 7.2 focuses on the Precision and Table 7.3 focuses on the Recall values on the MS COCO dataset.

Table 7.1: Performance comparison among different configurations for the proposed BackNet. The detector performs best when the newly introduced subnetwork originates from Conv_3. Best Precision and Recall values are presented in Bold.

MS COCO											
Conv_3 (56×56)	Conv_4 (28×28)	2x Up-sample	4x Up-sample	B Norm	mAP	AP_S	AP_M	AP_L	AR_S	AR_M	AR_L
✓			✓		38.15	22.58	42.11	49.76	34.16	62.05	71.49
✓			✓	✓	**38.55**	**22.88**	**42.46**	50.31	**34.95**	**62.30**	72.85
	✓	✓		✓	37.45	19.90	41.77	**50.68**	30.01	59.26	**73.96**
	✓	✓			37.15	19.60	41.45	50.40	29.62	58.85	73.56

Table 7.2: Proposed BackNet based Faster RCNN improves Precision on small objects while maintaining stable performance on medium and large objects. Best Precision values are presented in Bold.

MS COCO				
Method	Mean Average Precision (mAP)	AP_S	AP_M	AP_L
Faster RCNN [101]	24.20	7.20	26.40	36.90
Faster RCNN + RS [185]	29.50	11.90	32.70	41.80
Faster RCNN + FPN [102]	35.80	17.50	38.70	47.80
ION [106]	30.70	11.82	32.78	44.80
DetNet [108]	38.20	22.05	42.10	**50.45**
BackNet-Faster RCNN	**38.55**	**22.88**	**42.46**	50.31

7.3.1 Dataset

We train and test our proposed method on MS COCO [184] dataset. It consists of 123,000 images which belong to a total of 80 object classes. We follow the standard split of 118,000/5,000 for training/validation. We use the standard metric of mean Average Precision (mAP) at Intersection over Union $IoU = 0.50 : 0.05 : 0.95$. Average Precision is also calculated for small (AP_S), medium (AP_M) and large (AP_L) objects separately with the specified IoU. According to MS COCO definition, objects with spatial dimension less than 32×32 are small, objects ranging from 32×32 to 96×96 are medium and objects with spatial dimension greater than 96×96 are considered large. Average Recall (AR) is computed at 1, 10 and 100 detections per image denoted by AR_1, AR_{10} and AR_{100} respectively.

Table 7.3: Proposed BackNet based Faster RCNN improves Recall on small objects while maintaining stable performance on medium and large objects. Best Recall values are presented in Bold.

MS COCO						
Method	AR_1	AR_{10}	AR_{100}	AR_S	AR_M	AR_L
Faster RCNN [101]	23.80	34.10	34.70	11.50	38.90	54.40
Faster RCNN + RS [185]	27.30	40.00	40.90	17.90	45.50	58.60
Faster RCNN + FPN [102]	30.90	46.30	47.90	26.40	52.40	63.00
ION [106]	27.70	42.80	45.40	23.00	50.08	63.02
DetNet [108]	32.00	58.33	49.35	28.80	52.10	67.15
BackNet-Faster RCNN	**32.40**	**59.64**	**56.70**	**34.95**	**62.30**	**72.85**

7.3.2 Experimental Setup

We use the ImageNet pretrained novel BackNet as the detector backbone. We follow the pragmatic four-step detector training scheme adopted in Faster RCNN[101]. First, the RPN is trained and it is allowed to output a maximum of 2,000 object proposals per image. Among these 2,000, 256 regions are randomly chosen to form a minibatch and calculate the loss. The RPN weights are initialised from a zero-mean Gaussian distribution with standard deviation of 0.01. In the second step, BackNet-based Fast RCNN network is trained using the RPN output (object proposals) from Step 1. In the third step, BackNet and RPN are trained together and the shared convolution layers along with the RPN layers are fine-tuned. In the last step, only the unique Fast RCNN layers are fine-tuned while the shared layer weights are frozen. Initial learning rate used for all four steps is 0.0001. SGD is used with a momentum value set to 0.9 and a weight decay of 0.0005 with adaptive dropout [55]. Minibatch size of one and a maximum of 40 epochs are used for each of the aforementioned training stages.

7.3.3 Quantitative Results Analysis

We experimented with different configurations of the proposed BackNet components. Table 7.1 provides a summary of the detectors overall performance on the MS COCO dataset. As stated earlier, VGG16 is used as the base network with upsampled last layer features. A new subnetwork is introduced which maintains the same feature map spatial dimension from its stage of origin. When the subnetwork originates from

Conv_3 stage (56×56), the detector performs the best both in terms of Precision and Recall; it scores an mAP of 38.55 and AR of 56.70. Note that this particular network configuration outperforms others on small object dataset with AP_S 22.88% and AR_S 34.95%. On the contrary, the subnetwork originating from Conv_4 stage improves the detector's performance on large objects (28×28), but both Precision and Recall drop on small objects. The reason can be explained by observing the lower resolution feature maps in Conv_4 (28×28) compared to Conv_3 (56×56). Some of the small object activations are lost when Conv_3 features are downsampled by a factor of 2 in the main branch and the Conv_4 origin subnetwork is also deprived of these activations. Consequently, the detector's performance does not degrade on large objects but its performance does degrade for smaller ones. Using Batch Normalisation before adding the upsampled and subnetwork feature maps is found to increase detector Precision (mAP) and Recall (AR) for both configurations.

Tables 7.2 and 7.3 compare the performance of the proposed and five contemporary object detectors in terms of Precision and Recall respectively. AR is calculated at 1, 10 and 100 detections per image denoted by AR_1, AR_{10} and AR_{100} respectively. AR_{100} is the mean Recall and mAP is the mean Precision across all object shapes available in the dataset. BackNet-based Faster RCNN achieves the highest mAP of 38.55% while securing an AP of 22.88% on small objects (second best DetNet 22.05%). Although DetNet produces the maximum AP for large objects (50.45%), our proposed method is comparable (50.31%). Since BackNet is capable of retaining both small and large object information in the final layer feature maps, the AR for BackNet-Faster RCNN is also better than other detectors on all three different sizes of objects.

7.4 Conclusion

Compared to the methods applied in the relevant literature to deal with the challenge posed by objects of small scale, our approach is theoretically straight forward and yields better results. The motif behind deploying the proposed BackNet subnetwork is to prevent the signals of small object from dying down across a number of down-sampling layers within a CNN. While the subnetwork takes care of the small objects, the regular objects cascade through the conventional VGG16 layers and the merging

layer still ensure a robust set of features inclusive of objects with all possible shapes and sizes. Although VGG16 is used as the base of the proposed method, BackNet can be incorporated with other conventional pretrained networks as well which further strengthens our contribution.

In the next chapter, we summarise our contributions presented throughout this thesis and discuss their compatibility with each other. We also provide potential directions for future research.

Chapter 8

Conclusion

Image classification is an important vision task with a diverge range of applications. CNNs have closed the gap between machine and human in image classification and off-the-shelf CNN classifiers are regularly used as the backbone network in more complex tasks including object detection and semantic segmentation. Therefore, any improvement in the backbone classifier, by extension, also improves performance in detection and segmentation. In this thesis, we aimed to develop a set of methods that improves the robustness of a CNN classifier on challenging datasets. Before we discuss the inter-compatibility of our methods, the key achievements of this thesis, along with their significance, are summarised below.

Improving Distortion Robustness with DCT Augmentation

In Chapter 3, we investigated CNN's vulnerability to a number of distortions including noise and blur. We showed that visually imperceptible distortion leads to misclassification even in SOTA CNNs. To this end, we proposed a distortion agnostic DCT augmentation method that can boost CNN's robustness against such challenges. Our method does not require any prior knowledge about the potential distortions which sets it apart from contemporary works. We also proposed an adaptive dropout based regularisation technique that prevents a CNN from overfitting to the clean training data. With the help of autoencoders, we also visually illustrated why and how DCT augmentation works.

Improving Distortion Robustness with LP-ReLU

In Chapter 4, we further improved CNN's robustness against distortion and perturbation by proposing a novel AF called LP-ReLU. Besides the primary objective

to introduce non-linearity in CNN, we cast LP-ReLU as a noise suppression unit by incorporating a low-pass filter in it. We also showed that DCT augmentation largely complements LP-ReLU in classifying distorted and perturbed images. At the end of Chapter 5, we proposed a novel method to visualise how CNNs draw classification boundaries. This strengthens our understanding behind CNN's vulnerability to imperceptible distortions and shows how our proposed methods improve robustness.

Making CNNs Invariant to Shift

In Chapter 5, we investigated the impact of signal aliasing in CNN and proposed a novel DS method called DAB-pool and AF called AA-ReLU. Our solution is based on the Nyquist sampling theorem which states that any signal should undergo low-pass filtering before DS to avoid aliasing. Aliasing largely impacts CNN's transformation invariance, especially against shift. Vulnerability to such trivial transformation can expose CNNs to adversarial attacks as well. Our proposed network with DAB-pool as the DS method and AA-ReLU as the AF can provide robustness and defence against such vulnerabilities.

Open Set Recognition

In Chapter 6, we highlighted conventional CNN's inability to deal with unknown samples. Since CNN classifiers are trained on a finite image set with an obligation to provide an output, such unknowns are bound to result in misclassification. To avoid such vulnerability, we proposed a novel training paradigm where our proposed OSRNet is trained using a novel KU data mining method.

Small Object Detection

In Chapter 7, we investigated deep detector's performance on detecting objects with large variation in scale. Our investigation showed that even SOTA detectors struggle to detect small objects as iterative DS in the intermediate layers wipe out the small object features. To this end, we proposed a novel backbone for Faster RCNN which can detect small objects with greater accuracy while maintaining stable performance on larger objects.

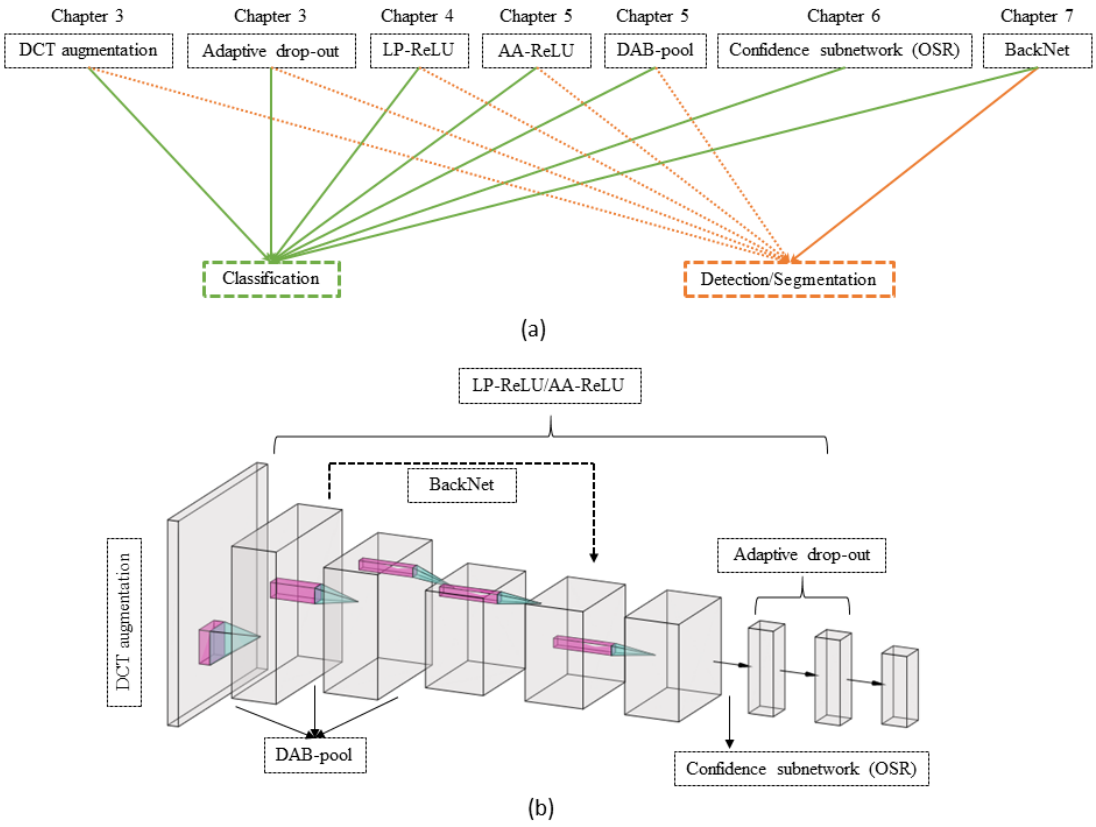

Figure 8.1: (a) Compatibility between proposed methods and tasks. (b) An illustration of a CNN hosting the proposed methods.

Relationship Among the Proposed Methods and Future Research Directions

Based on the requirements, most of the methods proposed in this thesis can be integrated in classification as well as detection and segmentation networks as depicted in Figure 8.1. Our proposed CS in OSRNet for OSR is the only method tailored for classification networks (see Figure 8.1(a)). This is because unlike classifiers, CNN detectors can abstain from making any prediction if there is no known object in an image and, hence, dealing with unknown class is not a major issue here. It is worth noting that in Figure 8.1, a solid line between a method and task pair indicates it has already been presented in the corresponding chapter of this thesis. A dashed line denotes a compatible pair but we leave further investigation of these pairs as future works.

One potential research direction could be investigating the impact of our novel

anti-aliasing units in detecting small objects. It would also be intriguing to study the role these units can play across diverse network architectures, such as the Transformers.

Deep learning based computer vision is seeing diverse applications coupled with mass adoption at an impressive rate. CNNs are at the core of this advancement and ensuring such powerful networks do not falter due to trivial challenges is of paramount importance. We believe our work on improving CNN's robustness on challenging datasets is a step towards further advancing the use of CNNs in more complex real-world applications.

Bibliography

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[4] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in brain research*, vol. 155, pp. 23–36, 2006.

[5] J. Hays and A. A. Efros, "Im2gps: estimating geographic information from a single image," in *2008 IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008, pp. 1–8.

[6] M. T. Hossain, S. W. Teng, D. Zhang, S. Lim, and G. Lu, "Enhancing the effectiveness of local descriptor based image matching," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2018, pp. 1–8.

[7] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *European Conference on Computer Vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[8] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision (IJCV)*, vol. 105, no. 3, pp. 222–245, 2013.

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

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 818–833.

[11] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning (ICML)*, 2010.

[13] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8609–8613.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[15] J. Li, T. Zhang, W. Luo, J. Yang, X.-T. Yuan, and J. Zhang, "Sparseness analysis in the pretraining of deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1425–1438, 2016.

[16] P. Dayan and L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[17] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning (ICML)*, vol. 30, no. 1, 2013, p. 3.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE*

International Conference on Computer Vision (ICCV), 2016, pp. 1026–1034.

[19] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[20] S. S. Liew, M. Khalil-Hani, and R. Bakhteri, "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems," *Neurocomputing*, vol. 216, pp. 718–734, 2016.

[21] A. Rozsa and T. E. Boult, "Improved adversarial robustness by reducing open space risk via tent activations," *arXiv preprint arXiv:1908.02435*, 2019.

[22] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5918–5926.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *International Conference on Learning Representations (ICLR)*, 2018.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[27] S. Zagoruyko and N. Komodakis, "Wide residual networks," *British Machine Vision Conference (BMVC)*, 2016.

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[29] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–7.

[30] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[31] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *International Conference on Learning Representations (ICLR)*, 2019.

[32] I. Vasiljevic, A. Chakrabarti, and G. Shakhnarovich, "Examining the impact of blur on recognition by convolutional networks," *arXiv preprint arXiv:1611.05760*, 2016.

[33] Y. Zhou, S. Song, and N.-M. Cheung, "On classification of distorted images with deep convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 1213–1217.

[34] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk, "Adversarial examples are a natural consequence of test error in noise," *International Conference on Machine Learning (ICML)*, 2019.

[35] E. Rusak, L. Schott, R. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel, "Increasing the robustness of dnns against image corruptions by playing the game of noise," *arXiv preprint arXiv:2001.06057*, 2020.

[36] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 113–123.

[37] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," in *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[38] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," *International Conference on Learning Representations (ICLR)*, 2020.

[39] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch gaussian augmentation," *arXiv preprint arXiv:1906.02611*, 2019.

[40] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 6023–6032.

[41] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *International Conference on Learning Representations (ICLR)*, 2017.

[42] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint:1708.04552*, 2017.

[43] S. F. Dodge and L. J. Karam, "Quality robust mixtures of deep neural networks," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5553–5562, 2018.

[44] S. Diamond, V. Sitzmann, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Optimizing image classification architectures for raw sensor data," *arXiv preprint arXiv:1701.06487*, 2017.

[45] J. Yim and K.-A. Sohn, "Enhancing the performance of convolutional neural networks on quality degraded datasets," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2017, pp. 1–8.

[46] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 15 637–15 648.

[47] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt, "Test-time training for out-of-distribution generalization," *arXiv preprint arXiv:1909.13231*, 2019.

[48] P. Benz, C. Zhang, A. Karjauv, and I. S. Kweon, "Revisiting batch normalization for improving corruption robustness," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[49] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," *Advances in Neural Information Processing Systems (NIPS)*, vol. 33, 2020.

[50] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[51] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations (ICLR)*, 2018.

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

[53] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM on Asia Conference on Computer and Communications Security (ACCS)*. ACM, 2017, pp. 506–519.

[54] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[55] M. T. Hossain, S. W. Teng, D. Zhang, S. Lim, and G. Lu, "Distortion robust image classification using deep convolutional neural network with discrete cosine transform," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 659–663.

[56] R. Geirhos, C. R. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, "Generalisation in humans and deep neural networks," in *Advances in Neural*

Information Processing Systems (NIPS), 2018, pp. 7538–7550.

[57] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt, "Measuring robustness to natural distribution shifts in image classification," *arXiv preprint arXiv:2007.00644*, 2020.

[58] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," *arXiv preprint arXiv:2006.16971*, 2020.

[59] R. Zhang, "Making convolutional networks shift-invariant again," *International Conference on Machine Learning (ICML)*, 2019.

[60] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *The Journal of Machine Learning Research (JMLR)*, 2018.

[61] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[62] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks (ICANN)*. Springer, 2010, pp. 92–101.

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

[64] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, 1983.

[65] X. Zou, F. Xiao, Z. Yu, and Y. J. Lee, "Delving deeper into anti-aliasing in convnets," in *British Machine Vision Conference (BMVC)*, 2020.

[66] O. S. Kayhan and J. C. v. Gemert, "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 14 274–14 285.

[67] Q. Li, L. Shen, S. Guo, and Z. Lai, "Wavelet integrated cnns for noise-robust image classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7245–7254.

[68] J. Ryu, M.-H. Yang, and J. Lim, "Dft-based transformation invariant pooling layer for visual classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 84–99.

[69] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Technical Report, University of Toronto*, 2010.

[70] K. Turkowski, "Filters for common resampling tasks," in *Graphics gems*. Academic Press Professional, Inc., 1990, pp. 147–165.

[71] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *2009 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 2146–2153.

[72] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 1757–1772, 2012.

[73] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, November 2014.

[74] E. Rudd, L. P. Jain, W. J. Scheirer, and T. Boult, "The extreme value machine," *in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI*, March 2018.

[75] H. Zhang and V. M. Patel, "Sparse representation-based open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2016.

[76] S. Dang, Z. Cao, Z. Cui, Y. Pi, and N. Liu, "Open set incremental learning for automatic target recognition," *IEEE Transactions on Geoscience and Remote Sensing*, 2019.

[77] Y. Li and L. Maguire, "Selecting critical patterns based on local geometrical and statistical information," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 1189–1201, 2010.

[78] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li, "Open set learning with counterfactual images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 613–628.

[79] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[80] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, "Generative openmax for multi-class open set classification," *in British Machine Vision Conference (BMVC)*, 2017.

[81] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4016–4025.

[82] P. Oza and V. M. Patel, "C2ae: Class conditioned auto-encoder for open-set recognition," *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[83] C. Geng, L. Tao, and S. Chen, "Guided cnn for generalized zero-shot and open-set recognition using visual and semantic prototypes," *Pattern Recognition*, pp. 107–263, 2020.

[84] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *International Conference on Learning Representations (ICLR)*, 2018.

[85] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke, "Out-of-distribution detection using an ensemble of self supervised leave-out classifiers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 550–564.

[86] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv:1702.06280*, 2017.

[87] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv:1610.02136*, 2016.

[88] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," in *International Conference on Learning Representations (ICLR)*, 2019, Journal Article.

[89] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 41–50.

[90] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *International Conference on Learning Representations (ICLR)*, 2017, Journal Article.

[91] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv:1802.04865*, 2018.

[92] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5764–5772.

[93] D. Mandal, S. Narayan, S. K. Dwivedi, V. Gupta, S. Ahmed, F. S. Khan, and L. Shao, "Out-of-distribution detection for generalized zero-shot action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9985–9993.

[94] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[95] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp.

7263–7271.

[96] J. Redmon and F. Ali, "Yolov3: An incremental improvement," *arXiv:1804.02767*, 2018.

[97] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision, ECCV*. Springer, 2016, pp. 21–37.

[98] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 142–158, 2016.

[99] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision (IJCV)*, vol. 104, no. 2, pp. 154–171, 2013.

[100] R. Girshick, "Fast rcnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[101] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[102] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125.

[103] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1281–1290.

[104] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu, "Scale-transferrable object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 528–537.

[105] B. Singh and L. S. Davis, "An analysis of scale invariance in object detection snip," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3578–3587.

[106] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2874–2883.

[107] W. Wang, B. Wu, J. Lv, and P. Dai, "Regular and small target detection," in *International Conference on Multimedia Modeling*. Springer, 2019, pp. 453–464.

[108] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Detnet: A backbone network for object detection," *arXiv preprint arXiv:1804.06215*, 2018.

[109] A.-K. Fattal, M. Karg, C. Scharfenberger, and J. Adamy, "Distant vehicle detection: How well can region proposal networks cope with tiny objects at low resolution?" in *European Conference on Computer Vision, ECCV*. Springer, 2018, pp. 289–304.

[110] Y. Gao, S. Guo, K. Huang, J. Chen, Q. Gong, Y. Zou, T. Bai, and G. Overett, "Scale optimization for full-image-cnn vehicle detection," in *in IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 785–791.

[111] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2129–2137.

[112] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision, ECCV*. Springer, 2016, pp. 354–370.

[113] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[114] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[115] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 86–94.

[116] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 1929–1958, 2014.

[117] M. Forouzanfar, H. A. Moghaddam, and M. Dehghani, "Speckle reduction in medical ultrasound images using a new multiscale bivariate bayesian mmse-based method," in *in 15th IEEE Signal Processing and Communications Applications, SIU 2007.*, 2007, pp. 1–4.

[118] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[119] Z. Wang, A. C. Bovik, and B. L. Evan, "Blind measurement of blocking artifacts in images," in *International Conference on Image Processing, (ICIP) Proceedings*, vol. 3. Ieee, 2000, pp. 981–984.

[120] J. Chou, M. Crouse, and K. Ramchandran, "A simple algorithm for removing blocking artifacts in block-transform coded images," in *International Conference on Image Processing (ICIP) Proceedings*, vol. 1. IEEE, 1998, pp. 377–380.

[121] A. K. Jain, *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[122] S.-C. Chan and K.-L. Ho, "Fast algorithms for computing the discrete cosine transform," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 3, pp. 185–190, 1992.

[123] J. Makhoul, "A fast cosine transform in one and two dimensions," *IEEE Transactions on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 28, no. 1, pp. 27–34,

1980.

[124] M. Narasimha and A. Peterson, "On the computation of the discrete cosine transform," *IEEE Transactions on Communications*, vol. 26, no. 6, pp. 934–936, 1978.

[125] C. M. Bishop, "Pattern recognition and machine learning (information science and statistics) springer-verlag new york," *Inc. Secaucus, NJ, USA*, 2006.

[126] K. P. Murphy, "Machine learning: A probabilistic perspective. adaptive computation and machine learning," 2012.

[127] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[128] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[129] S. Jetley, N. Lord, and P. Torr, "With friends like these, who needs adversaries?" in *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 10 749–10 759.

[130] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[131] S. Dube, "High dimensional spaces, deep learning and adversarial examples," *arXiv preprint arXiv:1801.00634*, 2018.

[132] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto, "Classification regions of deep neural networks," *arXiv:1705.09552*, 2017.

[133] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, 2008, pp. 1096–1103.

[134] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[135] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[136] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research (JMLR)*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[137] M. T. Hossain, S. W. Teng, F. Sohel, and G. Lu, "Robust image classification using a low-pass activation function and dct augmentation," *IEEE Access*, vol. 9, pp. 86 460–86 474, 2021.

[138] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, and M. Guo, "The many faces of robustness: A critical analysis of out-of-distribution generalization," *arXiv preprint arXiv:2006.16241*, 2020.

[139] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 3–14.

[140] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *In Proceedings ofthe 35th International Conference on Machine Learning, (ICML)*, July 2018.

[141] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le, "Smooth adversarial training," *arXiv preprint arXiv:2006.14536*, 2020.

[142] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations (ICLR)*, 2018.

[143] F. W. Campbell and J. G. Robson, "Application of fourier analysis to the visibility of gratings," *The Journal of physiology*, vol. 197, no. 3, p. 551, 1968.

[144] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[145] "Tiny imagenet visual recognition challenge," *URL https://tiny-imagenet.herokuapp.com/*, 2015.

[146] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 586–595.

[147] A. Bietti and J. Mairal, "Group invariance, stability to deformations, and complexity of deep convolutional representations," *The Journal of Machine Learning Research (JMLR)*, vol. 20, no. 1, pp. 876–924, 2019.

[148] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2016, pp. 1–6.

[149] J. Mohapatra, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, "Towards verifying robustness of neural networks against a family of semantic perturbations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 244–252.

[150] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning (ICML)*, 2019, pp. 1802–1811.

[151] L. Engstrom, "A rotation and a translation suffice: Fooling cnns with simple transformations," *arXiv preprint arXiv:1712.02779*, 2017.

[152] G. Sundaramoorthi and T. E. Wang, "Translation insensitive cnns," *arXiv preprint arXiv:1911.11238*, 2019.

[153] M. Manfredi and Y. Wang, "Shift equivariance in object detection," *arXiv preprint arXiv:2008.05787*, 2020.

[154] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[155] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[156] H. J. Keisler, *Elementary calculus: An infinitesimal approach*. Courier Corporation, 2013.

[157] P. Perera, V. I. Morariu, R. Jain, V. Manjunatha, C. Wigington, V. Ordonez, and V. M. Patel, "Generative-discriminative feature representations for open-set recognition," pp. 11 814–11 823, 2020.

[158] I. Goodfellow, "Generative adversarial networks," *Neural Information Processing Systems (NIPS) 2016*, 2016.

[159] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*, 2017.

[160] E. A. El-Sherif and S. Abdelazeem, "A two-stage system for arabic handwritten digit recognition tested on a new large database." in *Artificial Intelligence and Pattern Recognition*, 2007, pp. 237–242.

[161] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv:1608.07690*, 2016.

[162] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," 2010.

[163] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," *in Advances in Neural Information Processing Systems (NIPS)*, 2011.

[164] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 951–10 960.

[165] A. Blum, *Neural networks in C++: an object-oriented framework for building connectionist systems*. John Wiley & Sons, Inc., 1992.

[166] J. A. Swets, "The science of choosing the right decision threshold in high-stakes diagnostics." *American Psychologist*, 1992.

[167] M. H. Zweig and G. Campbell, "Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine." *Clinical chemistry*, p. 561, 1993.

[168] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[169] A. R. Dhamija, M. Günther, and T. Boult, "Reducing network agnostophobia," *Advances in Neural Information Processing Systems (NIPS)*, pp. 9157–9168, 2018.

[170] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[171] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding (CVIU)*, vol. 179, pp. 41–65, 2019.

[172] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," *NIPS*, 2017.

[173] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of Machine Learning Research (JMLR)*, vol. 2, no. Nov, pp. 67–93, 2001.

[174] M. T. Hossain, S. W. Teng, and G. Lu, "Backnet: An enhanced backbone network for accurate detection of objects with large scale variations," in *Pacific-Rim Symposium on Image and Video Technology*. Springer, 2019, pp. 52–64.

[175] S. W. Teng, M. T. Hossain, and G. Lu, "Multimodal image registration technique based on improved local feature descriptors," *Journal of Electronic Imaging*, vol. 24, no. 1, p. 013013, 2015.

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

[177] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester, "Discriminatively trained deformable part models," 2012.

[178] H. Azizpour and I. Laptev, "Object detection using strongly-supervised deformable part models," in *European Conference on Computer Vision*. Springer, 2012, pp. 836–849.

[179] N. Zhang, R. Farrell, F. Iandola, and T. Darrell, "Deformable part descriptors for fine-grained recognition and attribute prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 729–736.

[180] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[181] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[182] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*. Springer, 2010, pp. 92–101.

[183] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[184] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[185] X. Chen and A. Gupta, "An implementation of faster rcnn with study for region sampling," *arXiv:1702.02138*, 2017.