Robust Deep Learning Through Selective Feature Regeneration

by

Tejas S. Borkar

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

Approved July 2020 by the
Graduate Supervisory Committee:

Lina J. Karam, Chair
Pavan Turaga
Suren Jayasuriya
Cihan Tepedelenlioglu

ARIZONA STATE UNIVERSITY

August 2020

ABSTRACT

In recent years, the widespread use of deep neural networks (DNNs) has facilitated great improvements in performance for computer vision tasks like image classification and object recognition. In most realistic computer vision applications, an input image undergoes some form of image distortion such as blur and additive noise during image acquisition or transmission. Deep networks trained on pristine images perform poorly when tested on such distortions. DNN predictions have also been shown to be vulnerable to carefully crafted adversarial perturbations. Specifically, so-called universal adversarial perturbations are image-agnostic perturbations that can be added to any image and can fool a target network into making erroneous predictions. This work proposes selective DNN feature regeneration to improve the robustness of existing DNNs to image distortions and universal adversarial perturbations.

In the context of common naturally occurring image distortions, a metric is proposed to identify the most susceptible DNN convolutional filters and rank them in order of the highest gain in classification accuracy upon correction. The proposed approach called DeepCorrect applies small stacks of convolutional layers with residual connections at the output of these ranked filters and trains them to correct the most distortion-affected filter activations, whilst leaving the rest of the pre-trained filter outputs in the network unchanged. Performance results show that applying DeepCorrect models for common vision tasks significantly improves the robustness of DNNs against distorted images and outperforms other alternative approaches.

In the context of universal adversarial perturbations, departing from existing defense strategies that work mostly in the image domain, a novel and effective defense which only operates in the DNN feature domain is presented. This approach identifies pre-trained convolutional features that are most vulnerable to adversarial perturbations and deploys trainable feature regeneration units which transform these DNN filter activations into resilient

features that are robust to universal perturbations. Regenerating only the top 50% adversarially susceptible activations in at most 6 DNN layers and leaving all remaining DNN activations unchanged can outperform existing defense strategies across different network architectures and across various universal attacks.

To my loving parents and brother Varun.

ACKNOWLEDGMENTS

"You never fail until you stop trying." - Albert Einstein

My Ph.D. journey, like countless before me, has been one full of highs and lows and this thesis is the culmination of the hard work, sacrifices and support of many who stood by me through this entire journey and gave me the courage to never stop trying. This acknowledgment is my tribute to those who made vital contributions in bringing this thesis to fruition.

I would like to thank my Ph.D. advisor Dr. Lina Karam for giving me the opportunity to work on high impact problems, improving my technical writing skills, providing strong guidance and support throughout this thesis. I would like to thank the members of my Ph.D. committee, Dr. Pavan Turaga, Dr. Suren Jayasuriya and Dr. Cihan Tepedelenlioglu for their time and valuable suggestions that helped to improve the quality of this thesis. I would like to thank Farshad Akhbari and Intel Corporation for funding the first few years of my Ph.D. and also providing internship opportunities to work on challenging problems in computer vision and machine learning. I would like to thank my ECEE graduate advisor Lynn Pratte for the constant encouragement, help and support in navigating through the degree requirements.

I would like to express my utmost gratitude to my parents, elder brother Varun and sister-in-law Shreya for their hard work and countless sacrifices which laid the foundation for my academic success. Their unwavering faith and unconditional love has served as the strongest motivating force for me to keep pushing myself to be better. Words alone cannot capture the enormous contribution they have made to my life and by extension, this thesis. I would also like to thank my cousins Amita, Chetan and their respective families for their strong support and encouragement throughout my time at ASU.

I would like to especially thank my wonderful friends aka ”my Arizona family”: Vinay Kashyap, Charan Prakash, Aditee Shrotre, Vimala Bharadwaj, Shruthi Venkat, Parag Chandakkar, Nutan Dev and Aditya Kulkarni for constantly cheering me on and standing by my side through the numerous personal and professional setbacks I encountered over the course of this thesis. They say, ”true friendship reveals itself through adversity”; I couldn't agree more. Every obstacle we faced has only served to bring us closer and make us stronger. Each one of them at one point or another played the crucial role of a ”therapist” and gave me the chance and space to vent out my frustration/thoughts without feeling judged, with Vinay Kashyap shouldering this important responsibility in a majority of these cases. Each one of them is accomplished in their own right and I like to think that the best parts of each of their personalities rubbed off on me and nudged me in the right direction to be more humble and grounded in reality. The wonderful memories I forged with each one of them during my time at ASU is one of the highlights of my Ph.D. and time in Arizona.

As a Graduate Research Assistant in the IVUlab at ASU, I had the pleasure of working with amazingly talented peers like Sam Dodge, Alireza Golestaneh, Bashar Haddad, Milind Gide and I had many wonderful and encouraging discussions with each one of them.

As I now look back, I can't help but draw parallels between my Ph.D. journey and my area of research. If I as a Ph.D. student were the *learning algorithm*, a successful Ph.D. thesis were the *optimal minima point* of a loss function and my Ph.D. experience represented the journey of the *learning algorithm* through the loss landscape, then my family would be the *initializer* and my friends would be the *regularizer*. Any expert worth their salt will tell you that no *learning algorithm* can ever reach an *optimal minima point* without having the right *initializer* and *regularizer* guiding it, i.e., I could never have finished this thesis without the contributions of my family and my friends.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

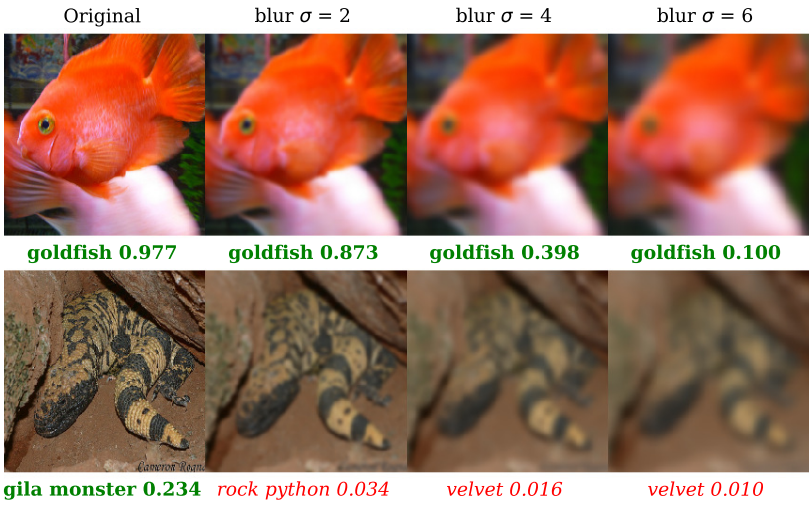
Chapter 1

INTRODUCTION

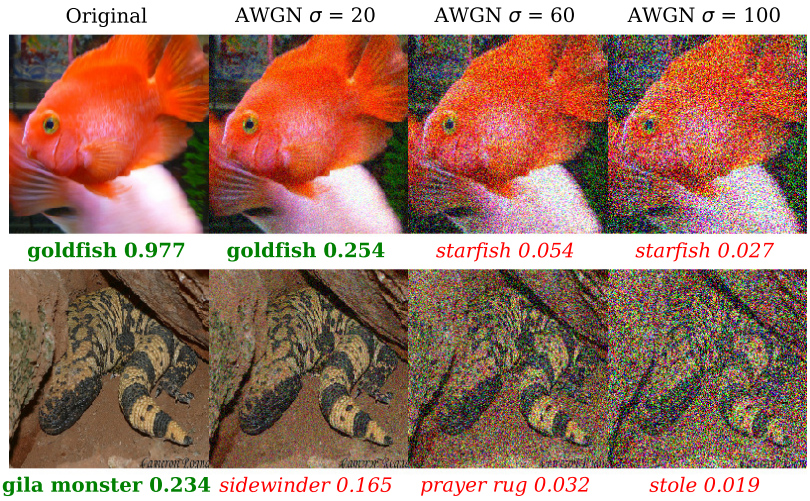
Today, state-of-the-art algorithms for computer vision tasks like image classification, object recognition and semantic segmentation employ some form of deep neural networks (DNNs). The ease of design for such networks, afforded by numerous open source deep learning libraries [11],[12], has established DNNs as the go-to solution for many computer vision applications. Even challenging computer vision tasks like image classification [10],[9],[7],[8] and object recognition [13],[14],[15], which were previously considered to be extremely difficult, have seen great improvements in their state-of-the-art results due to the use of DNNs. An important factor contributing to the success of such deep architectures in computer vision tasks is the availability of large scale annotated datasets [3],[16].

The visual quality of input images is an aspect very often overlooked while designing DNN based computer vision systems. In most realistic computer vision applications, an input image undergoes some form of image distortion including blur and additive noise during image acquisition, transmission or storage. However, most popular large scale datasets do not have images with such artifacts. Dodge and Karam [17] showed that even though such image distortions do not represent adversarial samples for a DNN, they do cause a considerable degradation in classification performance. Figure 1.1 shows the effect of image quality on the prediction performance of a DNN trained on high-quality images devoid of such distortions.

Testing distorted images with a pre-trained DNN model for AlexNet [8], we observe that adding even a small amount of distortion to the original image results in a misclassification (Figure 1.1), even though the added distortion does not hinder the human ability to classify the same images [18, 19]. In the cases where the predicted label for a distorted



(a)



(b)

Figure 1.1: Effect of image quality on DNN predictions, with predicted label and corresponding prediction confidence for top-1 class after softmax normalization, generated by a pre-trained AlexNet [8] model. Distortion severity increases from left to right, with the left-most image in a row having no distortion (original). Bold green text indicates correct classification, while italic red text denotes misclassification. (a) Examples from the ILSVRC-2012 [3] validation set distorted by Gaussian blur. (b) Examples from the ILSVRC-2012 validation set distorted by Additive White Gaussian Noise (AWGN).

image is correct, the prediction confidence drops significantly as the distortion severity increases. This indicates that features learnt from a dataset of high-quality images are not invariant to image distortion or noise and cannot be directly used for applications where the quality of images is different than that of the training images. Some issues to consider while deploying DNNs in noise/distortion affected environments include the following. For a network trained on undistorted images, are all convolutional filters in the network equally susceptible to noise or blur in the input image? Are networks able to learn some filters that are invariant to input distortions, even when such distortions are absent from the training set? Is it possible to identify and rank the convolutional filters that are most susceptible to image distortions and recover the lost performance, by only correcting the outputs of such ranked filters?

Similarly, DNNs also make erroneous predictions when a small magnitude, carefully crafted perturbation (adversarial perturbations) that is almost visually imperceptible to humans is added to an input image [20, 21, 22, 23, 24, 25, 26, 27, 28]. Furthermore, such perturbations have been successfully placed in a real-world scene via physical adversarial objects [29, 30, 27], thus posing a security risk.

Adversarial attacks can be divided into two major categories: 1) White-box attacks, where the adversary has complete knowledge (i.e., access to gradients, trained parameters, network architecture) of the targeted DNN and 2) Black-box attacks, where the adversary has limited knowledge (i.e., access to only network predictions, limited queries for DNN parameters and network architecture) or no knowledge of the baseline DNN being targeted. White-box attacks are usually much stronger than black-box attacks due to less constraints on the adversary to generate an attack. Most existing adversarial attacks use target network gradients (white-box) to construct an image-dependent adversarial example [20, 21, 27, 25, 28, 23] that has limited transferability to other networks or images [20, 31, 32]. Other methods to generate image-dependent adversarial samples include ac-

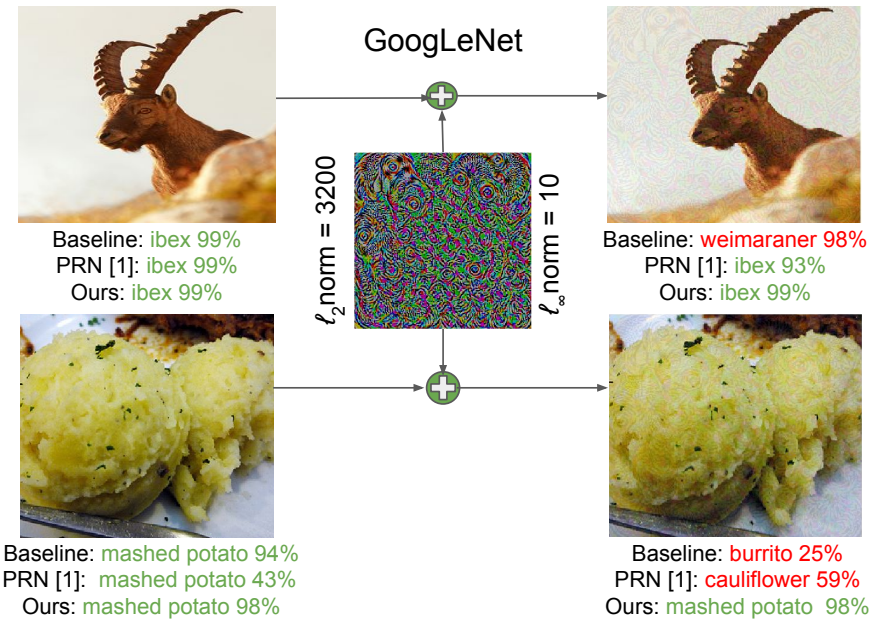


Figure 1.2: DNN label predictions and confidence score for different images perturbed with the same universal adversarial perturbation. A DNN with no defense (Baseline) misclassifies perturbed images (right) with high confidence (red text) even when it classifies clean images (left) correctly (green text). The proposed defense (Ours) correctly classifies both perturbed and clean images with high confidence.

cessing only the network predictions [33, 34, 35] (black-box), using surrogate networks [26] and gradient approximation [36]. Although there is significant prior work on adversarial defenses such as adversarial training [20, 21, 22, 37], ensemble training [38], randomized image transformations and denoising [39, 40, 41, 42, 40, 43, 41, 44, 45] and adversarial sample rejection [46, 47, 48, 49, 50], a DNN is still vulnerable to adversarial perturbations added to a non-negligible portion of the input [36, 51]. These defenses mostly focus on making a DNN robust to image-dependent adversarial perturbations which are less likely to be encountered in realistic vision applications [52, 2].

Unlike the aforementioned image-dependent adversarial attacks, universal adversarial attacks [1, 53, 54, 55, 56, 2, 57, 58, 59] construct a single *image-agnostic* perturbation

that, when added to any unseen image, fools DNNs into making erroneous predictions with very high confidence (Figure 1.2). These universal perturbations are also not unique and many adversarial directions may exist in a DNN's feature space [60, 61, 62]. Furthermore, universal perturbations generated for one DNN can transfer to other DNNs, making them *doubly universal* [1]. Such *image-agnostic* perturbations pose a strong realistic threat model [2] for many vision applications since perturbations can easily be pre-computed and then inserted in real-time (e.g., in the form of a printed adversarial patch or sticker) into any scene [63, 64]. For example, while performing semantic segmentation, such *image-agnostic* perturbations can completely hide a target class (i.e., pedestrian) in the resulting segmented scene output and adversely affect the braking action of an autonomous car [65].

This work proposes two approaches to improve the robustness of existing pre-trained DNNs to image distortions and universal adversarial perturbations, respectively. The first method which is called *DeepCorrect*, corrects distortion-affected DNN feature maps to improve classification performance under various image distortions. The second proposed method defends against universal adversarial attacks by regenerating select DNN features such that the resulting DNNs are resilient to universal adversarial perturbations in the input.

1.1 Contributions

In the context of DNN robustness to image distortions, the following novel contributions are proposed:

- Evaluating the effect of common image distortions like Gaussian blur and Additive White Gaussian Noise (AWGN) on the outputs of pre-trained convolutional filters. It is observed that for every layer of convolutional filters in the DNN, certain filters are

more susceptible to input distortions than others and that correcting the activations of these filters can help recover lost performance.

- Measuring the susceptibility of convolutional filters to input distortions and ranking filters in the order of highest susceptibility to input distortion.

- Correcting the worst distortion-affected filter activations by appending *correction units*, which are small blocks of stacked convolutional layers trained using a target-oriented loss, at the output of select filters, whilst leaving the rest of the pre-trained filter outputs in a DNN unchanged.

- Representing full-rank convolutions in *DeepCorrect* models with rank-constrained approximations consisting of a sequence of separable convolutions with rectangular kernels to mitigate the additional computational cost introduced by *correction units*. This results in pruned *DeepCorrect* models that have almost the same computational cost as the respective pre-trained DNN, during inference.

Similarly, DNN robustness to universal adversarial perturbations [1, 54, 53, 55, 56, 2] is improved through the following novel contributions:

- Showing the existence of a set of vulnerable convolutional filters, that are largely responsible for erroneous predictions made by a DNN in an adversarial setting and that the ℓ_1-norm of the convolutional filter weights can be used to identify such filters.

- Unlike, existing image-domain defenses, the proposed DNN feature domain-based defense uses trainable *feature regeneration units*, which regenerate activations of the aforementioned vulnerable convolutional filters into resilient features (adversarial noise masking). Extensive evaluation of the proposed defense on a variety of DNN architectures which shows that the proposed defense outperforms all other existing defenses [52, 40, 37, 45, 22, 2].

- A fast method is proposed to generate strong synthetic adversarial perturbations for training.

- Without any additional attack-specific training, the defense trained on one type of universal attack [1] effectively defends against other different unseen universal attacks [53, 54, 55, 2, 56, 58] and this is the first work in the literature to show such broad generalization across different universal attacks.

1.2 Organization

This thesis is organized as follows. Chapter 2 discusses the existing work and their drawbacks, followed by a brief overview of the necessary technical concepts that are used in this thesis. Chapter 3 describes *DeepCorrect*, a novel method that improves DNN robustness to image distortions by identifying and correcting the most distortion-affected DNN activation maps. *DeepCorrect* is extensively validated with different DNN architectures and multiple datasets covering image classification, object detection and scene classification. Chapter 4 describes a selective DNN feature regeneration defense that effectively defends against universal adversarial attacks, followed by evaluation of defense performance against prior art. Chapter 5 summarizes the novel contributions of this work and highlights directions for future research.

Chapter 2

BACKGROUND

This chapter provides an overview of concepts and prior work that is useful in understanding the proposed methods that are described in subsequent chapters of this thesis. Section 2.1 provides an overview of the prior work in assessing and improving the robustness of DNNs to image distortions. Section 2.2 discusses the prior work in addressing the susceptibility of DNNs to adversarial attacks. Common DNN architectures used for image classification are described in Section 2.3. Section 2.4 presents a survey of image-based datasets used for evaluation in this thesis. Sections 2.5 and 2.6 provide, respectively, a detailed overview of the image distortions and adversarial threat model used in this thesis.

2.1 Effect of Image Quality on DNNs

DNNs have recently been shown to be susceptible to a carefully crafted small magnitude image perturbation, visually imperceptible to humans (adversarial noise), which when added to an image causes DNNs to misclassify the image even when the original perurbation-free version of such an image is correctly classified with high confidence [20, 21, 23, 25, 26]. Proposed defenses against such adversarial attacks include: 1) *adversarial training* [21], which augments adversarial examples in the training stage of a baseline DNN, 2) transformation-based methods which utilize image denoising and pixel rearrangement to improve robustness (e.g., [40]) and 3) defenses that detect the presence of adversarial perturbations in the input and abstain from making a prediction (e.g., [46]). Such defenses are only designed to work effectively against a small magnitude noise perturbation (i.e., visually imperceptible or quasi-perceptible) and have not been tested against visually perceptible image quality degradations such as those encountered in sensing and transmis-

sion. Conversely, the concept of rubbish samples proposed in [66] studies the vulnerability of DNNs to making arbitrary high confidence predictions for random noise images that are completely unrecognizable to humans, i.e., the images contain random noise and no actual object. However, both adversarial samples and rubbish samples are relatively less encountered in common computer vision applications as compared to other common distortions due to image acquisition, storage, transmission and reproduction.

Karam and Zhu [67] present QLFW, a face matching dataset consisting of images with five types of quality distortions. Basu *et al.* [68] present the n-MNIST dataset, which adds Gaussian noise, motion blur and reduced contrast to the original images of the MNIST dataset. Dodge and Karam [17] evaluate the impact of a variety of quality distortions such as Gaussian blur, AWGN and JPEG compression on various state-of-the-art DNNs and report a substantial drop in classification accuracy on the ImageNet (ILSVRC-2012) dataset in the presence of blur and noise. A similar evaluation for the task of face recognition is presented in [69].

Rodner *et al.* [70] assess the sensitivity of various DNNs to image distortions like translation, AWGN and salt & pepper noise, for the task of fine grained categorization on the CUB-200-2011 [71] and Oxford flowers [72] datasets, by proposing a first-order Taylor series based gradient approximation that measures the expected change in final layer outputs for small perturbations to input image. Since a gradient approximation assumes small perturbations, Rodner *et al.*'s sensitivity measure does not work well for higher levels of distortion as shown by [70] and does not assess susceptibility at a filter level within a DNN. Furthermore, Rodner *et al.* do not present a solution for making the network more robust to input distortions; instead, they simply fine-tune the whole network with the distorted images added as part of data augmentation during training. Retraining large networks such as VGG16 [10] or ResNet-50 [7] on large-scale datasets is computationally expensive. Unlike Rodner *et al.*'s work, the proposed ranking measure (Chapter 3) assesses sensitivity of in-

dividual convolutional filters in a DNN, is not limited to differentiable distortion processes, and holds good for both small and large perturbations.

Zheng *et al.* [73] improve the general robustness of DNNs to unseen small perturbations in the input image through the introduction of distortion agnostic *stability training*, which minimizes the KL-divergence between DNN output predictions for a clean image and a noise perturbed version of the same image. The perturbed images are generated by adding uncorrelated Gaussian noise to the original image. *Stability training* provides improved DNN robustness against JPEG compression, image scaling and cropping. However, the top-1 accuracy of *stability trained* models is lower than the original model, when tested on most distortions including motion blur, defocus blur and additive noise among others [74]. Sun *et al.* also propose a distortion agnostic approach to improve DNN robustness by introducing 3 feature quantization operations, i.e., a floor function with adaptive resolution, a power function with learnable exponents and a power function with data dependent exponents, that act on the convolutional filter activations before the ReLU non-linearity is applied. However, similar to *stability training*, the performance of *feature quantized* models is lower than the original model, when tested on distortions like defocus blur and additive noise. Additionally, no single feature quantization function consistently outperforms the other two for all types of distortion. Although both distortion agnostic methods [73], [74] improve DNN robustness, their top-1 accuracy is much lower than the accuracy of the original DNN on distortion free images, making it difficult to deploy these models.

A non-blind approach to improve the resilience of networks trained on high quality images would be to retrain the network parameters (fine-tune) on images with observed distortion types. Vasiljevic *et al.* [75] study the effect of various types of blur on the performance of DNNs and show that DNN performance for the task of classification and segmentation drops in the presence of blur. Vasiljevic *et al.* [75] and Zhou *et al.* [76] show

that fine-tuning a DNN on a dataset comprised of both distorted and undistorted images helps to recover part of the lost performance when the degree of distortion is low.

Diamond *et al.* [77] propose a joint denoising, deblurring and classification pipeline. This involves an image preprocessing stage that denoises and deblurs the image in a manner that preserves image features optimal for classification rather than aesthetic appearance. The classification stage has to be fine-tuned using distorted and clean images, while the denoising and deblurring stages assume a priori knowledge of camera parameters and the blur kernel, which may not be available at the time of testing.

2.2 DNN Susceptibility to Adversarial Attacks

Adversarial training (Adv. tr.) [20, 21, 22] has been shown to improve DNN robustness to image-dependent adversarial attacks through augmentation, in the training stage, with adversarial attack examples, which are computed on-the-fly for each mini-batch using gradient-ascent to maximize the DNN's loss. The robustness of adversarial training to black-box attacks can be improved by using perturbations computed against different target DNNs that are chosen from an ensemble of DNNs [38]. Kannan *et al.* [78] scale adversarial training to ImageNet [3] by encouraging the adversarial loss to match logits for pairs of adversarial and perturbation-free images (logit pairing) but this latter method fails against stronger iterative attacks [79]. In addition to adversarially training the baseline DNN, prior works ([37], [80]) further improved DNN robustness to image-dependent attacks by denoising intermediate DNN feature maps, either through a non-local mean denoiser (feature denoising [37]) or a denoising auto-encoder (fortified nets [80]). Although Xie *et al.* report effective robustness against a strong PGD attack [22] evaluated on ImageNet [3], the additional non-local mean denoisers only add a 4% improvement over a DNN trained using standard adversarial training.

Image-domain defenses mitigate the impact of adversarial perturbations by utilizing non-differentiable transformations of the input such as image compression [41, 81, 44], frequency domain denoising [43] and image quilting and reconstruction [39, 42] etc. However, such approaches introduce unnecessary artifacts in clean images resulting in accuracy loss [52][40]. Prakash *et al.* [40] propose a two-step defense that first performs random local pixel redistribution, followed by wavelet denoising. Liao *et al.* [45] append a denoising autoencoder at the input of the baseline DNN and train it using a reconstruction loss that minimizes the error between higher layer representations of the DNN for an input pair of clean and denoised adversarial images (high level guided denoiser). Another popular line of defenses explores the idea of first detecting an adversarially perturbed input and then either abstaining from making a prediction or further pre-processing adversarial input for reliable predictions [46, 47, 48, 49, 50].

All of the aforementioned defenses are geared towards image-specific gradient-based attacks and none of them has, as of yet, been shown to defend against *image-agnostic* attacks. Initial attempts at improving robustness to universal attacks involved modelling the distribution of such perturbations [1, 82, 83], followed by model fine-tuning over this distribution of universal perturbations. However, the robustness offered by these methods has been unsatisfactory [2, 1] as the retrained network ends up overfitting to the small set of perturbations used. Extending adversarial training for image-dependent attacks to universal attacks has been attempted in [2] and [84]. Ruan and Dai [85] use additional shadow classifiers to identify and reject images perturbed by universal perturbations. Akhtar *et al.* [52] propose a defense against the universal adversarial perturbations attack (UAP) [1], using a detector which identifies adversarial images and then denoises them using a learnable *Perturbation Rectifying Network* (PRN).

2.3 DNN Architectures for Image Classification

Some of the DNN architectures commonly used in image recognition tasks and also used for evaluation in this thesis are described in this section.

2.3.1 AlexNet

AlexNet [8] is an 8-layered DNN that achieved the highest top-1 accuracy ($\approx 58\%$) on the ImageNet visual recognition challenge in 2012, with more than a 20% improvement in accuracy over prior state-of-the-art methods. The AlexNet DNN was the first convolutional DNN proposed for large scale image recognition, consisting of 5 convolutional layers and 3 fully-connected dense layers resulting in 61 million trainable parameters operating on images of size 256×256 pixels, as shown in Fig. 2.1a. Every convolutional layer is followed by a ReLU non-linearity for the AlexNet DNN. In addition to the ReLU non-linearity, the first and second convolutional layers of the AlexNet DNN are also followed by a local response normalization operation [8]. A major breakthrough of the AlexNet DNN was the successful implementation, both in terms of GPU memory used and training convergence, of large convolutional kernels (e.g., 11×11 and 5×5). Additionally certain convolutional layers were split across two GPUs for efficient training and memory utilization [8]. CaffeNet [11] is a DNN model derived from AlexNet [8] that achieves a similar accuracy but does not split any of its convolutional layers. VGG-F [86] is another DNN model derived from AlexNet [8] with the same depth (i.e., 8 layers) but different number of convolutional filters in each layer as compared to those of AlexNet (e.g., conv1 layer of AlexNet has 96 filters whereas conv1 layer of VGG-F has only 64). VGG-F uses lesser trainable parameters (easier to train) than AlexNet and still achieves a better accuracy than AlexNet.

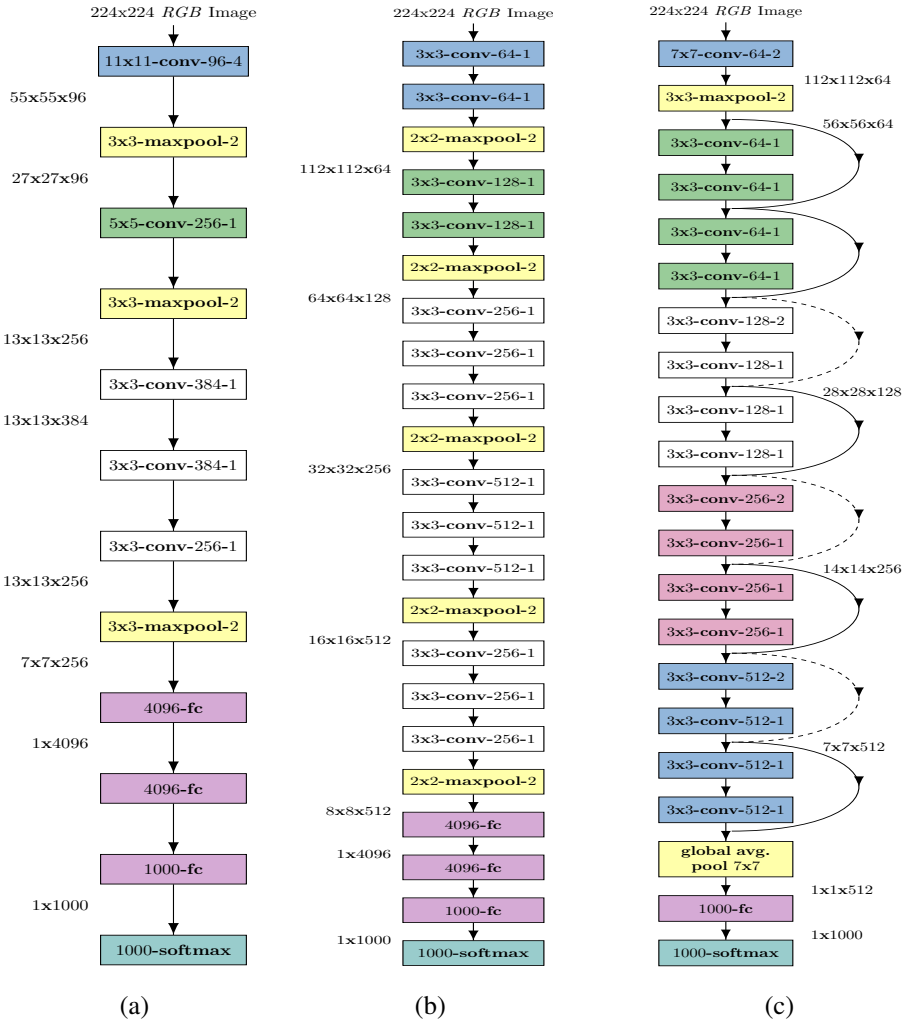


Figure 2.1: Network architectures for common DNN models used in large scale image recognition tasks. Convolutional layers are parameterized by *k*x*k*-conv-*d*-*s*, where *k*x*k* is the spatial extent of the filter, *d* is the number of output filters in a layer and *s* represents the filter stride. Maxpooling layers are parameterized as *k*x*k*-maxpool-*s*, where *s* is the spatial stride. (a) AlexNet [8]; (b) VGG-16 [10]; (c) ResNet18 [7].

2.3.2 VGG-16

VGG-16 [10], shown in Fig. 2.1b, is a 16-layered (double the depth of AlexNet) DNN with 144 million trainable parameters that significantly outperformed (by almost 20%) the previous DNNs like AlexNet [8] on image recognition tasks. Before VGG-16, it was common practice to use convolutional kernels with large spatial receptive fields such as 11×11, 7×7 and 5×5. Such large kernels required more computational memory and resulted in shallower networks. VGG-16 made a breakthrough contribution in DNN model architecture by showing that it was feasible to construct deeper DNNs by replacing all large convolutional kernels with stacks of small 3×3 kernels, resulting in lesser trainable parameters and memory requirements per layer. Simonyan and Zisserman [10] showed that deeper nets made by stacking several 3×3 convolutional kernels greatly improved accuracy and was the natural design choice for building DNNs. Both VGG-F [86] and VGG16 [10] have been developed at the Visual Geometry Group (VGG) at University of Oxford.

2.3.3 Residual Networks

In the area of deep learning, especially in image-based recognition tasks, there is a broad consensus on the idea that going deeper (more layers in a DNN) improves recognition accuracy. Starting from 8 layers [8] in 2012, DNN depth had already doubled [10] (16 layers) and even tripled [9] (27 layers) by the end of 2014, resulting in a top-1 accuracy of almost 76% on the ImageNet image recognition challenge. However, researchers were unable to train deeper DNNs (top-1 accuracy started decreasing after a certain depth). This was due to a combination of vanishing gradients as well as exploding gradients generated by backpropogating the classification loss function's gradients from the DNN output to the input through successive non-linearities (e.g., ReLU) [7]. Residual networks (ResNets) were proposed as a way to circumvent the issue of vanishing or exploding gradi-

ents by adding an identity skip connection across multiple convolutional layers (Fig. 2.1c, ResNet18), and to facilitate the backpropogation of network gradients from the output to the input. Every convolutional layer is followed by a batch normalization operation and a ReLU non-linearity for the ResNet18 model. Skip connections and residual feature maps are combined through an element-wise addition. Dashed-line skip connections perform an identity mapping using a stride of 2 to reduce feature map size and pad zero entries along the channel axis to increase dimensionality [7]. The inclusion of identity skip-connections was yet another breakthrough contribution in deep learning research which made it feasible to successfully train very deep networks (i.e., 152 layers) and is a default choice for designing any new DNN. In this thesis, ResNet18 (deep, 18 layers) and ResNet152 (very very deep, 152 layers) are used during evaluation.

2.4 Existing Image-based Datasets

The datasets used in this thesis are described in this section.

2.4.1 ImageNet (ILSVRC-2012)

The ImageNet (ILSVRC-2012) dataset is the largest publicly available annotated dataset for image recognition. ImageNet [3] has images of varying resolution, usually greater than 224x224 pixels consisting of around 1.3 million training images covering 1000 object classes and 50000 validation images, with 50 validation images per class. Fig. 2.2 shows sample images from the ILSVRC-2012 dataset, highlighting the large visual diversity in its images.

2.4.2 Caltech-101 and Caltech-256

The Caltech-101 [4] dataset consists of 101 object classes with a minimum of 40 images per class. Similarly, the Caltech-256 [5] dataset consists of 256 object classes with a

Figure 2.2: Sample images from the ILSVRC-2012 dataset [3].

Figure 2.3: Sample images from the Caltech-101 [4] and Caltech-256 [5] object detection datasets.

minimum of 80 images per class. Prior to ImageNet [3], the Caltech datasets were commonly used to evaluate object detection and image recognition methods. Due to the large scale and size of images in the ImageNet dataset, most visual recognition algorithms are first evaluated and benchmarked on the Caltech datasets. Fig. 2.3 shows sample images from the Caltech-101 [4] and Caltech-256 [5] datasets. The Caltech datasets share some common object classes with the ImageNet dataset.

2.4.3 SUN-397

Unlike the task of object recognition, where the aim is to classify the main object in the image, the goal of scene recognition is to classify the entire scene of the image. The SUN-397 [6] dataset consists of 397 scene categories with at least 100 images per class. Unlike

Figure 2.4: Sample images from the SUN-397 [6] scene recognition dataset.

object recognition datasets like Caltech-101 and Caltech-256, which bear some similarity to an image classification/object recognition dataset like ImageNet, a scene recognition dataset like SUN-397 bears no similarity to the ImageNet dataset. Fig. 2.4 shows sample images from the SUN-397 scene classification dataset.

2.5 Image Distortions

The different naturally occurring image distortions used for evaluating the effectiveness of *DeepCorrect* are described in this section.

2.5.1 Additive White Gaussian Noise

Additive White Gaussian noise (AWGN) is commonly used to model additive noise encountered during image acquisition and transmission. Throughout different evaluations in this thesis, a noise standard deviation $\sigma_n \in \{10, 20, 40, 60, 80, 100\}$ is used for AWGN. All input images are assumed to have a pixel intensity value in the range [0, 255]. Column 2 in Fig. 2.5 shows visual examples of images affected by AWGN distortions.

2.5.2 Gaussian Blur (G. Blur)

Gaussian blur, often encountered during image acquisition and compression [87], represents a distortion that eliminates high frequency discriminative object features like edges and contours. Throughout different evaluations in this thesis, a blur standard deviation $\sigma_b \in \{1, 2, 3, 4, 5, 6\}$ is used for Gaussian blur. The size of the blur kernel is set to 4 times the value of σ_b. Row 1 in Fig. 2.6 visualizes the 6 different Gaussian blur kernels used in this thesis. Column 3 in Fig. 2.5 shows visual examples of images affected by Gaussian blur.

2.5.3 Camshake Blur (Cam. Blur)

For simulating camera shake blur, the recently proposed method in [88] is used to generate random blur kernels. Rows 4-8 in Fig. 2.6 visualize 50 randomly generated blur kernels. Column 4 in Fig. 2.6 shows visual examples of images affected by camera shake blur.

2.5.4 Defocus Blur (D. Blur)

Spatially uniform disk kernels of varying radii are used to simulate defocus blur as outlined by Vasiljevic *et al.* [75]. The different blur strengths of disk kernels are generated by using a kernel disk radius $r_d \in \{2, 4, 6, 8, 10, 12\}$. Although such uniform disk kernels are typically not referred to as defocus blur kernels, the term "defocus blur" is used only to maintain consistency with the work of Vasiljevic *et al.* [75]. Row 2 in Fig. 2.6 shows examples of the different disk kernels used. Column 5 in Fig. 2.6 shows visual examples of images affected by defocus blur.

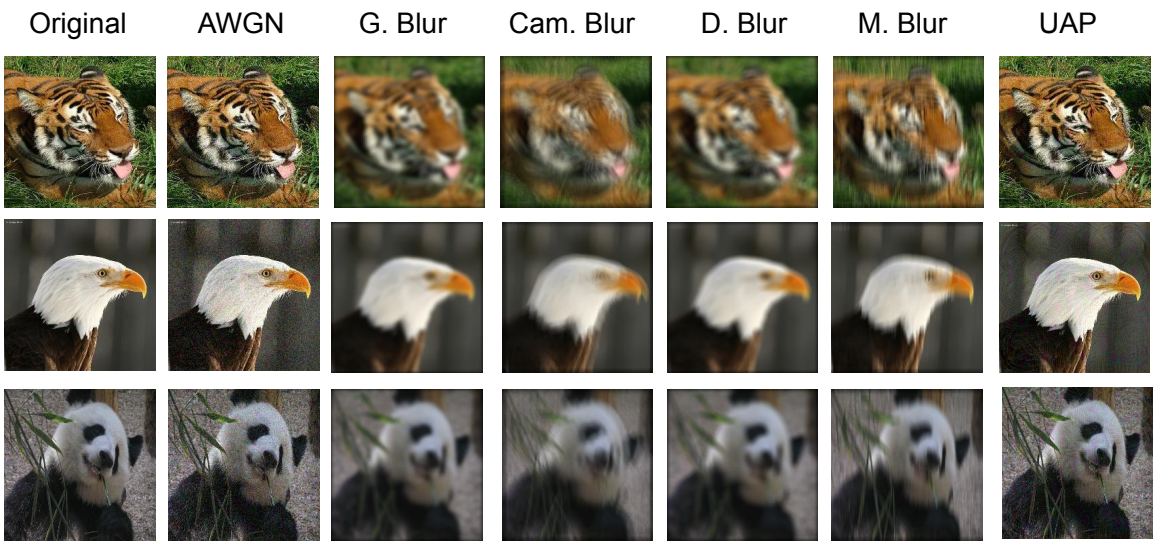


Figure 2.5: Visualization of image distortions and adversarial perturbations on input images from ILSVRC-2012. Column 1 shows original images devoid of any image distortions or perturbations. Columns 2-7, show visual examples of images distorted/perturbed with AWGN, Gaussian blur, camshake blur [88], defocus blur [75], uniform motion blur [75] and universal adversarial perturbations.

2.5.5 Motion Blur (M. Blur)

Motion blur is simulated by using horizontal and vertical box kernels of single pixel width and varying length (uniform linear motion) [75]. Three kernels each of different length are used for representing both uniform horizontal motion and vertical motion, respectively, with the kernel length $l_m \in \{6, 10, 14\}$. Row 3 in Fig. 2.6 shows the different horizontal and vertical box kernels used to simulate uniform motion blur. Column 6 in Fig. 2.6 shows visual examples of images affected by uniform motion blur.

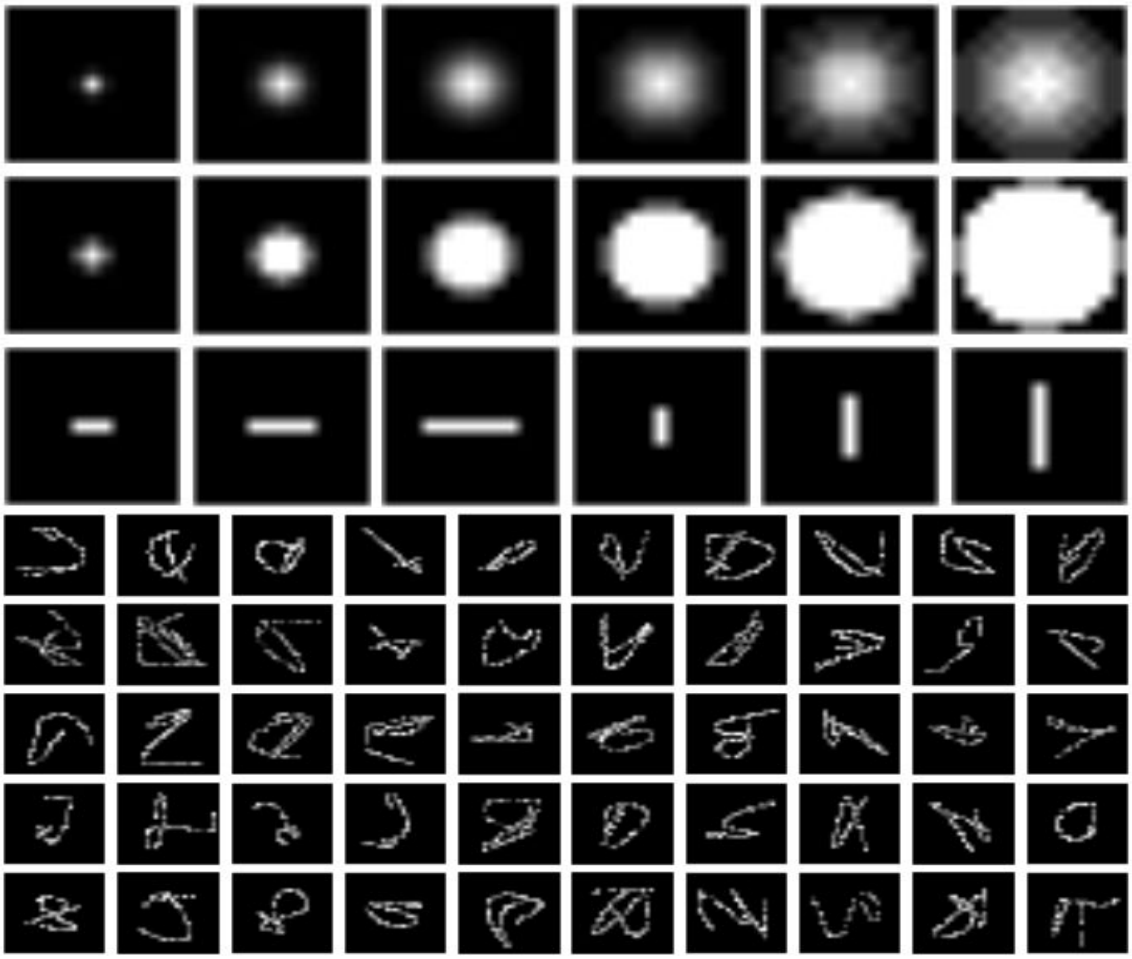


Figure 2.6: Visualization of blur kernels. Row 1 shows 6 different Gaussian blur kernels increasing in blur strength from left to right. Row 2 shows 6 different defocus blur (disk blur) kernels increasing in blur strength from left to right [75]. Row 3 shows different motion blur kernels with the first 3 representing horizontal motion and the next three showing vertical motion [75]. Rows 4-8 show 50 randomly generated camera shake kernels generated using the method outlined in [88].

2.6 Universal Adversarial Threat Model

Let μ_c represent the distribution of clean (unperturbed) images in $\mathbb{R}^d$, $\mathscr{F}(\cdot)$ be a classifier that predicts a class label $\mathscr{F}(x)$ for an image $x \in \mathbb{R}^d$. The universal adversarial perturbation attack seeks a perturbation vector $v \in \mathbb{R}^d$ under the following constraints [1]:

$$\underset{x \sim \mu_c}{P}\Big(\mathscr{F}(x+v) \neq \mathscr{F}(x)\Big) \geq (1-\delta) \text{ s.t. } \|v\|_p \leq \xi \tag{2.1}$$

where $P(\cdot)$ denotes probability, $\|\cdot\|_p$ is the ℓ_p norm with $p \in [1,\infty)$, $(1-\delta)$ is the target *fooling ratio* with $\delta \in [0,1)$, (i.e., the fraction of samples in μ_c that change labels when perturbed by an adversary) and ξ controls the magnitude of adversarial perturbations. Column 7 in Fig. 2.6 shows visual examples of images affected by universal adversarial perturbations [1].

Chapter 3

DEEPCORRECT: CORRECTING DNNS AGAINST IMAGE DISTORTIONS

In this chapter, *DeepCorrect*, a novel method to make pre-trained DNNs robust to commonly occurring image distortions by identifying and correcting highly susceptible DNN feature activations is discussed in detail, followed by an extensive experimental validation of the proposed approach with different DNN architectures and multiple vision tasks like image classification, scene classification and object detection.

3.1 Introduction

In the proposed method, the effect of image distortions like Gaussian blur and additive noise on the activations of pre-trained convolutional filters is first evaluated. Then, a metric is proposed to identify the most noise susceptible convolutional filters and rank them in order of the highest gain in classification accuracy upon correction. In the proposed approach called *DeepCorrect*, small stacks of convolutional layers with *residual connections* are applied at the output of these ranked filters and trained to correct the worst distortion affected filter activations, whilst leaving the rest of the pre-trained filter outputs in the network unchanged. Applying the proposed *DeepCorrect* models for common vision tasks like image classification [3], object recognition [4] [5] and scene classification [6] significantly improves the robustness of DNNs against distorted images and also outperforms other alternative approaches, while training significantly lesser parameters.

This chapter is organized as follows. A detailed description of the proposed approach is presented in Section 3.2 followed, in Section 3.3, by extensive experimental validation with different DNN architectures and multiple datasets covering image classification, object recognition and scene classification. Concluding remarks are given in Section 3.4.

Table 3.1: Top-1 accuracy of pre-trained networks for distortion affected images as well as undistorted images (original). For Gaussian blur and AWGN, accuracy is reported by averaging over all levels of distortion.

Models	**Original**	**Gaussian blur**	**AWGN**
AlexNet	0.5694	0.2305	0.2375
ResNet18	0.6912	0.3841	0.3255

3.2 DeepCorrect

Although pre-trained networks perform poorly on test images with significantly different image statistics than those used to train these networks (Table 3.1), it is not obvious if only some convolutional filters in a network layer are responsible for most of the observed performance gap or if all convolutional filters in a layer contribute more or less equally to the performance degradation. If only a subset of the filters in a layer are responsible for most of the lost performance, one can focus on restoring only the most severely affected activations and avoid modifying all the remaining filter activations in a DNN.

3.2.1 Ranking Filters Through Correction Priority

One can define the output of a single convolutional filter $\phi_{i,j}$ to the input $\mathbf{x}_i$ by $\phi_{i,j}(\mathbf{x}_i)$, where i and j correspond to layer number and filter number, respectively. If $g_i(\cdot)$ is a transformation that models the distortion acting on filter input $\mathbf{x}_i$, then the output of a convolutional filter $\phi_{i,j}$ to the distortion affected input is given by $\widetilde{\phi_{i,j}}(\mathbf{x}_i) = \phi_{i,j}(g_i(\mathbf{x}_i))$. It should be noted that $\widetilde{\phi_{i,j}}(\mathbf{x}_i)$ represents the filter activations generated by distorted inputs and $\phi_{i,j}(\mathbf{x}_i)$ represents the filter activations for undistorted inputs. Assuming one has access to $\phi_{i,j}(\mathbf{x}_i)$ for a given set of input images, replacing $\widetilde{\phi_{i,j}}(\mathbf{x}_i)$ with $\phi_{i,j}(\mathbf{x}_i)$ in a deep network is akin to perfectly correcting the activations of the convolutional filter $\phi_{i,j}$ against input image distortions. Computing the output predictions by swapping a distortion affected

filter output with its corresponding clean output for each of the ranked filters would improve classification performance. The extent of improvement in performance is indicative of the susceptibility of a particular convolutional filter to input distortion and its contribution to the associated performance degradation.

One can now define the correction priority of a convolutional filter $\phi_{i,j}$ as the improvement in DNN performance on a validation set, generated by replacing $\widetilde{\phi_{i,j}}(\mathbf{x}_i)$ with $\phi_{i,j}(\mathbf{x}_i)$ for a pre-trained network. Let the baseline performance (computed over distorted images) for a network be p_b, which can be obtained by computing the average top-1 accuracy of the network over a set of images or another task-specific performance measure. Let $p_{swp}(i,j)$ denote the new improved performance of the network after swapping $\widetilde{\phi_{i,j}}(\mathbf{x}_i)$ with $\phi_{i,j}(\mathbf{x}_i)$. As our implementation focuses on classification tasks, the average top-1 accuracy over a set of distorted images is used to measure p_b and $p_{swp}(i,j)$. The correction priority for filter $\phi_{i,j}$ is then given by:

$$\tau(i,j) = p_{swp}(i,j) - p_b \tag{3.1}$$

A higher $\tau(i,j)$ indicates higher susceptibility of the convolutional filter $\phi_{i,j}$ to input distortion. Using the proposed ranking measure in Equation (3.1) and 5000 images (i.e., 5 images per class) randomly sampled from the ILSVRC-2012 training set, one can compute correction priorities for every convolutional filter in the network and rank the filters in descending order of correction priority. The detailed overview and pseudo-code for computing correction priorities is summarized in Algorithm 1.

Figure 3.1 evaluates the effect of correcting different percentages, β_i, of the ranked filter activations in the i^{th} DNN layer of AlexNet for distortion affected images. For the AlexNet model, it is possible to recover a significant amount of the lost performance, by correcting only 50% of the filter activations in any one layer, which indicates that a select subset of convolutional filters in each layer are indeed more susceptible to distortions than the rest.

Algorithm 1 Computing Correction Priority

Input: $(\mathbf{x}_{1,i}, g_i(\mathbf{x_{1,i}}), y_1), \ldots, (\mathbf{x}_{M,i}, g_i(\mathbf{x_{M,i}}), y_M)$ are given triplets with $1 \leq i \leq L$, where i represents the layer number, $\mathbf{x_{m,i}}$ is the m^{th} undistorted input for layer i and $g_i(\mathbf{x_{m,i}})$ is the corresponding distorted version, M is the total number of images in the validation set and y_m is the ground-truth label for the m^{th} input image.

Output: Correction priority τ

$p_b := 0$
for $m = 1$ to M **do**
    Predict class label $ypred_m$ for distorted image $g_1(\mathbf{x}_{m,1})$
    Compute $p_b = p_b + \frac{1}{M} h(y_m, ypred_m)$,
where $h(y_m, ypred_m) = 1$, if $y_m = ypred_m$ and 0 otherwise.
end for
for $i = 1$ to L **do**
    $N_i \leftarrow$ number of filters in layer i
    $\phi_{i,j} \leftarrow j^{th}$ convolutional filter in the i^{th} layer
    for $j = 1$ to N_i **do**
        $p_{swp}(j) = 0$
        for $m = 1$ to M **do**
            $\phi_{i,j}(g_i(\mathbf{x}_{m,i})) \leftarrow \phi_{i,j}(\mathbf{x}_{m,i})$
            Predict class label $ypred_m$
            $p_{swp}(j) = p_{swp}(j) + \frac{1}{M} h(y_m, ypred_m)$,
where $h(y_m, ypred_m) = 1$, if $y_m = ypred_m$ and 0 otherwise.
        end for
    end for
    $\tau(i, j) \leftarrow p_{swp}(j) - p_b$
end for
return τ

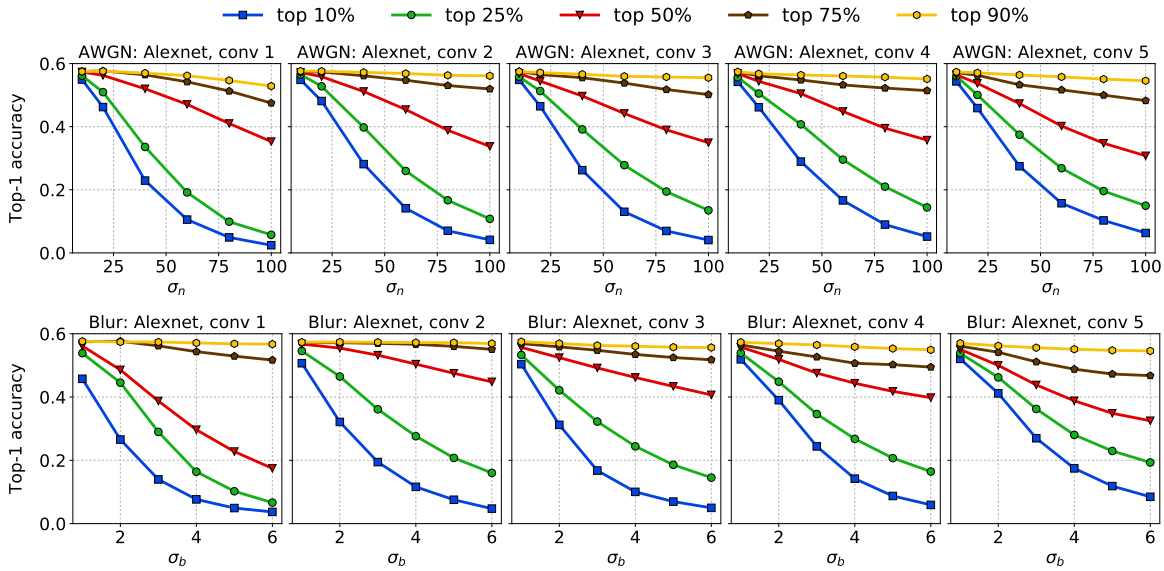
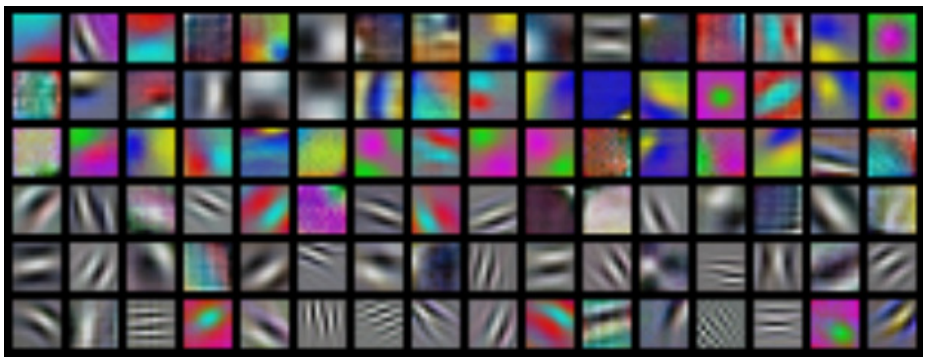


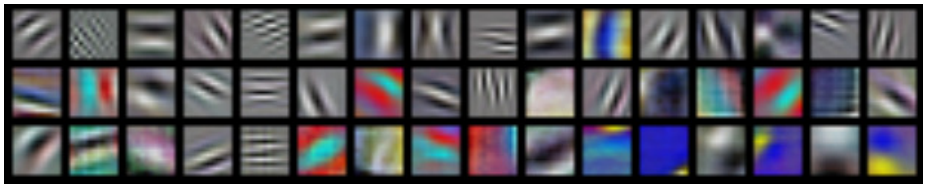
Figure 3.1: Effect of varying the percentage of corrected filter activations $\beta_i \in \{10\%, 25\%, 50\%, 75\%, 90\%\}$, in the i^{th} convolutional layer (conv i) of pre-trained AlexNet, for AWGN and Gaussian blur affected images, respectively.

Although graphs are shown only for the AlexNet model, similar observations can be made for the ResNet18 model as well.

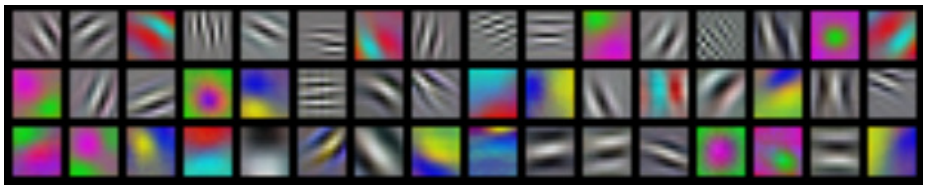
Convolutional filter visualizations from the first layer of the pre-trained AlexNet model (Figure 3.2a) reveal two types of filter kernels: 1) mostly color agnostic, frequency- and orientation-selective filters that capture edges and object contours and 2) color specific blob shaped filters that are sensitive to specific color combinations. Figs. 3.2b and 3.2c visualize the top 50% filters in the first convolutional layer of AlexNet, that are most susceptible to Gaussian blur and AWGN, respectively, as identified by our proposed ranking metric. The identified filters most susceptible to Gaussian blur are mainly frequency- and orientation-selective filters, most of which are color agnostic, while filters most susceptible to AWGN are a mix of both color specific blobs and frequency- and orientation-selective filters. This is in line with our intuitive understanding that Gaussian blur majorly affects edges and object contours and not object color, while AWGN affects color as well as object contours.



(a)



(b)



(c)

Figure 3.2: (a) 96 convolutional filter kernels of size 11x11x3 in the first convolutional layer of pre-trained AlexNet. (b) Convolutional filter kernels most susceptible to Gaussian blur (top 50%), as identified by the proposed ranking metric. (c) Convolutional filter kernels most susceptible to AWGN (top 50%), as identified by the proposed ranking metric. In (b) and (c), the filters are sorted in descending order of susceptibility going row-wise from top left to bottom right.

3.2.2 Correcting Ranked Filter Outputs

Here, a novel approach, referred to as *DeepCorrect* is proposed in this work, where a task-driven corrective transform that acts as a distortion masker for convolutional filters

that are most susceptible to input distortion is learnt, while leaving all the other pre-trained filter outputs in the layer unchanged. Let R_i represent a set consisting of the N_i ranked filter indices in the i^{th} layer of the network, computed using the procedure in Section 3.2.1. Also let R_{i,β_i} represent a subset of R_i consisting of the top $\beta_i N_i$ ranked filter indices in network layer i, where N_i is the total number of convolutional filters in layer i and β_i is the percentage of filters corrected in layer i, as defined in Section 3.2.1. If Φ_i represents the set of convolutional filters in the i^{th} layer, the objective is to learn a transform $F_{corr_i}(:)$ such that:

$$F_{corr_i}(\Phi_{R_{i,\beta_i}}(g_i(\mathbf{x}_i))) \approx \Phi_{R_{i,\beta_i}}(\mathbf{x}_i) \tag{3.2}$$

where $\mathbf{x}_i$ is the undistorted input to the i^{th} layer of convolutional filters and $g_i(\cdot)$ is a transformation that models the distortion acting on $\mathbf{x}_i$. Since no specific form is assumed for the image distortion process, the corrective transform $F_{corr_i}(:)$ is set to take the form of a shallow *residual block*, which is a small stack of convolutional layers (4 layers) with a single skip connection [7], such as the one shown in Figure 3.3. Such a *residual block* is referred to as a *correction unit* in the remainder of this chapter. $F_{corr_i}(:)$ can now be estimated using a target-oriented loss such as the one used to train the original network, through backpropogation [89], but with much less number of parameters.

Consider an L layered DNN Φ that has been pre-trained for an image classification task using clean images. Φ can be interpreted as a function that maps network input $\mathbf{x}$ to an output vector $\Phi(\mathbf{x}) \in \mathbb{R}^d$, such that:

$$\Phi = \Phi_L \circ \Phi_{L-1} \circ \ldots \Phi_2 \circ \Phi_1 \tag{3.3}$$

where Φ_i is the mapping function (set of convolutional filters) representing the i^{th} DNN layer and d is the dimensionality of the network output.

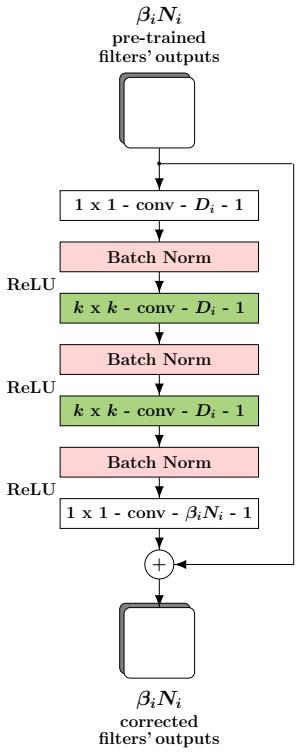


Figure 3.3: *Correction unit* based on a residual function [7], acting on the outputs of $\beta_i N_i$ $(0 < \beta_i < 1)$ filters out of N_i total filters in the i^{th} convolutional layer of a pre-trained DNN. All convolutional layers in the *residual block*, except the first and last layer, are parameterized by *k*x*k*-conv-D_i-*s*, where *k*x*k* is spatial extent of the filter, D_i (correction unit kernel depth) is the number of output filters in a layer, *s* represents the filter stride and *i* represents the layer number of the convolutional layer being corrected in the pre-trained DNN.

Without loss of generality, if one adds a *correction unit* that acts on the top $\beta_1 N_1$ ranked filters in the first network layer, then the resultant network Φ_{corr} is given by:

$$\Phi_{corr} = \Phi_L \circ \Phi_{L-1} \circ \ldots \Phi_2 \circ \Phi_{1_{corr}} \tag{3.4}$$

where $\Phi_{1_{corr}}$ represents the new mapping function for the first layer, in which the corrective transform $F_{corr_1}(:)$ acts on the activations of the filter subset $\Phi_{R_{1,\beta_1}}$ and all the remaining

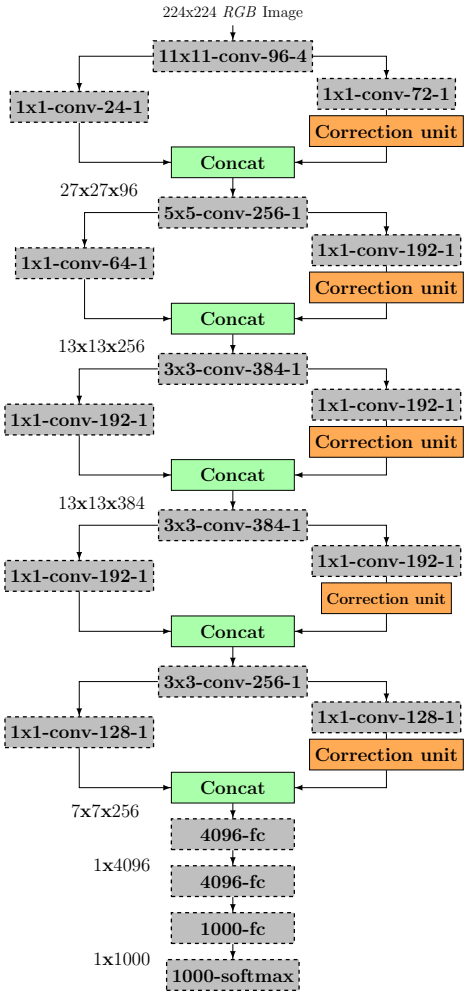


Figure 3.4: *DeepCorrect* model for AlexNet, with 75% filter outputs corrected in the first two layers and 50% filter outputs corrected in the next three layers. Convolution layers from the original architecture in Figure 2.1, shown in gray with dashed outlines, are non-trainable layers and their weights are kept the same as those of the pre-trained model. The 1×1 convolution layer preceding each *correction unit* implements a one-hot encoding kernel, where the number of output activation maps represent the number of filter activation maps being corrected ($\beta_i N_i$) and a kernel weight of 1 represents the index of the input filter activation map being selected for correction. A similar 1×1 convolution layer is used for selecting activation maps that are left unchanged from the original DNN.

filter activations are left unchanged. If $\mathbf{W}_1$ represents the trainable parameters in F_{corr_1}, then F_{corr_1} can be estimated by minimizing :

$$E(\mathbf{W}_1) = \lambda \mathscr{R}(\mathbf{W_1}) + \frac{1}{M}\sum_{m=1}^{M} \mathscr{L}(y_m, \Phi_{corr}(\mathbf{x}_m)) \tag{3.5}$$

where λ is a constant, $\mathscr{R}$ is a regularizer such as l_1 norm or l_2 norm, $\mathscr{L}$ is a standard cross-entropy classification loss, y_m is the target output label for the m^{th} input image $\mathbf{x}_m$, M represents the total number of images in the training set and, since a collection of both distorted and clean images are used during training, $\mathbf{x}_m$ represents a clean or a distorted image. The trainable parameters in Equation (3.5) are $\mathbf{W}_1$, while all other network parameters are fixed and kept the same as those in the pre-trained models. Although Equation (3.5) shows *correction unit* estimation for only the first layer, it is possible to add such units at the output of distortion susceptible filters in any layer and in one or more layers. Figure 3.4 shows an example of a *DeepCorrect* model for the pre-trained AlexNet model in Figure 2.1a.The *correction unit* is applied before the ReLU non-linearity acting upon the distortion-susceptible convolutional filter outputs. Max pooling layers following convolutional layers 1, 2 and 5 in the pre-trained AlexNet have not been shown in the ImageNet *DeepCorrect* model for uniformity.

3.2.3 Rank-constrained DeepCorrect Models

The inference time of the proposed *DeepCorrect* models (Section 3.2.2) can be slower than the respective baseline models due to the additional computational cost introduced by the *correction units*. To mitigate the impact of these additional computations, a rank-constrained approximation is used in place of the full-rank *DeepCorrect* model, which not only has the same computational cost as the corresponding baseline DNN but also retains almost 99% of the performance of our full-rank *DeepCorrect* models.

Consider the n^{th} full-rank 3-D convolutional filter with weights $\mathbf{W}_n \in \mathbb{R}^{k\times k\times C}$ in a DNN convolutional layer with N filters, where $k \times k$ represents the filter's spatial extent and C is the number of input channels for the filter; then a rank constrained convolution is implemented by factorizing the convolution of $\mathbf{W}_n$ with input z into a sequence of separable convolutions (i.e., horizontal and vertical filters) as in [90]:

$$\mathbf{W}_n * z \approx \sum_{p=1}^{P} h_n^p * (v_p * z) = \sum_{p=1}^{P} h_n^p * \sum_{c=1}^{C} v_p^c * z^c \tag{3.6}$$

where the first convolutional filter bank consists of P vertical filters $\{v_p \in \mathbb{R}^{k\times 1\times C} : p \in [1...P]\}$ and the second convolution consists of a horizontal filter that operates on P input feature maps $\{h_n \in \mathbb{R}^{1\times k\times P}\}$. The number of intermediate filter maps, P, controls the rank of the low-rank approximation. The computational cost of the original full-rank convolutional layer for N output feature maps with width W' and height H' is $O(Nk^2CH'W')$, whereas the rank-constrained approximation has a computational cost of $O((N+C)kPH'W')$ and a speedup can be achieved when $NkC > (N+C)P$. For the special case of convolutional layers in our *correction units*, where $N = C$ (Figure 3.3), if $P = N/2$, the computational cost of a convolutional layer can be reduced k times (typically, $k = 3$).

The rank-constrained *DeepCorrect* model is thus generated by replacing each full-rank convolutional layer (except 1×1 layers) with its respective rank-constrained approximation with P for each approximation chosen such that the total computational cost of the proposed model is the same as the baseline DNN. Instead of using iterative methods or training the separable filters from random weights, the simple yet fast, matrix decomposition approach of Tai *et. al.* [91] is used to get the exact global optimizer of the rank-constrained approximation from its respective trained full-rank *DeepCorrect* model.

3.3 Experimental Results

The proposed *DeepCorrect* models are evaluated against the alternative approaches of network fine-tuning and *stability training* [73], for the DNN architectures mentioned in Section 2.3. The DNN models are trained and tested using a single Nvidia Titan-X GPU. Unlike common image denoising and deblurring methods like BM3D [92] and NCSR [93] which expect the distortion level to be known during both train and test phases or learning-based methods that train separate models for each distortion level, *DeepCorrect* trains a single model for all distortion levels at once and, consequently, there is no need to know the distortion level at test time.

3.3.1 AlexNet Analysis

Finetune Model

The AlexNet model in Figure 2.1a is fine-tuned on a mix of distortion affected images and clean images to generate a single fine-tuned model and refer to this model as Finetune in the presented results. Starting with an initial learning rate (= 0.001) that is 10 times lower than that used to generate the pre-trained model in Figure 2.1a, a fixed number of iterations (62500 iterations $\approx$ 10 epochs) is adopted, with the learning rate reduced by a factor of 10 after every 18750 iterations (roughly 3 epochs). Additionally, a data augmentation method as proposed by [7] is also used.

Stability Trained Model

Following the *stability training* method outlined in [73] that considers that unseen distortions can be modelled by adding AWGN to the input image, all the fully connected layers of the pre-trained AlexNet model are fine-tuned by minimizing the KL-divergence between the classification scores for a pair of images (I, I'), where $I' = I + \eta$ and $\eta \sim \mathcal{N}(0, \sigma^2)$.

Table 3.2: Top-1 accuracy of AlexNet-based DNN models for distortion affected images of the ImageNet validation set (ILSVRC-2012), averaged over all levels of distortion and clean images. Bold numbers show best accuracy and underlined numbers show next best accuracy.

Method	**Gaussian blur**	**AWGN**
Baseline	0.2305	0.2375
Finetune	0.4596	0.4894
Finetune-rc	0.4549	0.4821
Deepcorr	**0.5071**	**0.5092**
Deepcorr-b	<u>0.5022</u>	<u>0.5063</u>
Deepcorr-rc	0.4992	0.5052
Stability[73]	0.2163	0.2305
NCSR[93]+AlexNet[8]	0.2193	-
BM3D[92]+AlexNet[8]	-	0.5032

The same hyper-parameters used for the classification task in [73] are adopted here: $\sigma^2 = 0.04$ and regularization coefficient $\alpha = 0.01$.

DeepCorrect **Models**

The main *DeepCorrect* model for AlexNet shown in Figure 3.4 and referred to as *Deepcorr* in Table 3.2 is generated by correcting 75% ranked filter outputs in the first two layers ($\beta_1, \beta_2 = 0.75$) and 50% ranked filter outputs in the next three layers ($\beta_3, \beta_4, \beta_5 = 0.5$) of the pre-trained AlexNet shown in Figure 2.1a. The *correction units* (Figure 3.3) in each convolutional layer are trained using an initial learning rate of 0.1 and the same learning rate schedule, data augmentation and total iterations used for generating the Finetune model in Section 3.3.1. Two additional variants are also generated based on the proposed *Deepcorr* model, *Deepcorr*-b, a computationally lighter model than *Deepcorr* based on a bottleneck

Table 3.3: Computational performance of AlexNet-based DNN models.

Metric	**Baseline/ Finetune**	***Deepcorr***	***Deepcorr*-b**	***Deepcorr*-rc**
FLOPs	7.4×10^8	23.9×10^8	11.8×10^8	7.8×10^8
Trainable params	60.96M	2.81M	1.03M	1.03M [1]

architecture for its *correction units* as described later in this section, and *Deepcorr*-rc, which is a rank-constrained model (Section 3.2.3) derived from the full-rank *Deepcorr*-b model such that its test-time computational cost is almost the same as the Finetune model. For comparison, a rank-constrained model is also derived for the Finetune model using similar decomposition parameters as *Deepcorr*-rc and the resulting model is denoted by Finetune-rc.

Table 3.2 shows the superior performance of the proposed method as compared to the alternative approaches and Table 3.3 summarizes the computational performance in terms of trainable parameters and floating point operations (FLOPs) of these DNN models during training and testing, respectively. The training computational cost is evaluated in terms of the number of trainable parameters that are updated during training. The test-time computational cost is evaluated in terms of the total FLOPs, i.e., total multiply-add operations needed for a single image inference, as outlined by He *et al.* in [7]. In particular, a $k \times k$ convolutional layer operating on C input maps and producing N output feature maps of width W' and height H' requires $Nk^2CH'W'$ FLOPs [7]. The detailed architecture and corresponding FLOPs for each *correction unit* in the different proposed *DeepCorrect* models for AlexNet are summarized in Table 3.4. Design choices for various *correction unit* architectures and their impact on inference-time computational cost are also discussed later in this section.

[1] Since *Deepcorr*-rc is derived from the *Deepcorr*-b model through a low-rank approximation, the number of trainable parameters and subsequently its effect on model convergence is the same as *Deepcorr*-b.

Table 3.4: *Correction unit* architectures for AlexNet-based *DeepCorrect* models. The number following Corr-unit specifies the layer at which *correction units* are applied. *Correction unit* convolutional layers are represented as *k*x*k*, *d*, where *k*x*k* is the spatial extent of a filter and *d* is the number of filters in a layer. Stacked convolutional layers are enclosed in brackets, followed by the number of layers stacked.

Model	**Corr-unit 1** $\beta_1 N_1 = 72$ **Output size** 55×55	**Corr-unit 2** $\beta_2 N_2 = 192$ **Output size** 27×27	**Corr-unit 3** $\beta_3 N_3 = 192$ **Output size** 13×13	**Corr-unit 4** $\beta_4 N_4 = 192$ **Output size** 13×13	**Corr-unit 5** $\beta_5 N_5 = 128$ **Output size** 13×13
Deepcorr	$1\times1, 72$ $\begin{bmatrix} 5\times5, 72 \end{bmatrix}\times2$ $1\times1, 72$ FLOPs: 8.1×10^8	$1\times1, 192$ $\begin{bmatrix} 3\times3, 192 \end{bmatrix}\times2$ $1\times1, 192$ FLOPs: 5.3×10^8	$1\times1, 192$ $\begin{bmatrix} 3\times3, 192 \end{bmatrix}\times2$ $1\times1, 192$ FLOPs: 1.2×10^8	$1\times1, 192$ $\begin{bmatrix} 3\times3, 192 \end{bmatrix}\times2$ $1\times1, 192$ FLOPs: 1.2×10^8	$1\times1, 128$ $\begin{bmatrix} 3\times3, 128 \end{bmatrix}\times2$ $1\times1, 128$ FLOPs: 5.5×10^7
Total FLOPs: 16.5×10^8					
Deepcorr-b	$1\times1, 36$ $\begin{bmatrix} 3\times3, 36 \end{bmatrix}\times3$ $1\times1, 72$ FLOPs: 1.2×10^8	$1\times1, 96$ $\begin{bmatrix} 3\times3, 96 \end{bmatrix}\times3$ $1\times1, 192$ FLOPs: 2.0×10^8	$1\times1, 96$ $\begin{bmatrix} 3\times3, 96 \end{bmatrix}\times3$ $1\times1, 192$ FLOPs: 4.8×10^7	$1\times1, 96$ $\begin{bmatrix} 3\times3, 96 \end{bmatrix}\times3$ $1\times1, 192$ FLOPs: 4.8×10^7	$1\times1, 64$ $\begin{bmatrix} 3\times3, 64 \end{bmatrix}\times3$ $1\times1, 128$ FLOPs: 2.1×10^7
Total FLOPs: 4.4×10^8					
Deepcorr-rc	$1\times1, 36$ $\begin{bmatrix} 3\times1, 27 \\ 1\times3, 36 \end{bmatrix}\times3$ $1\times1, 72$ FLOPs: 6.8×10^7	$1\times1, 96$ $\begin{bmatrix} 3\times1, 72 \\ 1\times3, 96 \end{bmatrix}\times3$ $1\times1, 192$ FLOPs: 1.1×10^8	$1\times1, 96$ $\begin{bmatrix} 3\times1, 72 \\ 1\times3, 96 \end{bmatrix}\times3$ $1\times1, 192$ FLOPs: 2.7×10^7	$1\times1, 96$ $\begin{bmatrix} 3\times1, 72 \\ 1\times3, 96 \end{bmatrix}\times3$ $1\times1, 192$ FLOPs: 2.7×10^7	$1\times1, 64$ $\begin{bmatrix} 3\times1, 48 \\ 1\times3, 64 \end{bmatrix}\times3$ $1\times1, 128$ FLOPs: 1.2×10^7
Total FLOPs: 2.5×10^8					

As can be seen from Table 3.2, the *Deepcorr* model, which is the best performing model in terms of top-1 classification accuracy, outperforms the Finetune model with $\approx 10\%$ and $\approx 4\%$ relative average improvement for Gaussian blur and AWGN, respectively, by just training $\approx$ 2.81M parameters (Table 3.3) as compared to 61M parameters for the Finetune model (i.e., 95.4% lesser parameters). *Deepcorr* significantly improves the robustness of a pre-trained DNN achieving an average top-1 accuracy of 0.5071 and 0.5092 for Gaussian blur and AWGN affected images, respectively, as compared to the corresponding top-1 accuracy of 0.2305 and 0.2375 for the pre-trained AlexNet DNN.

All the proposed *DeepCorrect* model variants consistently outperform the Finetune and Stability models for both distortion types (Table 3.2). One can also observe that fine-tuning DNN models on distortion specific data significantly outperforms DNN models trained through distortion agnostic *stability training*. For completeness, the classification performance is compared with AlexNet when combined with a commonly used non-blind image denoising method (BM3D) proposed by [92] and a deblurring method (NCSR) proposed by [93], where BM3D and NCSR are applied prior to the baseline AlexNet, for AWGN and Gaussian blur, respectively. Table 3.2 shows top-1 accuracy for these two methods with the *Deepcorr* model outperforming each for AWGN and Gaussian blur, respectively.

Correction Unit Architectures

Increasing the correction unit kernel depth (D_i in Figure 3.3) makes the proposed *correction unit* fatter, whereas decreasing D_i makes the *correction unit* thinner. A natural choice for D_i would be to make it equal to the number of distortion susceptible filters that need correction ($\beta_i N_i$), in a DNN layer i; this is also the default parameter setting used in the *correction units* for *Deepcorr*. Since D_i is always equal to the number of distortion susceptible filters that need correction (i.e., $D_i = \beta_i N_i$), the number of trainable parameters in the *correction units* of *Deepcorr* scale linearly with the number of corrected filters ($\beta_i N_i$) in each convolutional layer, even though *Deepcorr* trains significantly lesser parameters than Finetune and still achieves a better classification accuracy (Tables 3.2 and 3.3).

As shown in Table 3.3, during the testing phase, the *correction units* in *Deepcorr* add almost 2 times more FLOPs relative to the baseline DNN, for evaluating a single image. As discussed in more detail later in this section, one way to limit the number of trainable parameters and FLOPs is to explore a bottleneck architecture for our *correction units*, where the convolutional kernel depth D_i (Figure 3.3) is set to 50% of the distortion susceptible filters that need correction in a DNN layer i (i.e., $D_i = \frac{\beta_i N_i}{2}$) as compared to D_i being set

to the number of filters to correct (i.e., $D_i = \beta_i N_i$) in *Deepcorr*. Replacing each *correction unit* in *Deepcorr* with such a bottleneck *correction unit*, which is referred to as *Deepcorr*-b, results in a *DeepCorrect* model that has significantly less trainable parameters and FLOPs than the original *Deepcorr* model (Table 3.4). Compared to the *correction units* in *Deepcorr*, bottleneck *correction units* provide a 60% reduction in FLOPs on average, with a 85% reduction in FLOPs for Corr-unit 1 as shown in Table 3.4. From Tables 3.2 and 3.3, it can be seen that *Deepcorr*-b achieves $\approx$ 99% of the average accuracy achieved by *Deepcorr*, with a 63% reduction in trainable parameters and a 73% reduction in FLOPs, relative to *Deepcorr*. As shown in Table 3.3, the *Deepcorr*-b model still requires 58% more FLOPs relative to the baseline DNN, for evaluating a single image at test time. A further reduction in the computational cost at test-time can be achieved by deriving a rank-constrained *DeepCorrect* model (*Deepcorr*-rc) from the *Deepcorr*-b model using the approach outlined in Section 3.2.3. By replacing each full rank convolutional layer in a *Deepcorr*-b *correction unit* with a pair of separable convolutions (Section 3.2.3), one can reduce the FLOPs for each *correction unit* by an additional 46% as shown in Table 3.4. Replacing the pre-trained convolutional layers of the baseline AlexNet (which are left unchanged in the *DeepCorrect* models and shown in gray in Figure 3.4) by their equivalent low-rank approximations provides an additional 29% reduction in FLOPs, relative to the baseline AlexNet model, such that the resultant *Deepcorr*-rc model now has almost the same FLOPs as Finetune (Table 3.3) and still retains almost 99% of the accuracy achieved by *Deepcorr*.

Effect of Ranking on *Correction Unit* Performance

If the superior performance of the *DeepCorrect* model is only due to the additional network parameters provided by the *correction unit*, then re-training the correction unit on the least susceptible $\beta_i N_i$ filter outputs in a DNN layer should also achieve the same performance as that achieved by applying a *correction unit* on the $\beta_i N_i$ most susceptible filter outputs

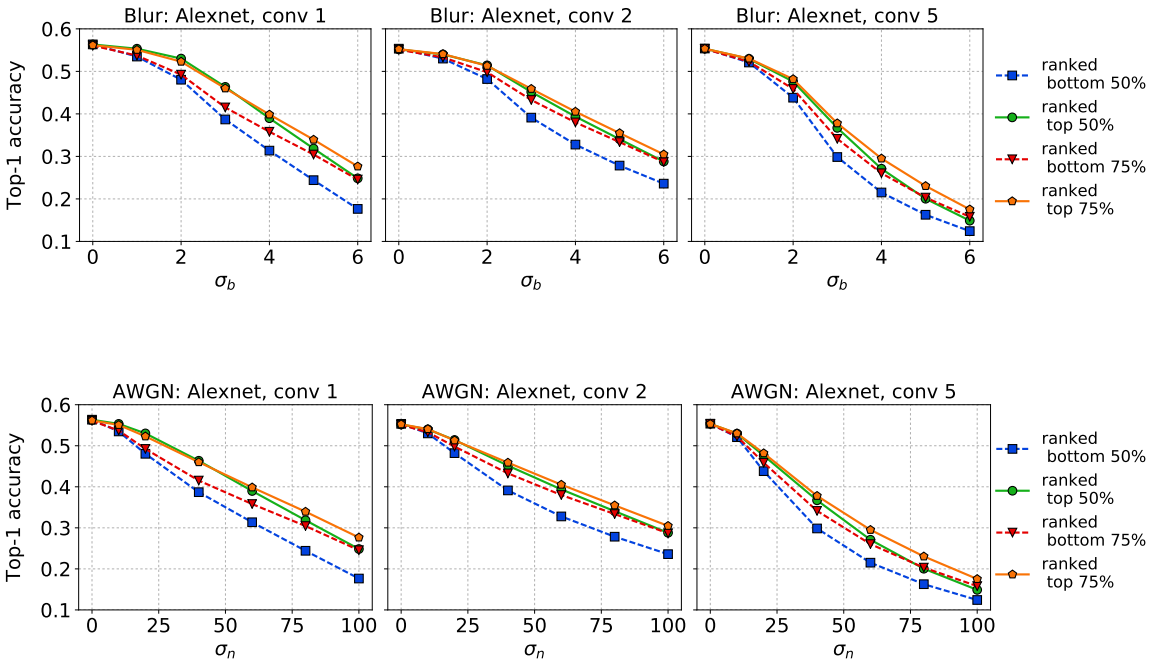


Figure 3.5: Effect of *DeepCorrect* ranking metric on *correction unit* performance when integrated with AlexNet[8]. Dashed lines represent *correction units* trained on the least susceptible filters in a DNN layer and solid lines represent *correction units* trained on the most susceptible filters, as identified by our ranking metric (Section 3.2.1).

identified by the proposed ranking metric (Section 3.2.1). β_i and N_i, as defined in Section 3.2.1, represent the percentage of filters corrected in the i^{th} layer and the total number of filters in the i^{th} layer, respectively. To this end, the performance of training *correction units* is evaluated on: 1) $\beta_i N_i$ filters most susceptible to distortion, and 2) $\beta_i N_i$ filters least susceptible to distortion, as identified by the proposed ranking metric. For this analysis, $\beta_i \in \{50\%, 75\%\}$.

As shown in Figure 3.5, *correction units* trained on the $\beta_i N_i$ most distortion susceptible filters (solid lines) outperform those trained on the least susceptible filters (dashed lines) of conv 1, conv 2 and conv 5 layers of the AlexNet model, for Gaussian blur and AWGN,

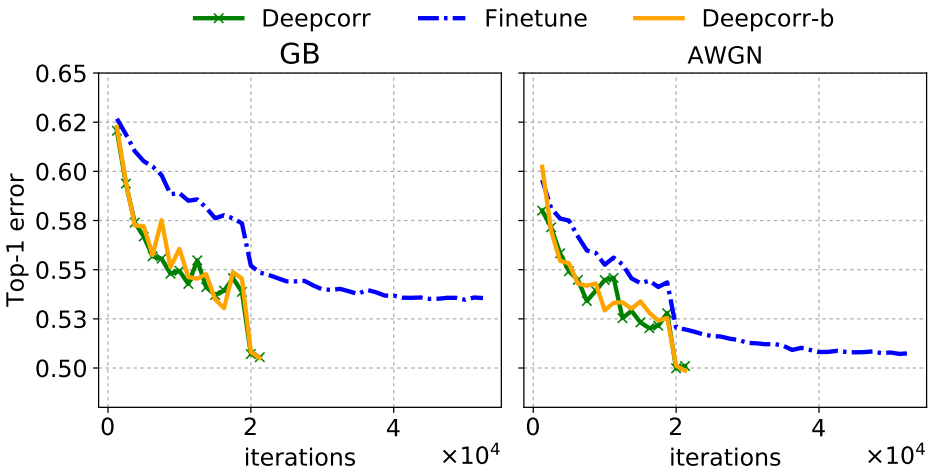


Figure 3.6: Top-1 error on the ILSVRC-2012 validation set, for *DeepCorrect* model variants and fine-tuning.

respectively. Although a similar trend is observed for the conv 3 and conv 4 layers, results are plotted for only conv 1, conv 2 and conv 5 as these show the largest difference in performance due to ranking distortion susceptible filters. Correcting the top 75% distortion susceptible filters (solid orange), as identified by the proposed ranking measure, achieves the best performance in all layers. Similarly, for all layers and distortions, *correction units* trained on the top 50% susceptible filters (solid green) not only significantly outperform *correction units* trained on the 50% least susceptible filters (dashed blue) but also outperform *correction units* trained on the 75% least susceptible filters (dashed red), where 25% filters are shared among both approaches.

Accelerating Training

The evolution of the validation set error over training iterations for *Deepcorr*, *Deepcorr*-b as well as the Finetune model is analyzed in this section. During this evaluation, training for the *Deepcorr* models is stopped when their respective validation error is lesser than or equal to the minimum validation error achieved by the Finetune model. For any particular value

of validation error achieved by the Finetune model in Figure 3.6, both the *Deepcorr* model variants are able to achieve the same validation error in much lesser number of training iterations. Thus, one can conclude that just learning corrective transforms for activations of a subset of convolutional filters in a layer accelerates training through reduced number of training epochs needed for convergence.

Generalization of *DeepCorrect* Features to Other Datasets

To analyze the ability of distortion invariant features learnt for image classification to generalize to related tasks like object recognition and scene recognition, the *Deepcorr* and Finetune models trained on the ImageNet dataset (Section 3.3.1) are evaluated as discriminative deep feature extractors on the Caltech-101 [4], Caltech-256 [5] and SUN-397 [6] datasets. Unlike object recognition datasets like Caltech-101 and Caltech-256, which bear some similarity to an image classification/object recognition dataset like ImageNet, a scene recognition dataset like SUN-397 bears no similarity to the ImageNet dataset and is expected to be challenging for features extracted using models learnt on ImageNet [94]. Following the experimental procedure proposed by [94], the output of the first (Caltech-101 and Caltech-256) or second (SUN-397) fully-connected layer in these models is used as a deep feature extractor for images affected by distortion.

Since the above deep feature models have not been trained on any one of these datasets (Caltech-101, Caltech-256 and SUN-397), for each dataset, linear SVMs are trained on top of the deep features, which are extracted from a random set of training data, and evaluate the performance in terms of mean accuracy per category averaged over 5 data splits, following the training procedure adopted by [94]. The training data for each split consists of 25 training images and 5 validation images per class, sampled randomly from the considered dataset and all remaining images are used for testing. A baseline accuracy for undistorted images is first established by training linear SVMs on features extracted only from undis-

Table 3.5: Mean accuracy per category of pre-trained AlexNet deep feature extractor for clean images.

Caltech-101	Caltech-256	SUN-397
0.8500	0.6200	0.3100

Table 3.6: Mean accuracy per category for Gaussian blur affected images, averaged over all distortion levels. Bold numbers show best accuracy.

Dataset	Baseline	Finetune	*Deepcorr*
Caltech-101	0.4980	0.7710	**0.8371**
Caltech-256	0.2971	0.5167	**0.5883**
SUN-397	0.1393	0.2369	**0.3049**

Table 3.7: Mean accuracy per category for AWGN affected images, averaged over all distortion levels. Bold numbers show best accuracy.

Dataset	Baseline	Finetune	*Deepcorr*
Caltech-101	0.3423	0.7705	**0.8034**
Caltech-256	0.1756	0.4995	**0.5482**
SUN-397	0.0859	0.1617	**0.2936**

torted images using the AlexNet DNN shown in Figure 2.1a, and results are reported in Table 3.5.

Similar to the evaluation in Section 3.3.1, Gaussian blur and AWGN are now independently added to train and test images using the same distortion levels as reported in Section 2.5 and performance averaged over all distortion levels is reported in Tables3.6-3.7 for deep features extracted using baseline AlexNet, Finetune and *Deepcorr* models trained on ImageNet. For each of the three models, a single set of linear SVMs is trained for images affected by different levels of distortion and also clean images. Both Gaussian blur

Table 3.8: Top-1 accuracy of ResNet18-based DNN models for distortion affected images of the ImageNet validation set (ILSVRC-2012), averaged over all levels of distortion and clean images. Bold numbers show best accuracy and underlined numbers show next best accuracy.

Method	**G.Blur**	**AWGN**	**M.Blur**	**D.Blur**	**Cam.Blur**
Baseline	0.3841	0.3255	0.4436	0.3582	0.4749
Finetune	0.5617	0.5970	0.6197	0.5615	0.6041
Finetune-rc	0.5548	0.5898	0.6113	0.5537	0.5973
Deepcorr	0.5808	<u>0.6058</u>	**0.6498**	**0.6005**	<u>0.6346</u>
Deepcorr-b	**0.5839**	**0.6087**	<u>0.6474</u>	<u>0.5831</u>	**0.6365**
Deepcorr-rc	<u>0.5821</u>	0.6033	0.6411	0.5785	0.6276
Stability[73]	0.3412	0.3454	0.4265	0.3182	0.4720

and AWGN significantly affect the accuracy of the baseline feature extractor for all three datasets, with a 41% and 60% drop in respective accuracies for Caltech-101, a 52% and 71% drop in respective accuracies for Caltech-256, and a 55% and 72% drop in respective mean accuracy for SUN-397, relative to the benchmark performance for clean images. For Caltech-101, the *Deepcorr* feature extractor outperforms the Finetune feature extractor with a 8.5% and 4.2% relative improvement in mean accuracy for Gaussian blur and AWGN affected images, respectively. For Caltech-256, the *Deepcorr* feature extractor outperforms the Finetune feature extractor with a 13.8% and 9.7% relative improvement for Gaussian blur and AWGN, respectively. Similarly, features extracted using the *Deepcorr* model significantly outperform those extracted using the Finetune model for SUN-397, with a 28.7% and 81.5% relative improvement in mean accuracy for Gaussian blur and AWGN, respectively. The large performance gap between *Deepcorr* and Finetune feature extractors highlights the generic nature of distortion invariant features learnt by our *DeepCorrect* models.

Table 3.9: Computational performance of ResNet18-based DNN models.

Metric	**Baseline/ Finetune**	***Deepcorr***	***Deepcorr*-b**	***Deepcorr*-rc**
FLOPs	1.8×10^9	3.5×10^9	2.9×10^9	1.8×10^9
Trainable params	11.7M	8.41M	5.5M	5.5M

3.3.2 ResNet18 Analysis

Similar to the AlexNet analysis in Section 3.3.1, for ResNet18, the performance of our proposed *DeepCorrect* models is evaluated against DNN models trained through fine-tuning and *stability training*.

Using the training procedures outlined in Sections 3.3.1 and 3.3.1, fine-tuned and *stability trained* models are generated from the baseline ResNet18 DNN by training all layers (11.7M parameters) of the DNN on a mix of distorted and clean images. Similarly, using the training procedure of Section 3.3.1, the competing *DeepCorrect* model is generated by training *correction units* that are appended at the output of the most susceptible filters in the odd-numbered convolutional layers (1 to 17) of the baseline ResNet18 model right after each skip connection merge (Figure 2.1c). Similar to the AlexNet analysis presented earlier (Section 3.3.1), three *DeepCorrect models* (i.e., *Deepcorr*, *Deepcorr*-b and *Deepcorr*-rc) are generated. For comparison, a rank-constrained model is derived for the Finetune model using similar decomposition parameters as *Deepcorr*-rc and the resulting model is denoted by Finetune-rc. Table 3.8 summarizes the accuracy of ResNet18-based DNN models against various distortions, while Table 3.9 shows the computational performance of the same DNN models, measured in terms of FLOPs and trainable parameters. Similar to the AlexNet analysis, the detailed architecture and corresponding FLOPs for each *correction unit* in our different *DeepCorrect* models for ResNet18 are shown in Table 3.10.

Table 3.10: *Correction unit* architectures for ResNet18-based *DeepCorrect* models. The number following Corr-unit specifies the layer at which *correction units* are applied. *Correction unit* convolutional layers are represented as *k*x*k*, *d*, where *k*x*k* is the spatial extent of a filter and *d* is the number of filters in a layer. Stacked convolutional layers are enclosed in brackets, followed by the number of layers stacked.

Model	**Corr-unit 1** $\beta_1 N_1 = 48$ **Output size** 112×112	**Corr-unit 3, 5** $\beta_2 N_2 = 48$ **Output size** 56×56	**Corr-unit 7, 9** $\beta_3 N_3 = 96$ **Output size** 28×28	**Corr-unit 11, 13** $\beta_4 N_4 = 192$ **Output size** 14×14	**Corr-unit 15, 17** $\beta_5 N_5 = 384$ **Output size** 7×7
Deepcorr	1×1, 48 $\begin{bmatrix} 3\times3, 48 \end{bmatrix}\times2$ 1×1, 48 Total FLOPs: 1.7×10^9	1×1, 48 $\begin{bmatrix} 3\times3, 48 \end{bmatrix}\times2$ 1×1, 48 FLOPs: 5.7×10^8	1×1, 96 $\begin{bmatrix} 3\times3, 96 \end{bmatrix}\times2$ 1×1, 96 FLOPs: 1.4×10^8	1×1, 192 $\begin{bmatrix} 3\times3, 192 \end{bmatrix}\times2$ 1×1, 192 FLOPs: 1.4×10^8	1×1, 384 $\begin{bmatrix} 3\times3, 384 \end{bmatrix}\times2$ 1×1, 384 FLOPs: 1.4×10^8
Deepcorr-b	1×1, 31 $\begin{bmatrix} 3\times3, 31 \end{bmatrix}\times3$ 1×1, 48 Total FLOPs: 1.1×10^9	1×1, 31 $\begin{bmatrix} 3\times3, 31 \end{bmatrix}\times3$ 1×1, 48 FLOPs: 9.2×10^7	1×1, 62 $\begin{bmatrix} 3\times3, 62 \end{bmatrix}\times3$ 1×1, 96 FLOPs: 9.2×10^7	1×1, 124 $\begin{bmatrix} 3\times3, 124 \end{bmatrix}\times3$ 1×1, 192 FLOPs: 9.2×10^7	1×1, 248 $\begin{bmatrix} 3\times3, 248 \end{bmatrix}\times3$ 1×1, 384 FLOPs: 9.2×10^7
Deepcorr-rc	1×1, 31 $\begin{bmatrix} 3\times1, 24 \\ 1\times3, 31 \end{bmatrix}\times3$ 1×1, 48 Total FLOPs: 0.6×10^9	1×1, 31 $\begin{bmatrix} 3\times1, 24 \\ 1\times3, 31 \end{bmatrix}\times3$ 1×1, 48 FLOPs: 5.1×10^7	1×1, 62 $\begin{bmatrix} 3\times1, 48 \\ 1\times3, 62 \end{bmatrix}\times3$ 1×1, 96 FLOPs: 5.1×10^7	1×1, 124 $\begin{bmatrix} 3\times1, 96 \\ 1\times3, 124 \end{bmatrix}\times3$ 1×1, 192 FLOPs: 5.1×10^7	1×1, 248 $\begin{bmatrix} 3\times1, 192 \\ 1\times3, 248 \end{bmatrix}\times3$ 1×1, 384 FLOPs: 5.1×10^7

From Table 3.8, one can observe that the *DeepCorrect* models not only significantly improve the robustness of the baseline DNN model to input distortions but also outperform the alternative approaches of model fine-tuning and *stability training* in terms of classification accuracy. *Deepcorr*, which trains almost 28% lesser parameters (Table 3.9) than Finetune, outperforms Finetune by 3.2% for Gaussian blur, 1.47% for AWGN, 4.86% for motion blur, 6.95% for defocus blur and 5% for camera shake blur. *Deepcorr*-b, which trains 50% lesser parameters (Table 3.9) than Finetune, outperforms Finetune by 3.95% for Gaussian blur, 1.9% for AWGN, 4.46% for motion blur, 3.84% for defocus blur and 5.36%

for camera shake blur. Similarly the rank constrained model (*Deepcorr*-rc), which trains 50% parameters less than Finetune (Table 3.9), outperforms Finetune by 3.63% for Gaussian blur, 1% for AWGN, 3.45% for motion blur, 3% for defocus blur and 3.89% for camera shake blur. As shown in Tables 3.9 and 3.10, during the testing phase, *Deepcorr*-b requires 60% more FLOPs relative to the baseline DNN, for evaluating a single image. On the other hand, for *Deepcorr*-rc, just using rank-constrained *correction units* results in a 50% reduction in FLOPs as compared to *Deepcorr*-b (Table 3.10), with a 30% additional reduction in FLOPs relative to baseline ResNet18 achieved by replacing the pre-trained convolutional layers of baseline ResNet18 with their low-rank approximations. From Tables 3.8 and 3.9, it can be seen that *Deepcorr*-rc requires almost the same number of FLOPs as Finetune but achieves a superior performance than the Finetune and Stability models without sacrificing inference speed.

3.4 Conclusion

Deep networks trained on pristine images perform poorly when tested on distorted images affected by image blur or additive noise. Evaluating the effect of Gaussian blur and AWGN on the activations of convolutional filters trained on undistorted images, one can observe that select filters in each DNN convolutional layer are more susceptible to input distortions than the rest. This work proposes a novel objective metric to assess the susceptibility of convolutional filters to distortion and use this metric to identify the filters that maximize DNN robustness to input distortions, upon correction of their activations.

Correction units, which are *residual blocks* comprised of a small stack of trainable convolutional layers and a single skip connection per stack, are proposed for correction DNN filter activations. These *correction units* are added at the output of the most distortion susceptible filters in each convolutional layer, whilst leaving the rest of the pre-trained (on undistorted images) filter outputs in the network unchanged. The resultant DNN mod-

els which are referred to as *DeepCorrect* models, significantly improve the robustness of DNNs against image distortions and also outperform the alternative approach of network fine-tuning on common vision tasks like image classification, object recognition and scene classification, whilst training significantly less parameters and achieving a faster convergence in training. Fine-tuning limits the ability of the network to learn invariance to severe levels of distortion, and re-training an entire network can be computationally expensive for very deep networks. By correcting the most distortion-susceptible convolutional filter outputs, one is not only able to make a DNN robust to severe distortions, but is also able to maintain a very good performance on clean images.

Although this work focuses majorly on image classification and object recognition, the proposed approach is not only generic enough to apply to a wide selection of tasks that use DNN models such as object detection [13],[95] or semantic segmentation [16] but also other less commonly occurring noise types like adversarial noise.

Chapter 4

DEFENDING AGAINST UNIVERSAL ATTACKS THROUGH SELECTIVE FEATURE REGENERATION

In this chapter, a novel method is proposed to defend DNNs against universal adversarial perturbations by identifying and robustly regenerating the most adversarially-susceptible DNN filter activations, followed by extensive experimental validation of the proposed approach with different DNN architectures, attack norms and also other unseen universal attacks.

4.1 Introduction

Deep neural network (DNN) predictions have been shown to be vulnerable to carefully crafted adversarial perturbations. Specifically, image-agnostic (universal adversarial) perturbations added to any image can fool a target network into making erroneous predictions. Departing from existing defense strategies that work mostly in the image domain, this work presents a novel defense which operates in the DNN feature domain and effectively defends against such universal perturbations. The proposed approach identifies pre-trained convolutional features that are most vulnerable to adversarial noise and deploys trainable *feature regeneration units* which transform these DNN filter activations into resilient features that are robust to universal perturbations. Regenerating only the top 50% adversarially susceptible activations in at most 6 DNN layers and leaving all remaining DNN activations unchanged outperforms existing defense strategies across different network architectures by more than 10% in restored accuracy. Furthermore, without any additional modification, the proposed defense trained on ImageNet with only one type of universal attack examples effectively defends against other types of unseen universal attacks.

This chapter is organized as follows. A detailed description of the proposed approach is presented in Section 4.2, followed with extensive experimental validation in Section 4.3. Concluding remarks are provided in Section 4.4.

4.2 Feature-Domain Adversarial Defense

This section describes the proposed defense framework which first identifies convolutional filters whose activations are significantly disrupted by adversarial perturbations (Section 4.2.1) and then deploys *feature regeneration units* that transform disrupted features into resilient features that are robust to universal perturbations (Section 4.2.2). The *feature regeneration units* are trained using a target-oriented loss function with synthetic adversarial perturbations (Section 4.2.4), leaving all the parameters for the underlying baseline DNN unchanged.

4.2.1 Stability of Convolutional Filters

In this work, the vulnerability of individual convolutional filters is assessed and it is shown that, for each layer, certain filter activations are significantly more disrupted than others, especially in the early layers of a DNN.

For a given layer, let $\phi_m(u)$ be the output (activation map) of the m^{th} convolutional filter with kernel weights W_m for an input u. Let $e_m = \phi_m(u+r) - \phi_m(u)$ be the additive noise (perturbation) that is caused in the output activation map $\phi_m(u)$ as a result of applying an additive perturbation r to the input u. It can be shown (A) that e_m is bounded as follows:

$$\|e_m\|_\infty \leq \|W_m\|_1 \|r\|_p \tag{4.1}$$

where as before $\|\cdot\|_p$ is the ℓ_p-norm with $p \in [1,\infty)$. Equation 4.1 shows that the ℓ_1-norm of the filter weights can be used to identify and rank convolutional filter activations in terms of their ability to restrict perturbation in their activation maps. For example, filters

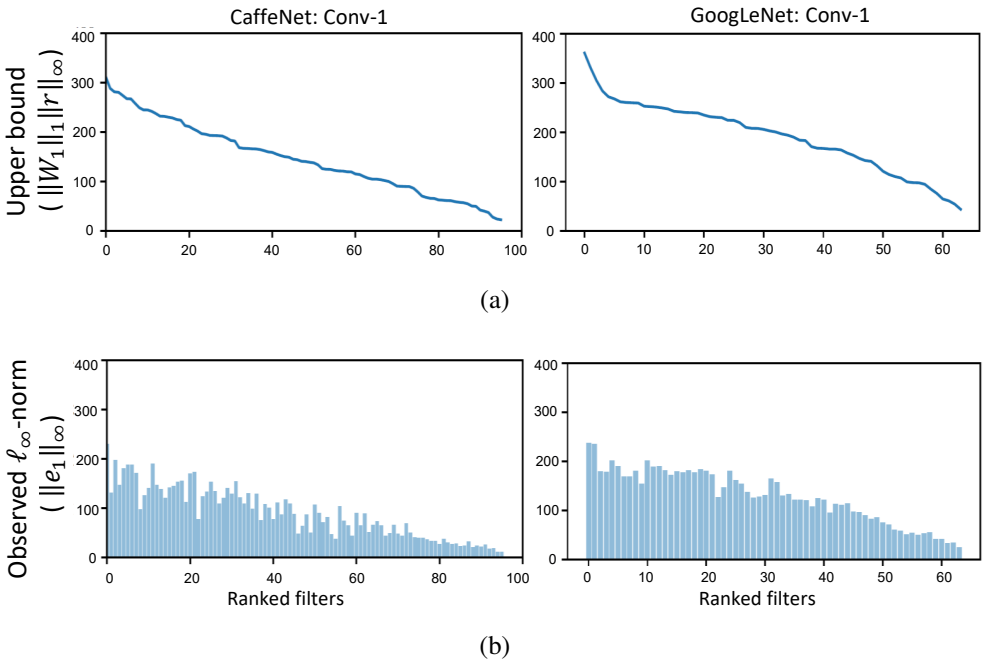


Figure 4.1: Observed ℓ_∞-norm for universal adversarial noise in the activation maps of ranked convolutional filters (ordered using the proposed ℓ_1-norm ranking measure, from most to least vulnerable) of the first layer of CaffeNet [8] and GoogLeNet [9]. The ℓ_∞-norm attack is used with $\xi \leq 10$, i.e. $\|\mathbf{r}\|_\infty \leq 10$. (a) Adversarial noise upper-bound (Equation 4.1) in ranked conv-1 filter activations of DNNs. (b) Observed ℓ_∞-norm for adversarial noise in ranked conv-1 filter activations of DNNs.

with a small weight ℓ_1-norm would result in insignificant small perturbations in their output when their input is perturbed, and are thus considered to be less vulnerable to perturbations in the input. For an ℓ_∞-norm universal adversarial input, Figure 4.1a shows the upper-bound on the adversarial noise in ranked (using the proposed ℓ_1-norm ranking) conv-1 filter activations of CaffeNet [8] and GoogLeNet [9], while Figure 4.1b shows the corresponding observed ℓ_∞-norm for adversarial noise in the respective DNN filter activations. One can see that the proposed $\|W\|_1$-based ranking correlates well with the degree of perturbation

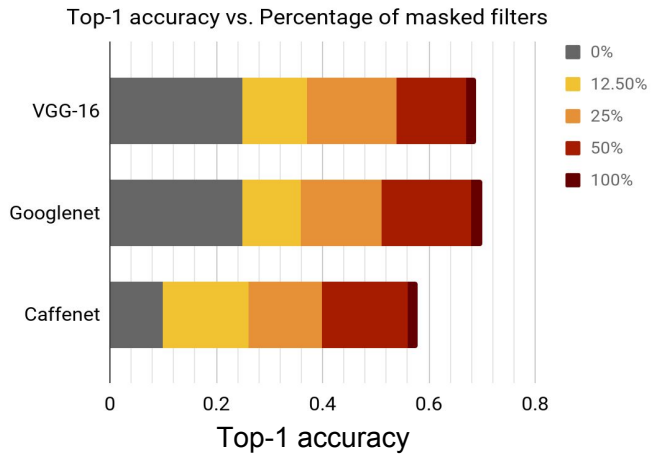


Figure 4.2: Effect of masking ℓ_∞-norm universal adversarial noise in ranked convolutional filter activations of the first layer in CaffeNet [8], GoogLeNet [9] and VGG-16 [10], evaluated on a 1000-image subset of the ImageNet [3] training set. Top-1 accuracies for perturbation-free images are 0.58, 0.70 and 0.69 for CaffeNet, GoogLeNet and VGG-16, respectively. Similarly, top-1 accuracies for adversarially perturbed images with no noise masking are 0.1, 0.25 and 0.25 for CaffeNet, GoogLeNet and VGG-16, respectively. Masking the noise in just 50% of the ranked filter activations restores most of the lost accuracy for all three DNNs.

(maximum magnitude of the noise perturbation) that is induced in the filter outputs. Similar observations can be made for other convolutional layers in the network.

Figure 4.2 shows the impact of masking the adversarial noise in such ranked filters on the overall top-1 accuracy of CaffeNet [8], VGG-16 [10] and GoogLeNet [9]. Specifically, a subset of 1000 images (1 image per class) is randomly chosen from the ImageNet [3] training set and adversarially perturbed images are generated by adding an ℓ_∞-norm universal adversarial perturbation [1]. The top-1 accuracies for perturbation-free images are 0.58, 0.70 and 0.69 for CaffeNet, GoogLeNet and VGG-16, respectively. Similarly, the top-1 accuracies for adversarially perturbed images of the same subset are 0.10, 0.25 and 0.25 for

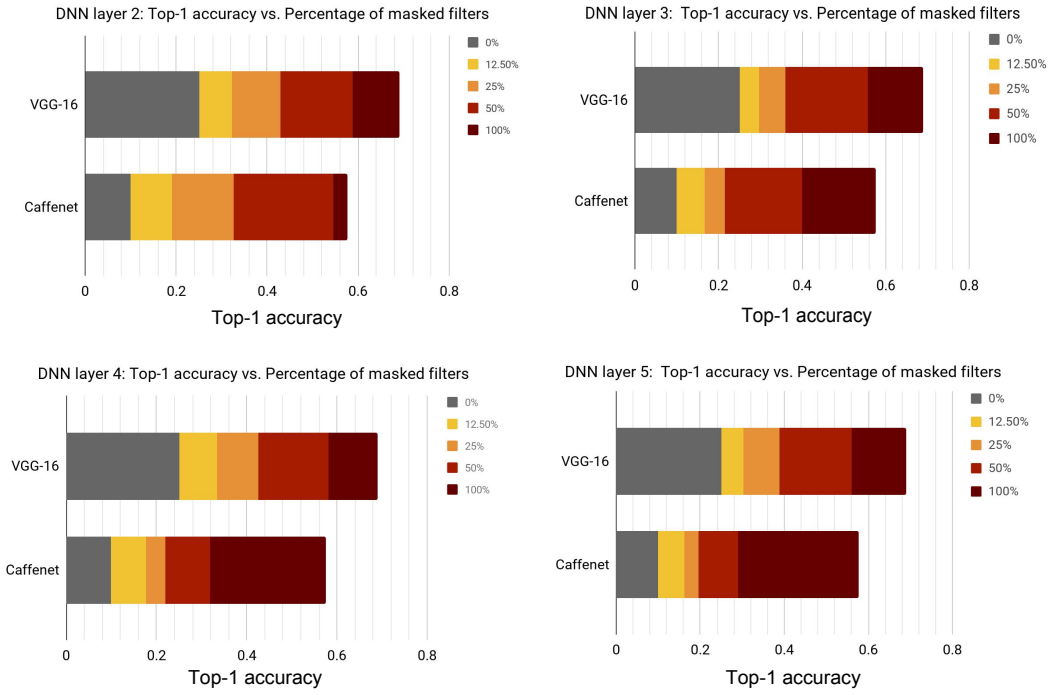


Figure 4.3: Effect of masking ℓ_∞-norm universal adversarial noise in ranked convolutional filter activations of layers 2-5 (starting from the input) in CaffeNet [8] and VGG-16 [10], evaluated on a 1000-image subset of the ImageNet [3] training set. Top-1 accuracies for perturbation-free images are 0.58, 0.69 for CaffeNet and VGG-16, respectively. Similarly, top-1 accuracies for adversarially perturbed images with no noise masking are 0.1 and 0.25 for CaffeNet and VGG-16, respectively. For VGG-16, masking the noise in just 50% of the ranked filter activations restores more than $\approx 80\%$ of the baseline accuracy on perturbation-free images.

CaffeNet, GoogLeNet and VGG-16, respectively. Masking the adversarial perturbations in 50% of the most vulnerable filter activations significantly improves DNN performance, resulting in top-1 accuracies of 0.56, 0.68 and 0.67 for CaffeNet, GoogLeNet and VGG-16, respectively, and validates our proposed selective feature regeneration scheme. Similarly, Figure 4.3, shows the effect of masking ℓ_∞-norm adversarial perturbations in ranked filter

activation maps of the convolutional layers 2, 3, 4 and 5 of CaffeNet [8] and VGG-16 [10]. For most DNN layers, masking the adversarial perturbations in just the top 50% most susceptible filter activation maps (identified by using the proposed ℓ_1-norm ranking measure) is able to recover most of the accuracy lost by the baseline DNN (Figure 4.3). Specifically, masking the adversarial perturbations in the top 50% ranked filters of VGG-16 is able to restore at least 84% of the baseline accuracy on perturbation-free images.

4.2.2 Resilient Feature Regeneration Defense

The proposed defense is illustrated in Figure 4.4 and seeks to learn a task-driven feature restoration transform (i.e., *feature regeneration unit*) for convolutional filter activations severely disrupted by adversarial input. The *feature regeneration unit* does not modify the remaining activations of the baseline DNN. Compared to feature denoising (FD) [37], the proposed feature regeneration approach has the following differences: (1) *Feature regeneration units* are not restricted to only perform denoising, but consists of stacks of trainable convolutional layers that provide this proposed defense the flexibility to learn an appropriate feature-restoration transform that effectively defends against universal attacks, unlike the non-local mean denoiser used in FD; (2) in a selected DNN layer, only a subset of feature maps which are the most susceptible to adversarial noise (identified by our ranking metric) are regenerated leaving all other feature maps unchanged, whereas FD denoises all feature maps, which can result in over-correction or introduce unwanted artifacts in feature maps that admit very low magnitude noise; (3) instead of adversarially training all the parameters of the baseline DNN as in FD, this defense only trains the parameters in the *feature renegeration units* (up to 90% less parameters than a baseline DNN, see Table 4.1) and leaves all parameters in the baseline DNN unchanged, which can speed up training, reduce the risk of over-fitting and also improve robustness to other unseen attacks. A simi-

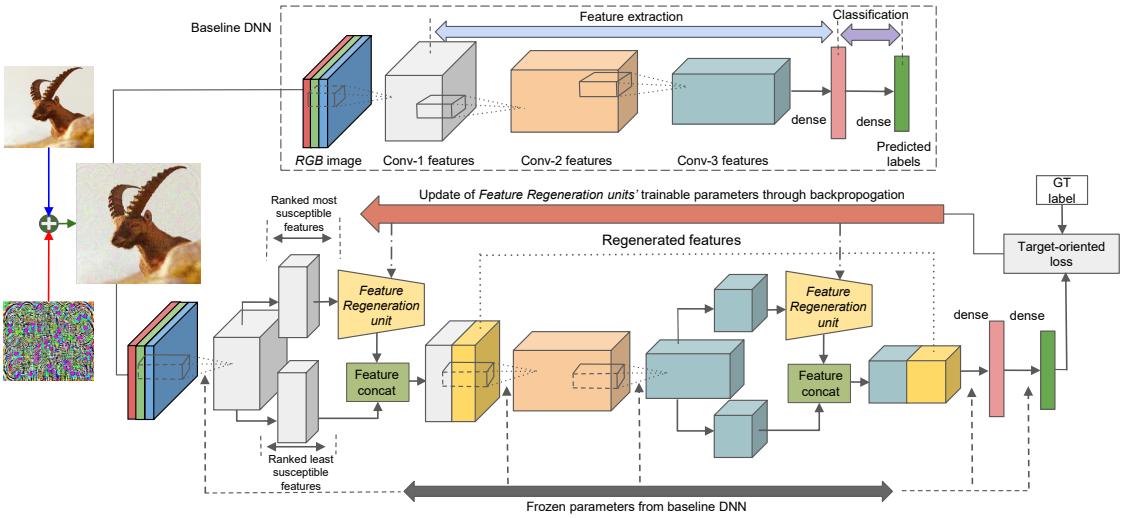


Figure 4.4: Resilient Feature Regeneration Defense: Convolutional filter activations in the baseline DNN (top) are first sorted in order of vulnerability to adversarial noise using their respective filter weight norms (Section 4.2.1). For each considered layer, a *feature regeneration unit* (Figure 4.7) regenerates only the most adversarially susceptible activations into resilient features that restore the lost accuracy of the baseline DNN, while leaving the remaining filter activations unchanged. *Feature regeneration units* are trained on both clean and perturbed images in every mini-batch using the same target loss as the baseline DNN such that all parameters of the baseline DNN are left unchanged during training.

lar approach of learning corrective transforms for making networks more resilient to image blur and additive white Gaussian noise has been explored in [96].

Let S_l represent a set consisting of indices for convolutional filters in the l^{th} layer of a DNN. Furthermore, let $S_{l_{reg}}$ be the set of indices for filters one wishes to regenerate (Section 4.2.1) and let $S_{l_{adv}}$ be the set of indices for filters whose activations are not regenerated (i.e., $S_l = S_{l_{reg}} \cup S_{l_{adv}}$). If $\Phi_{S_{l_{reg}}}$ represents the convolutional filter outputs to be regenerated in the l^{th} layer, then the *feature regeneration unit* in layer l performs a feature regeneration

transform $\mathscr{D}_l(\cdot)$ under the following conditions:

$$\mathscr{D}_l(\Phi_{S_{l_{reg}}}(u+r)) \approx \Phi_{S_{l_{reg}}}(u) \tag{4.2}$$

and

$$\mathscr{D}_l(\Phi_{S_{l_{reg}}}(u)) \approx \Phi_{S_{l_{reg}}}(u) \tag{4.3}$$

where u is the unperturbed input to the l^{th} layer of convolutional filters and r is an additive perturbation that acts on u. In Equations 4.2 and 4.3, $\approx$ denotes similarity based on classification accuracy in the sense that features are restored to regain the classification accuracy of the original perturbation-free activation map. Equation 4.2 forces $\mathscr{D}_l(\cdot)$ to pursue task-driven feature regeneration that restores lost accuracy of the DNN while Equation 4.3 ensures that prediction accuracy on unperturbed activations is not decreased, without any additional adversarial perturbation detector. $\mathscr{D}_l(\cdot)$ (i.e., *feature regeneration unit*) is implemented as a shallow *residual block* [7], consisting of two stacked 3×3 convolutional layers sandwiched between a couple of 1×1 convolutional layers and a single skip connection (Figure 4.7). $\mathscr{D}_l(\cdot)$ is estimated using a target loss from the baseline network, through backpropagation, see Figure 4.4, but with significantly fewer trainable parameters compared to the baseline network (Table 4.1).

Given an L layered DNN Φ, pre-trained for an image classification task, Φ can be represented as a function that maps network input x to an N-dimensional output label vector $\Phi(x)$ as follows:

$$\Phi = \Phi_L \circ \Phi_{L-1} \circ \ldots \Phi_2 \circ \Phi_1 \tag{4.4}$$

where Φ_l is a mapping function (set of convolutional filters, typically followed by a non-linearity) representing the l^{th} DNN layer and N is the dimensionality of the DNN's output (i.e., number of classes). Without any loss of generality, the resulting DNN after deploying a *feature regeneration unit* that operates on the set of filters represented by $S_{l_{reg}}$ in layer l is

given by:

$$\Phi_{reg} = \Phi_L \circ \Phi_{L-1} \circ \ldots \Phi_{l_{reg}} \ldots \Phi_2 \circ \Phi_1 \tag{4.5}$$

where $\Phi_{l_{reg}}$ represents the new mapping function for layer l, such that $\mathscr{D}_l(\cdot)$ regenerates only activations of the filter subset $\Phi_{S_{l_{reg}}}$ and all the remaining filter activations (i.e., $\Phi_{S_{l_{adv}}}$) are left unchanged. If $\mathscr{D}_l(\cdot)$ is parameterized by θ_l, then the *feature regeneration unit* can be trained by minimizing:

$$\mathscr{J}(\theta_l) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}(y_k, \Phi_{reg}(x_k)) \tag{4.6}$$

where $\mathcal{L}$ is the same target loss function of the baseline DNN (e.g., cross-entropy classification loss), y_k is the target output label for the k^{th} input image x_k, K represents the total number of images in the training set consisting of both clean and perturbed images. As both clean and perturbed images are used during training, x_k in Equation 4.6, represents a clean or an adversarially perturbed image. Figure 4.5 visualizes DNN feature maps perturbed by various universal perturbations and the corresponding feature maps regenerated by the *feature regeneration units*, which are only trained on UAP [1] attack examples. Compared to the perturbation-free feature map (clean), corresponding feature maps for adversarially perturbed images (Row 1) have distinctly visible artifacts that reflect the universal perturbation pattern in major parts of the image. In comparison, feature maps regenerated by the proposed *feature regeneration units* (Row 2) effectively suppress these adversarial perturbations, preserve the object discriminative attributes of the clean feature map and are also robust to unseen attacks (e.g, NAG [53], GAP [55] and sPGD [2]), as illustrated in Figure 4.5 and Table 4.6. Similarly, additional high resolution visualizations of DNN feature maps before and after resilient feature regeneration using the proposed defense framework are shown in Figure 4.6.

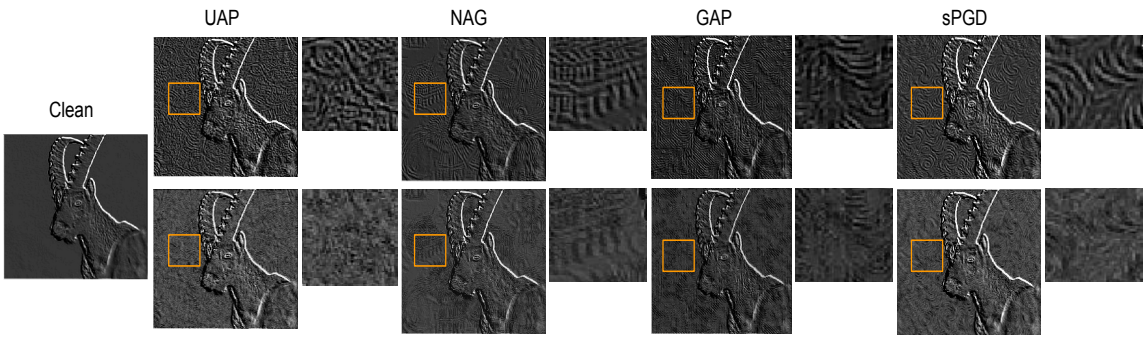


Figure 4.5: Effectiveness of *feature regeneration units* at masking adversarial perturbations in DNN feature maps for images perturbed by universal perturbations (UAP [1], NAG [53], GAP [55] and sPGD [2]). Perturbation-free feature map (clean), different adversarially perturbed feature maps (Row 1) and corresponding feature maps regenerated by *feature regeneration units* (Row 2) are obtained for a single filter channel in conv1_1 layer of VGG-16 [10], along with an enlarged view of a small region in the feature map (yellow box). *Feature regeneration units* are only trained on UAP [1] attack examples but are very effective at suppressing adversarial artifacts generated by unseen attacks (e.g., NAG [53], GAP [55] and sPGD [2]).

4.2.3 Design of Feature Regeneration Unit

A defense that only alters the existing weights of a network has to effectively relearn all its features to not only be discriminative but also be adversarially robust, whereas my motivation is to preserve the original discriminative (adversarially susceptible) features of the network and simply augment its knowledge with additional simple restoration transforms (adversarial robustness) learnt by the *feature regeneration units*. The *feature regeneration unit* architecture adopted in this work (Figure 4.7) is guided by the following constraints: 1) have adequate capacity to learn an effective restoration transform but also minimize the number of additional trainable parameters added, 2) be simple to implement and generic enough to be readily integrated into all existing DNN architectures. With these constraints,

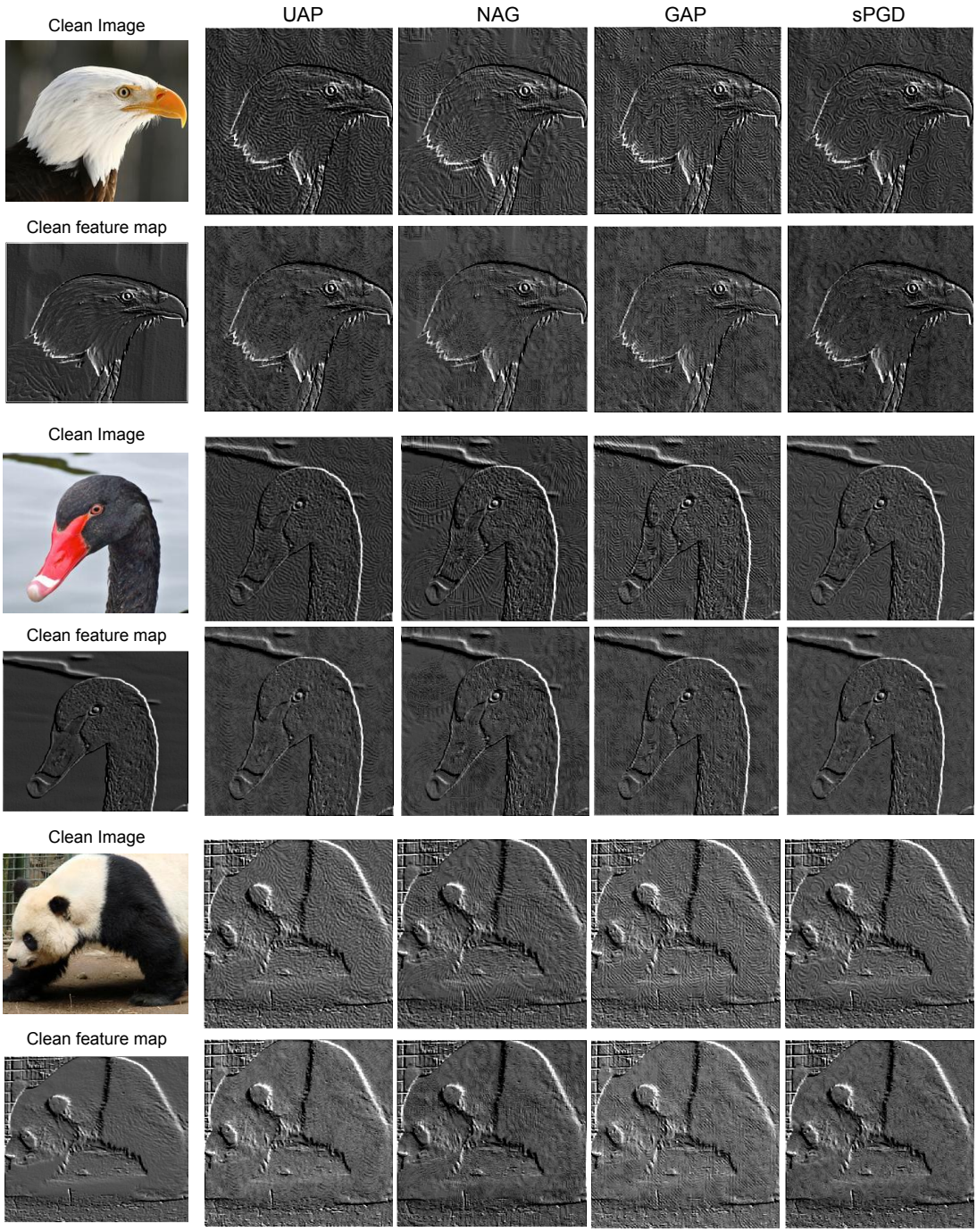


Figure 4.6: Examples of DNN feature maps before and after *feature regeneration* using the proposed method. Perturbation-free feature map (clean feature map), different adversarially perturbed feature maps (Rows 1, 3 and 5) and corresponding feature maps regenerated by *feature regeneration units* (Rows 2, 4 and 6) are obtained for a single filter channel in conv1_1 layer of VGG-16 [10].

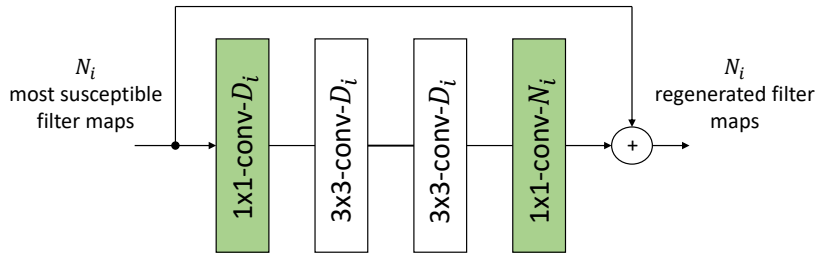


Figure 4.7: *Feature regeneration unit* acting on the activations of the N_i most susceptible filters in layer i of a DNN. D_i represents the *feature regeneration unit* kernel depth and has a default value of N_i. All convolutional layers except the final 1×1 layer are also followed by batch normalization and a ReLU non-linearity. # parameters per *feature regeneration unit* $\approx 18D_i^2 + 2N_iD_i$.

a shallow residual block such as the one shown in Figure 4.7 works well. Using 1×1 convolutions as the input and output mapping layers of the *feature regeneration unit* also enables it to adopt a bottleneck architecture (i.e. $D_i < N_i$ in Figure 4.7) for resource-constrained environments. The proposed *feature regeneration unit* architecture is not the only possible solution and other more complex sub-networks could also be incorporated into the proposed defense as an interesting direction for future research. As the proposed defense formulation does not rely on assumptions about the exact architecture of the *feature regeneration unit* except that it has adequate capacity, the defense is expected to be insensitive to the specific choice of the *feature regeneration unit* architecture. Table 4.1 shows the total number of parameters used by the *feature regeneration units* for each evaluated DNN. Compared to the number of trainable parameters in the baseline DNN, the *feature regeneration units* require far less parameters.

4.2.4 Generating Synthetic Perturbations

Training-based approaches are susceptible to data overfitting, especially when the training data is scarce or does not have adequate diversity. Generating a diverse set of adver-

Table 4.1: Trainable parameters for DNN models.

Methods	CaffeNet	VGG-F	GoogLeNet	VGG-16	Res152
Baseline	61M	61M	6.9M	131M	60M
Ours (FRUs)	2.2M	1.7M	1.4M	1.6M	2.5M

sarial perturbations (≥ 100) using existing attack algorithms (e.g., [1, 53, 55, 2]), in order to avoid overfitting, can be computationally prohibitive. A fast method (Algorithm 2) is proposed to construct synthetic universal adversarial perturbations from a small set of adversarial perturbations, $V \subseteq \mathbb{R}^d$, that is computed using any existing universal attack generation method ([1, 53, 55, 2]). Starting with the synthetic perturbation v_{syn} set to zero, at each iteration t, a random perturbation $v_{new} \in V$ and a random scale factor $\alpha \in [0,1]$ are selected and v_{syn} is updated as follows:

$$v_{syn}(t) = \alpha v_{new} + (1-\alpha)v_{syn}(t-1) \tag{4.7}$$

This process is repeated until the ℓ_2-norm of v_{syn} exceeds a threshold η. The threshold η is set to be the minimum ℓ_2-norm of perturbations in the set V.

Unlike the approach of Akhtar *et al.* [52], which uses an iterative random walk along pre-computed adversarial directions, the proposed algorithm has two distinct advantages: 1) the same algorithm can be used for different types of attack norms without any modification, and 2) Equation 4.7 (Step 5 in Algorithm 2) automatically ensures that the ℓ_∞-norm of the perturbation does not violate the constraint for an ℓ_∞-norm attack (i.e., ℓ_∞-norm $\leq \xi$) and, therefore, no additional steps, like computing a separate perturbation unit vector and ensuring that the resultant perturbation strength is less than ξ, are needed.

Sample visualizations of synthetic adversarial perturbations generated using the algorithm proposed in Section 4.2.4 (Algorithm 2) are provided in Figure 4.8.

Algorithm 2 Generating Synthetic Adversarial Perturbation

Input: Set of pre-computed perturbations $V \subseteq \mathbb{R}^d$ such that $v_i \in V$ is the i^{th} perturbation and η is an adjustable threshold

Output: Synthetic perturbation $v_{syn} \in \mathbb{R}^d$

$v_{syn} = 0$

while $\|\mathbf{v}_{syn}\|_2 \leq \eta$ **do**

    $\alpha \sim \text{uniform}(0,1)$

    $v_{new} \overset{\text{rand}}{\sim} V$

    $v_{syn} = \alpha v_{new} + (1-\alpha) v_{syn}$

end while

return v_{syn}

4.3 Assessment

The ImageNet validation set (ILSVRC2012) [3] with all 50000 images and a single crop evaluation (unless specified otherwise) is used in all experiments. All experiments are implemented using Caffe [11] and for each tested attack the publicly provided code is used. Results are reported in terms of top-1 accuracy and the restoration accuracy proposed by Akhtar *et al.* [52]. Given a set I_c containing clean images and a set $I_{p/c}$ containing clean and perturbed images in equal numbers, the restoration accuracy is given by:

$$\text{Restoration accuracy} = \frac{\text{acc}(I_{p/c})}{\text{acc}(I_c)} \tag{4.8}$$

where $\text{acc}(\cdot)$ is the top-1 accuracy and $\text{acc}(I_c)$ refers to the top-1 accuracy of the baseline DNN (no defense) on I_c. The universal adversarial perturbation (UAP) attack [1] is used for evaluation (unless specified otherwise) and 5 independent universal adversarial test perturbations per network are computed using a set of 10000 held out images randomly chosen from the ImageNet training set with the fooling ratio for each perturbation lower-bounded

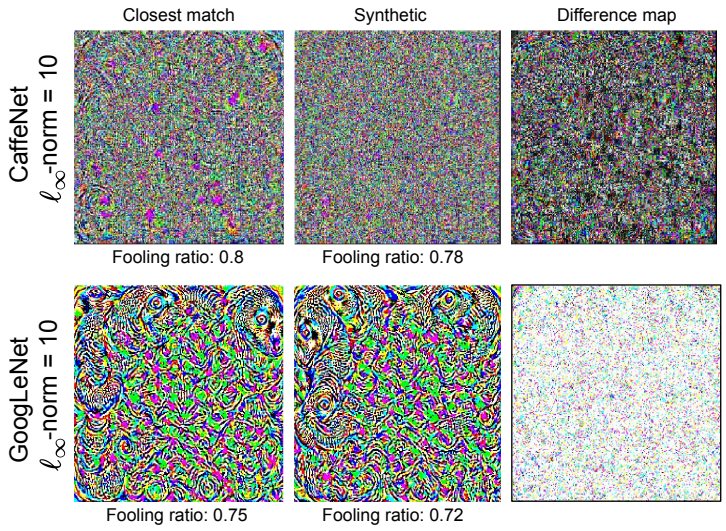


Figure 4.8: Visualization of synthetic perturbations (center) computed for CaffeNet [8] and GoogLeNet [9] along with their closest match in the set of original perturbations (left) and a per pixel difference map between the two (right). Closest match is identified as the the perturbation in the original set V which produces the smallest pixel-wise mean squared error with the synthetic perturbation.

to 0.8 on the held out images and the maximum normalized inner product between any two perturbations for the same DNN upper-bounded to 0.15. Sample visualizations of ℓ_∞-norm and ℓ_2-norm UAP [1] attack perturbations used for testing are shown in Figure 4.9.

4.3.1 Defense Training Methodology

In this proposed defense (Figure 4.4), only the parameters for *feature regeneration units* have to be trained and these parameters are updated to minimize the cost function given by Equation 4.6. Although one can expect the prediction performance of defended models to improve with higher regeneration ratios (i.e., fraction of convolutional filter activations regenerated), only 50% of the convolutional filter activations in a layer are regenerated and the number of deployed *feature regeneration units* (1 per layer) is limited to

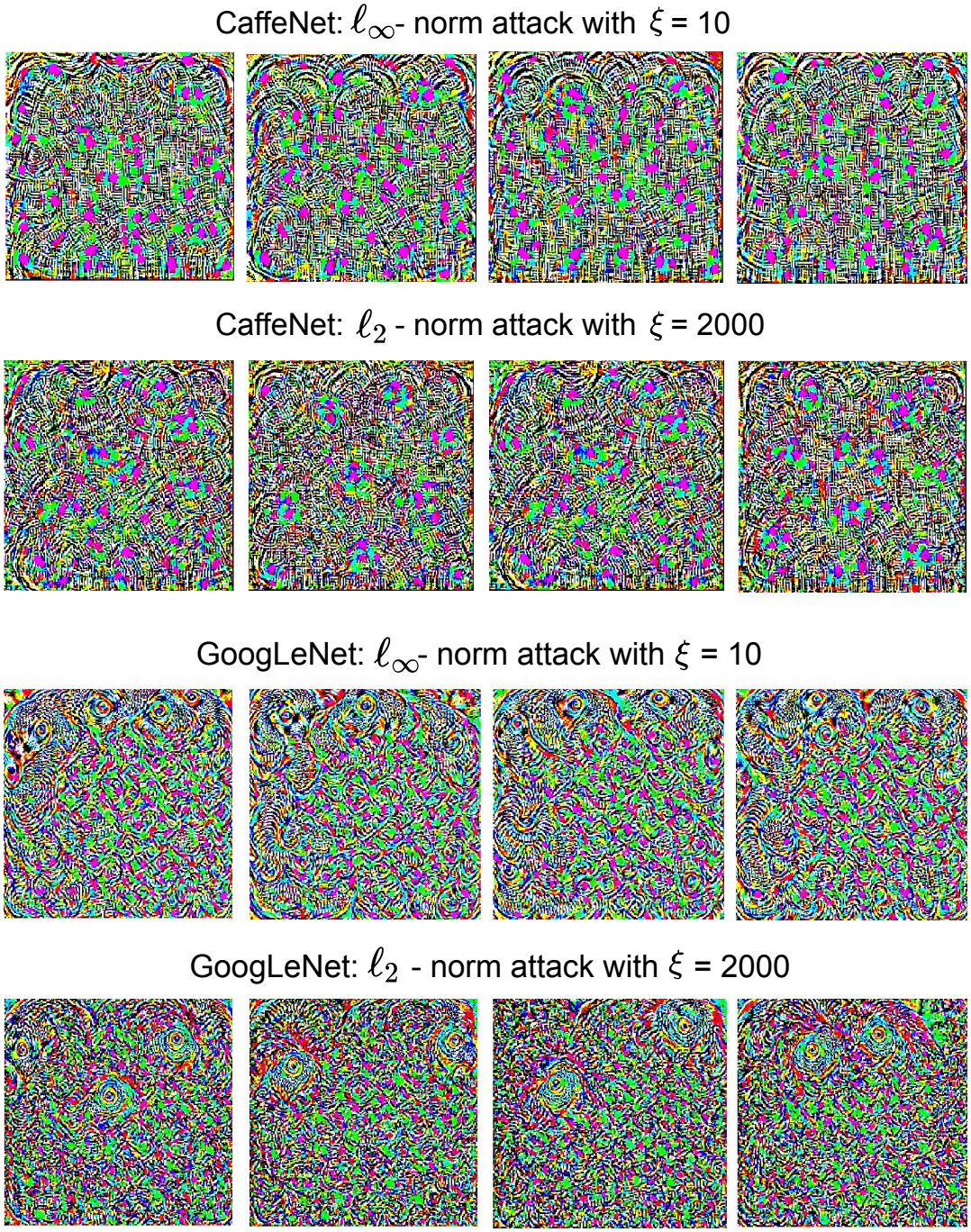


Figure 4.9: Visual examples of ℓ_∞-norm and ℓ_2-norm UAP [1] attack test perturbations for CaffeNet [8] and GoogLeNet [9].

$\min(\#\text{DNN layers}, 6)$. From Figure 4.2 and Figure 4.3, one can observe that an empirical regeneration ratio of 50% works well. Similarly, although *feature regeneration units* can be deployed for each layer in a DNN, it can be shown (Section 4.3.2 and Figure 4.10) that regenerating features in at most 6 layers in a DNN effectively recovers lost prediction performance. Using Algorithm 2, 2000 synthetic perturbations are generated from a set V of 25 original perturbations [1] and *feature regeneration units* are trained using a single Nvidia Titan-X using a standard SGD optimizer, momentum of 0.9 and a weight decay of 0.0005 for 4 epochs of the ImageNet training set [3]. The learning rate is dropped by a factor of 10 after each epoch with an initial learning rate of 0.1. After a defense model has been trained as outlined above, one can further iterate through the training of this defense with additional adversarial perturbations computed against the proposed defense, which ensures robustness to secondary attacks against this proposed defense (Section 4.3.2).

4.3.2 Analysis and Comparisons

Robustness Across DNN Architectures

Top-1 accuracy of adversarially perturbed test images for various DNNs (no defense) and the proposed defense for respective DNNs is reported in Table 4.2 under both white-box (same network used to generate and test attack) and black-box (tested network is different from network used to generate attack) settings. As universal adversarial perturbations can be *doubly universal*, under a black-box setting, a target DNN defense (defense is trained for attacks on target DNN) is evaluated against a perturbation generated for a different network. Top-1 accuracy for baseline DNNs is severely affected by both white-box and black-box attacks, whereas the proposed defense is not only able to effectively thwart the white-box attacks but is also able to generalize to attacks constructed for other networks without further training (Table 4.2). Since different DNNs can share common adversarial

Table 4.2: Cross-DNN evaluation on ILSVRC2012: Top-1 accuracy against a ℓ_∞-norm UAP [1] attack with $\xi = 10$ and target fooling ratio of 0.8. DNNs in column one are tested with attacks generated for DNNs in row one.

	CaffeNet	**VGG-F**	**GoogleNet**	**VGG-16**	**Res152**
CaffeNet [8], orginal accuracy 56.4%					
CaffeNet	0.109	0.298	0.456	0.431	0.405
Ours	**0.542**	**0.524**	**0.510**	**0.457**	**0.470**
VGG-F [86], original accuracy 58.4%					
VGG-F	0.299	0.150	0.461	0.417	0.426
Ours	**0.556**	**0.550**	**0.548**	**0.492**	**0.513**
GoogLeNet [9], original accuracy 68.6%					
GoogLeNet	0.519	0.539	0.260	0.472	0.473
Ours	**0.651**	**0.653**	**0.653**	**0.637**	**0.642**
VGG-16 [10], original accuracy 68.4%					
VGG-16	0.549	0.559	0.519	0.240	0.484
Ours	**0.615**	**0.622**	**0.646**	**0.655**	**0.631**
Res152 [7], original accuracy 79%					
Res152	0.720	0.726	0.692	0.626	0.270
Ours	**0.764**	**0.769**	**0.769**	**0.763**	**0.761**

directions in their feature space, the proposed *feature regeneration units* learn to regularize such directions against unseen data and, consequently, to defend against black-box attacks.

Robustness Across Attack Norms

Here, defense robustness is evaluated against both ℓ_∞-norm and ℓ_2-norm UAP [1] attacks. Since an effective defense must not only recover the DNN accuracy against adversarial images but must also maintain a high accuracy on clean images, restoration accuracy (Equa-

Table 4.3: Same-norm evaluation on ILSVRC2012: Restoration accuracy of DNNs and defenses against an ℓ_∞-norm UAP [1] attack with $\xi = 10$.

Methods	**CaffeNet**	**VGG-F**	**GoogLeNet**	**VGG-16**	**Res152**
ℓ_∞-norm attack, $\xi = 10$					
Baseline	0.596	0.628	0.691	0.681	0.670
PRN [52]	0.936	0.903	0.956	0.690	0.834
PRN+det [52]	0.952	0.922	0.964	0.690	0.834
PD [40]	0.873	0.813	0.884	0.818	0.845
JPEG comp. [41]	0.554	0.697	0.830	0.693	0.670
Feat. Distill. [44]	0.671	0.689	0.851	0.717	0.676
HGD [45]	n/a	n/a	n/a	n/a	0.739
Adv. tr. [22]	n/a	n/a	n/a	n/a	0.778
FD [37]	n/a	n/a	n/a	n/a	0.819
Ours	**0.976**	**0.967**	**0.970**	**0.963**	**0.982**

tion 4.8) is used to measure adversarial defense robustness (Tables 4.3 and 4.4). While Akhtar *et al.* [52] (PRN and PRN+det) only report defense results on the UAP [1] attack, the obtained results are also compared with pixel-domain defenses such as Pixel Deflection (PD [40]) and High Level Guided Denoiser (HGD [45]), defenses that use JPEG compression (JPEG comp. [41]) or DNN-based compression like Feature Distillation (Feat. Distill. [44]), defenses that use some variation of adversarial training like Feature Denoising (FD [37]) and standard Adversarial training (Adv. tr. [22]).

In Table 4.3, results are reported for an ℓ_∞-norm UAP attack [1] against various DNNs and show that the proposed defense outperforms all the other defenses [1] for all networks with the highest restoration accuracy (98.2%) being achieved for Res152 [7]. The *feature*

[1]FD [37], HGD [45] and Adv. tr. [22] defenses publicly provide trained defense models only for Res152 [7]) among the evaluated DNNs; results are reported using only the DNN models provided by the respective authors.

Table 4.4: Cross-norm evaluation on ILSVRC2012: Restoration accuracy against an ℓ_2-norm UAP [1] attack. The proposed defense, as well as the other defense models, are trained only on ℓ_∞-norm attack examples with $\xi = 10$.

Methods	**CaffeNet**	**VGG-F**	**GoogLeNet**	**VGG-16**	**Res152**
ℓ_2-norm attack, ξ = 2000					
Baseline	0.677	0.671	0.682	0.697	0.709
PRN [52]	0.922	0.880	0.971	0.834	0.868
PRN+det [52]	0.936	0.900	**0.975**	0.835	0.868
PD [40]	0.782	0.784	0.857	0.809	0.840
HGD [45]	n/a	n/a	n/a	n/a	0.730
Adv. tr. [22]	n/a	n/a	n/a	n/a	0.778
FD [37]	n/a	n/a	n/a	n/a	0.818
Ours	**0.964**	**0.961**	0.912	**0.876**	**0.926**

regeneration units are trained on ℓ_∞-norm attack examples (same-norm evaluation). Even without a perturbation detector, the proposed defense outperforms the existing defense with a perturbation detector (PRN+det) of Akhtar *et al.* [52] for all networks. Similarly, for Res152 [7], the proposed defense outperforms adversarially trained defenses (FD [37], Adv. tr. [22]) and pixel denoisers (PD [40], HGD [45]) by more than 10%. Table 4.4 also evaluates how well the proposed defense trained on an ℓ_∞-norm attack defends against an ℓ_2-norm attack (cross-norm evaluation). The *feature regeneration units* are able to effectively generalize to even cross-norm attacks and outperform all other defenses for most DNNs.

Feature Regeneration Unit: An Ablation Study

In general, *feature regeneration units* can be added at the output of each convolutional layer in a DNN. However, this may come at the cost of increased computations, due to an

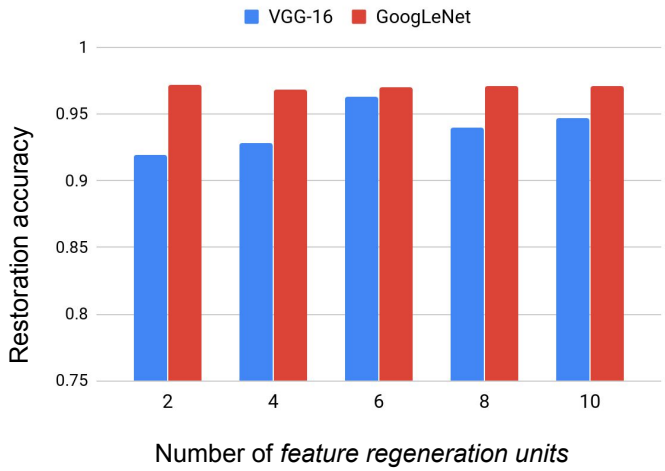


Figure 4.10: Effect of adding *feature regeneration units* on the restoration accuracy of the proposed defense. Adding just two *feature regeneration units* in GoogLeNet [9] achieves a restoration accuracy of 97% and adding more *feature regeneration units* to the DNN does not improve results any further. For VGG-16 [10], adding 6 *feature regeneration units* provides best results.

increase in the number of DNN parameters. As mentioned in Section 4.3.1, the number of *feature regeneration units* added to the DNN is constrained in order to avoid drastically increasing the training and inference cost for larger DNNs (i.e., VGG-16, GoogLeNet and ResNet-152). Here, an ablation study is performed to identify the least number of *feature regeneration units* needed to at least achieve a 95% restoration accuracy across most DNNs. Specifically, VGG-16 [10] and GoogLeNet [9] are used for this analysis. The restoration accuracy is evaluated on the ImageNet [3] validation set (ILSVRC2012) by adding an increasing number of *feature regeneration units*, starting from a minimum value of 2 towards a maximum value of 10 in steps of 2. Starting from the first convolutional layer in a DNN, each additional *feature regeneration unit* is added at the output of every second convolutional layer. Figure 4.10 reports the results of this ablation study and it can be observed that for GoogLeNet, adding just two *feature regeneration units* achieves a restoration accu-

Table 4.5: Restoration accuracy on ILSVRC2012 for ℓ_∞-norm UAP [1] attack with stronger perturbation strengths (ξ) against CaffeNet. The proposed defense, as well as the other defense models, are trained only on ℓ_∞-norm attack examples with ξ=10.

Method	$\xi = 10$	$\xi = 15$	$\xi = 20$	$\xi = 25$
Baseline	0.596	0.543	0.525	0.519
PRN [52]	0.936	0.603	0.555	0.526
PRN+det [52]	0.952	0.604	0.555	0.526
PD [40]	0.873	0.616	0.549	0.524
Ours	**0.976**	**0.952**	**0.896**	**0.854**

racy of 97% and adding any more *feature regeneration units* does not have any significant impact on the restoration accuracy. However, for VGG-16, adding only 2 *feature regeneration units* achieves a restoration accuracy of only 91%. For VGG-16, adding more *feature regeneration units* improves the performance with the best restoration accuracy of 96.2% achieved with 6 *feature regeneration units*. Adding more than 6 *feature regeneration units* resulted in a minor drop in restoration accuracy and this may be due to data over-fitting. As a result, the number of *feature regeneration units* deployed for any DNN is restricted to $\min(\#\text{DNN layers}, 6)$.

Stronger Attack Perturbations ($\xi > 10$)

Although an attack perturbation strength ξ = 10 is used during training, in Table 4.5, the robustness of the proposed defense is evaluated for the case when the adversary violates the attack threat model using a higher perturbation strength. Compared to the baseline DNN (no defense) as well as PRN [52] and PD [40], the proposed defense is much more effective at defending against stronger perturbations, outperforming other defenses by almost 30% even when the attack strength is more than double the value used to train our defense. Al-

Table 4.6: Robustness to unseen attacks: Restoration accuracy evaluated on ILSVRC2012, against other unseen universal attacks using our defense trained on just ℓ_∞-norm UAP [1] attack examples with a fooling ratio and ℓ_∞-norm of 0.8 and 10, respectively. Results for all other defenses are reported using publicly provided defense models. Attacks are constructed for the baseline DNN.

	CaffeNet			**Res152**		
Methods	**FFF** [54]	**NAG** [53]	**S.Fool** [56]	**GAP** [55]	**G-UAP** [58]	**sPGD** [2]
Baseline	0.645	0.670	0.815	0.640	0.726	0.671
PRN [52]	0.729	0.660	0.732	0.774	0.777	0.823
PD [40]	0.847	0.767	0.871	0.784	0.807	0.890
HGD [45]	n/a	n/a	n/a	0.663	0.782	0.932
Adv. tr [22]	n/a	n/a	n/a	0.776	0.777	0.775
FD [37]	n/a	n/a	n/a	0.815	0.813	0.815
Ours	**0.941**	**0.840**	**0.914**	**0.922**	**0.914**	**0.976**

though defense robustness decreases for unseen higher perturbation strengths, the proposed defense handles this drop-off much more gracefully and shows much better generalization across attack perturbation strengths, as compared to existing defenses. Furthermore, one can also see that adversarial perturbations are no longer visually imperceptible at $\xi = 25$.

Generalization to Unseen Universal Attacks

Although the proposed method effectively defends against UAP [1] attacks (Tables4.2-4.5), its robustness to other unseen universal attacks is also assessed without additional attack-specific training. Note that [52] and [2] do not cover this experimental setting. Since existing attacks in the literature are tailored to specific DNNs, CaffeNet [8] and Res152 [7] DNNs are used for covering a variety of universal attacks like Fast Feature Fool (FFF) [54], Network for adversary generation (NAG) [53], Singular fool (S.Fool) [56], Generative ad-

versarial perturbation (GAP) [55], Generalizable data-free universal adversarial perturbation (G-UAP) [58], and stochastic PGD (sPGD) [2].

The proposed defense trained on just UAP [1] attack examples is able to effectively defend against all other universal attacks and outperforms all other existing defenses (Table 4.6). Even against stronger universal attacks like NAG [53] and GAP [55], the proposed defense outperforms all other defenses including PRN [52], which is also trained on similar UAP [1] attack examples, by almost 10%. From the results in Table 4.6, one can see that the proposed *feature regeneration units* learn transformations that generalize effectively across perturbation patterns (Figure 4.5). Note that this work is the first to show such broad generalization across universal attacks.

Robustness to Attacks Using Surrogate Defense DNNs

The work in this section evaluates if it is possible for an attacker/adversary to construct a surrogate defense network if it was known that this proposed defense was adopted. In situations where exact defense (*feature regeneration units* + baseline DNN) is typically hidden from the attacker (*oracle*), a DNN predicting output labels similar to the defense (*surrogate*) can be useful to generate new attacks only if an attack generated using the *surrogate* is transferable to the *oracle*. UAP [1] attacks are transferable across baseline DNNs (Table 4.2), i.e., adversarial perturbation computed for a DNN whose model weights and architecture are known (surrogate) can also effectively fool another target DNN that has a similar prediction accuracy, but whose model weights and architecture are not known to the attacker (oracle). Assuming that the proposed defense (*feature regeneration units* + baseline DNN) for CaffeNet [8] and Res152 [7] is available publicly as a *surrogate*, universal attack examples computed from these DNNs may be used to attack proposed defenses for other DNNs, e.g. VGG-F or VGG-16 as an *oracle*. Table 4.7 shows that the proposed de-

Table 4.7: Defense restoration accuracy for oracle DNNs equipped with our defense for an ℓ_∞-norm UAP [1] attack ($\xi = 10$) using surrogate defense DNNs equipped with our defense.

Surrogate	**Oracle**		
	VGG-F + defense	**GoogLeNet + defense**	**VGG-16 + defense**
CaffeNet + defense	0.906	0.963	0.942
Res152 + defense	0.889	0.925	0.925

fense mechanism successfully breaks attack transferability and is not susceptible to attacks from *surrogate* DNNs that are based on the proposed defense.

Robustness to Secondary White-Box Attacks

Although in practical situations, an attacker may not have full or even partial knowledge of a defense, for completeness, the proposed defense is evaluated against a white-box attack on the defense (secondary attacks), i.e., adversary has full access to the gradient information of the *feature regeneration units*. The UAP [1] attack (on CaffeNet) and sPGD [2] (on Res152) attacks are used for this evaluation.

Figure 4.11 shows the robustness of the proposed defense to such a secondary UAP [1] attack seeking to achieve a target fooling ratio of 0.85 on it for the CaffeNet [8] DNN. Such an attack can easily converge (achieve target fooling ratio) against a baseline DNN in less than 2 attack epochs, eventually achieving a final fooling ratio of 0.9. Similarly, one can observe that even PRN [52] is susceptible to a secondary UAP [1] attack, achieving a fooling ratio of 0.87, when the adversary can access gradient information for its *Perturbation Rectifying Network*. In comparison, using the proposed defense model with iterative adversarial example training (as described in Section 4.3.1), the white-box adversary can achieve a maximum fooling ratio of only 0.42, which is 48% lower than the fooling ratio achieved

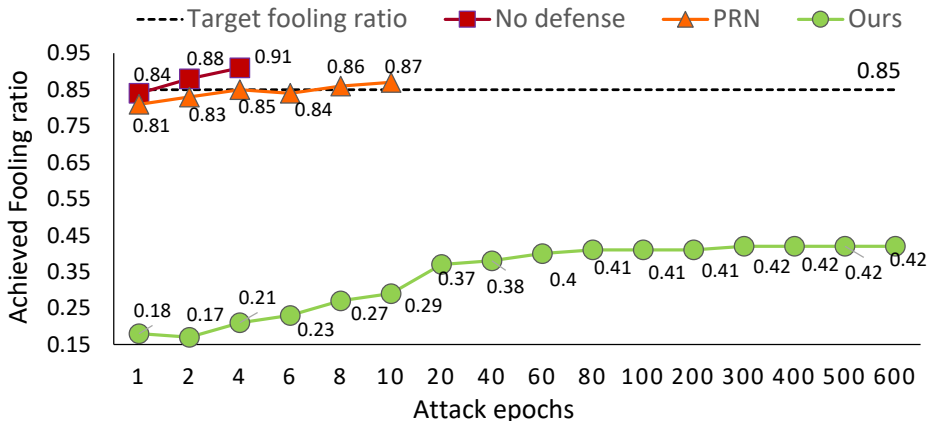


Figure 4.11: Robustness to white-box attacks against defense (secondary attacks): Achieved fooling ratio by attacker vs. attack epochs for an ℓ_∞-norm UAP [1] attack ($\xi = 10$) against CaffeNet [8], where the attacker has full knowledge of the baseline DNN and also the defense. The target fooling ratio for attack is set to 0.85.

Table 4.8: Top-1 accuracy for white-box ℓ_∞-norm sPGD [2] attack against Res152-based ℓ_∞-norm defenses ($\xi = 10$), evaluated on ILSVRC2012. Top-1 accuracy for Res152 on clean images is 0.79.

Baseline	**Ours**	**FD** [37]	**Adv. tr.** [22]	**HGD** [45]	**Shared tr.** [2][2]
0.270	**0.731**	0.641	0.635	0.689	0.727

against PRN [52], even after attacking this proposed defense for 600 attack epochs. Similarly, using the same attack setup outlined in [2], Table 4.8 shows results for white-box sPGD [2] attacks computed by utilizing gradient-information of both the defense and the baseline DNNs, for Res152 [7]. Table 4.8 shows that the proposed defense trained using sPGD attack examples computed against both the baseline DNN and the proposed defense, is robust to subsequent sPGD white-box attacks.

[2]As an implementation of Shared Adversarial Training (Shared tr. [2]) was not publicly available, results are reported as published by the authors in [2] and which were only provided for white-box attacks computed against the defense, whereas results for white-box attacks against the baseline DNN were not provided.

4.4 Conclusion

This work shows that masking adversarial noise in a few select DNN activations significantly improves their adversarial robustness. To this end, a novel selective feature regeneration approach that effectively defends against universal perturbations is proposed, unlike existing adversarial defenses which either pre-process the input image to remove adversarial noise and/or retrain the entire baseline DNN through adversarial training. The ℓ_1-norm of the convolutional filter kernel weights was shown to be an effective way to rank convolutional filters in terms of their susceptibility to adversarial perturbations. Regenerating only the top 50% ranked adversarially susceptible features in a few DNN layers is enough to restore DNN robustness and outperform all existing defenses. Extensive validation of the proposed method by comparing against existing state-of-the-art defenses and shows better generalization across different DNNs, attack norms and even unseen attack perturbation strengths. In contrast to existing approaches, the proposed defense trained solely on one type of universal adversarial attack examples effectively defends against other unseen universal attacks, without additional attack-specific training.

Chapter 5

CONCLUSION

This work proposes methods for improving the robustness of DNNs against commonly occurring image distortions as well as carefully crafted universal (image-agnostic) adversarial perturbations. The first method which is called *DeepCorrect*, identifies the most distortion susceptible DNN activations using a novel objective ranking measure and corrects these activation maps to improve classification in the presence of image distortions. The second method tackles the challenge of making DNNs robust to carefully crafted image-agnostic universal adversarial perturbations by identifying adversarially susceptible DNN filter activations and transforming such activations into adversarially resilient features using trainable *feature regeneration units*. This chapter summarizes the main contributions of this work and suggests directions for future research.

5.1 Contributions

The main contributions of this thesis are summarized as follows:

- The effect of common image distortions like Gaussian blur and Additive White Gaussian Noise (AWGN) on the outputs of pre-trained convolutional filters is evaluated. It is observed that for every layer of convolutional filters in the DNN, certain filters are more susceptible to input distortions than others and that correcting the activations of these filters can help recover lost performance.

- A novel metric which uses the change in DNN prediction accuracy before and after correction of distortion-affected feature activations is proposed in order to identify and rank DNN filters in the order of highest susceptibility to input distortion.

- The worst distortion-affected filter activations in a DNN layer are corrected by appending *correction units*, which are small blocks of stacked convolutional layers trained using a target-oriented loss, at the output of select filters, whilst the rest of the pre-trained filter outputs in a DNN are left unchanged.

- Full-rank convolutions in *DeepCorrect* models are replaced with rank-constrained approximations consisting of a sequence of separable convolutions with rectangular kernels to mitigate the additional computational cost introduced by the proposed *correction units*. The resulting pruned *DeepCorrect* models have almost the same computational cost as the respective pre-trained DNN, during inference.

- Showing the existence of a set of vulnerable convolutional filters, that are largely responsible for erroneous predictions made by a DNN in an adversarial setting. Furthermore, the ℓ_1-norm of the convolutional filter weights is shown to effectively identify such filters, and masking the noise in their activations is able to prevent erroneous predictions.

- Unlike, existing image-domain defenses, the proposed DNN feature domain-based defense uses trainable *feature regeneration units*, which regenerate activations of the aforementioned vulnerable convolutional filters into resilient features (adversarial noise masking) whilst leaving all remaining activations in the DNN unchanged.

- A fast method is proposed to generate strong synthetic adversarial perturbations for training.

- Extensive evaluation on a variety of DNN architectures shows that the proposed defense outperforms all other existing defenses [52, 40, 37, 45, 22, 2]

- Without any additional attack-specific training, the proposed defense trained only on one type of universal attack [1] is shown to effectively defend against other differ-

ent unseen universal attacks [53, 54, 55, 2, 56, 58] and this is the first work in the literature to show such broad generalization across different universal attacks.

5.2 Future Research Directions

The proposed methods in this work provide interesting insights into the susceptibility of DNNs and open up other future research directions. Some of the possible directions for further research using the proposed methods are:

- The proposed convolutional filter ranking measure can be further extending to improve the generic technique of fine-tuning where only select filters are fine-tuned, reducing computational complexity during training and also improving generalization.

- The proposed method of feature correction can be further extended to other vision tasks such as object detection, image segmentation and image transformation.

- A Neural Architecture Search (NAS) can be explored for identifying the optimal design and layer depth for the *feature regeneration units*.

- Adversarially robust DNN architectures and training methods that produce convolutional filter kernels that have a small ℓ_1-norm (i.e. DNNs with sparse convolutional filter kernels) can be explored.

REFERENCES

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

[2] C. K. Mummadi, T. Brox, and J. H. Metzen, "Defending against universal perturbations with shared adversarial training," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[4] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *Computer Vision and Image Understanding*, vol. 106, no. 1. Elsevier, 2007, pp. 59–70. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[5] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech256/

[6] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3485–3492.

[7] 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*, 2016, pp. 770–778.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] 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*, 2015, pp. 1–9.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[12] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[14] 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*, 2015, pp. 91–99.

[15] 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*, 2016, pp. 779–788.

[16] 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*. Springer, 2014, pp. 740–755.

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

[18] S. F. Dodge and L. J. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *Proceedings of the IEEE International Conference on Computer Communications and Networks*, 2017, pp. 1–7.

[19] ——, "Human and DNN Classification Performance on Images with Quality Distortions: A Comparative Study," *ACM Transactions on Applied Perception*, pp. 1–19, 2019.

[20] 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.

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

[22] A. S. L. T. D. V. A. Madry, A. Makelov, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[23] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.

[24] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," *CoRR*, vol. abs/1702.06832, 2017.

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016, pp. 2574–2582.

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

[27] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016.

[28] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy*, 2016, pp. 372–387.

[29] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018, pp. 284–293.

[30] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning models," *CoRR*, vol. abs/1707.08945, 2017.

[31] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial samples and black-box attacks," *CoRR*, vol. abs/1611.02770, 2016.

[32] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1607.02533, 2016.

[33] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," *CoRR*, vol. abs/1804.08598, 2018.

[34] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," *CoRR*, vol. abs/1612.06299, 2016.

[35] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *CoRR*, vol. abs/1710.08864, 2017.

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

[37] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[38] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *International Conference on Learning Representations*, 2018.

[39] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," *CoRR*, vol. abs/1711.00117, 2017.

[40] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 8571–8580.

[41] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," *CoRR*, vol. abs/1608.00853, 2016.

[42] S. Moosavi-Dezfooli, A. Shrivastava, and O. Tuzel, "Divide, denoise, and defend against adversarial attacks," *CoRR*, vol. abs/1802.06806, 2018.

[43] S. Song, Y. Chen, N. Cheung, and C. J. Kuo, "Defense against adversarial attacks with Saak transform," *CoRR*, vol. abs/1808.01785, 2018.

[44] Z. Liu, Q. Liu, T. Liu, Y. Wang, and W. Wen, "Feature Distillation: DNN-oriented JPEG compression against adversarial examples," *International Joint Conference on Artificial Intelligence*, 2018.

[45] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," *CoRR*, vol. abs/1712.02976, 2017.

[46] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *IEEE International Conference on Computer Vision*, Oct 2017, pp. 5775 – 5783.

[47] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *IEEE International Conference on Computer Vision*, Oct 2017, pp. 446–454.

[48] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed System Security Symposium*, 2018.

[49] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 135–147.

[50] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *International Conference on Learning Representations*, 2017.

[51] J. Uesato, B. O'Donoghue, A. van den Oord, and P. Kohli, "Adversarial risk and the dangers of evaluating against weak attacks," *CoRR*, vol. abs/1802.05666, 2018.

[52] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 3389–3398.

[53] K. R. Mopuri, U. Ojha, U. Garg, and R. V. Babu, "NAG: Network for adversary generation," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018.

[54] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017, pp. 1–12.

[55] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[56] V. Khrulkov and I. Oseledets, "Art of singular vectors and universal adversarial perturbations," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 8562 – 8570.

[57] K. Reddy Mopuri, P. Krishna Uppala, and R. Venkatesh Babu, "Ask, acquire, and attack: Data-free uap generation using class impressions," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[58] K. R. Mopuri, A. Ganeshan, and R. V. Babu, "Generalizable data-free objective for crafting universal adversarial perturbations," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, 2019, pp. 2452–2465.

[59] Y. Li, S. Bai, C. Xie, Z. Liao, X. Shen, and A. L. Yuille, "Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses," *CoRR*, vol. abs/1904.00979, 2019.

[60] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Analysis of universal adversarial perturbations," *CoRR*, vol. abs/1705.09554, 2017.

[61] A. Fawzi, S. Moosavi-Dezfooli, P. Frossard, and S. Soatto, "Classification regions of deep neural networks," *CoRR*, vol. abs/1705.09552, 2017.

[62] A. Fawzi, S. Moosavi-Dezfooli, and P. Frossard, "The robustness of deep networks: A geometrical perspective," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 50–62, 2017.

[63] J. Li, F. R. Schmidt, and J. Z. Kolter, "Adversarial camera stickers: A physical camera-based attack on deep learning systems," *CoRR*, vol. abs/1904.00759, 2019.

[64] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *CoRR*, vol. abs/1712.09665, 2017.

[65] J. Hendrik Metzen, M. Chaithanya Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *IEEE International Conference on Computer Vision*, Oct 2017, pp. 2774–2783.

[66] 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*, June 2015, pp. 427–436.

[67] L. J. Karam and T. Zhu, "Quality labeled faces in the wild (QLFW): a database for studying face recognition in real-world environments," in *Proc. SPIE 9394, Human Vision and Electronic Imaging*, vol. XX, 2015.

[68] S. Basu, M. Karki, S. Ganguly, R. DiBiano, S. Mukhopadhyay, S. Gayaka, R. Kannan, and R. Nemani, "Learning sparse feature representations using probabilistic quadtrees and deep belief nets," in *Neural Processing Letters*. Springer, 2015, pp. 1–13.

[69] S. Karahan, M. K. Yildirum, K. Kirtac, F. S. Rende, G. Butun, and H. K. Ekenel, "How image degradations affect deep CNN-based face recognition?" in *International Conference of the Biometrics Special Interest Group*. IEEE, 2016, pp. 1–5.

[70] E. Rodner, M. Simon, R. B. Fisher, and J. Denzler, "Fine-grained recognition in the noisy wild: Sensitivity analysis of convolutional neural networks approaches," *arXiv preprint arXiv:1610.06756*, 2016.

[71] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," *Tech. Rep.*, 2011. [Online]. Available: http://www.vision.caltech.edu/visipedia/CUB-200.html

[72] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008, pp. 722–729.

[73] S. Zheng, Y. Song, T. Leung, and I. J. Goodfellow, "Improving the robustness of deep neural networks via stability training," *CoRR*, vol. abs/1604.04326, 2016. [Online]. Available: http://arxiv.org/abs/1604.04326

[74] Z. Sun, M. Ozay, Y. Zhang, X. Liu, and T. Okatani, "Feature quantization for defending against distortion of images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[75] I. Vasiljevic, A. Chakrabarti, and G. Shakhnarovich, "Examining the impact of blur on recognition by convolutional networks," *CoRR*, vol. abs/1611.05760, 2016. [Online]. Available: http://arxiv.org/abs/1611.05760

[76] Y. Zhou, S. Song, and N.-M. Cheung, "On classification of distorted images with deep convolutional neural networks," *arXiv preprint arXiv:1701.01924*, 2017.

[77] S. Diamond, V. Sitzmann, S. P. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Optimizing image classification architectures for raw sensor data," *CoRR*, vol. abs/1701.06487, 2017. [Online]. Available: http://arxiv.org/abs/1701.06487

[78] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *CoRR*, vol. abs/1803.06373, 2018.

[79] L. Engstrom, A. Ilyas, and A. Athalye, "Evaluating and understanding the robustness of adversarial logit pairing," *CoRR*, vol. abs/1807.10272, 2018.

[80] A. Lamb, J. Binas, A. Goyal, D. Serdyuk, S. Subramanian, I. Mitliagkas, and Y. Bengio, "Fortified networks: Improving the robustness of deep networks by modeling the manifold of hidden representations," *CoRR*, vol. abs/1804.02485, 2018.

[81] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "SHIELD: Fast, practical defense and vaccination for deep learning using JPEG compression," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 196–204.

[82] J. Hayes and G. Danezis, "Learning universal adversarial perturbations with generative models," *CoRR*, vol. abs/1708.05207, 2017.

[83] J. Perolat, M. Malinowski, B. Piot, and O. Pietquin, "Playing the game of universal adversarial perturbations," *CoRR*, vol. abs/1809.07802, 2018.

[84] A. Shafahi, M. Najibi, Z. Xu, J. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," *CoRR*, vol. abs/1811.11304, 2018.

[85] Y. Ruan and J. Dai, "TwinNet: A double sub-network framework for detecting universal adversarial perturbations." in *Future Internet*, vol. 10, 2018, pp. 1–13.

[86] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *BMVC*, 2014.

[87] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti, "TID2008-A database for evaluation of full-reference visual quality assessment metrics," in *Advances of Modern Radioelectronics*, vol. 10, no. 4, 2009, pp. 30–45.

[88] A. Chakrabarti, "A neural approach to blind motion deblurring," *CoRR*, vol. abs/1603.04771, 2016. [Online]. Available: http://arxiv.org/abs/1603.04771

[89] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation," Y. Chauvin and D. E. Rumelhart, Eds. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1995, ch. Backpropagation: The Basic Theory, pp. 1–34.

[90] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *CoRR*, vol. abs/1405.3866, 2014.

[91] C. Tai, T. Xiao, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," *CoRR*, vol. abs/1511.06067, 2015. [Online]. Available: http://arxiv.org/abs/1511.06067

[92] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," in *Proc. Workshop on Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[93] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," in *IEEE Trans. on Image Processing*, vol. 22, no. 4, 2013, pp. 1620–1630.

[94] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *International Conference on Machine Learning*, 2014, pp. 647–655.

[95] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," in *International Journal of Computer Vision*, vol. 88, no. 2, Jun. 2010, pp. 303–338.

[96] T. S. Borkar and L. J. Karam, "DeepCorrect: Correcting DNN models against image distortions," *CoRR*, vol. abs/1705.02406, 2017. [Online]. Available: http://arxiv.org/abs/1705.02406

APPENDIX A

MAXIMUM ADVERSARIAL PERTURBATION

Section 4.2.1 shows that the maximum possible adversarial perturbation in a convolutional filter activation map is proportional to the ℓ_1-norm of its corresponding filter kernel weights. Here, a proof is provided for Equation 4.1 in Section 4.2.1. For simplicity but without loss of generality, let A be a single-channel $n \times n$ input to a $k \times k$ convolutional filter with kernel W. For illustration, consider a 3×3 input A and a 2×2 kernel W as shown below:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \text{ and } W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

Assuming the origin for the kernel W is at the top-left corner and no padding for A (same proof applies also if padding is applied), then the vectorized convolutional output

$$\begin{aligned} \mathbf{e} &= vec(A * W) \\ &= \begin{bmatrix} w_{11} \cdot a_1 + w_{12} \cdot a_2 + w_{21} \cdot a_4 + w_{22} \cdot a_5 \\ w_{11} \cdot a_2 + w_{12} \cdot a_3 + w_{21} \cdot a_5 + w_{22} \cdot a_6 \\ w_{11} \cdot a_4 + w_{12} \cdot a_5 + w_{21} \cdot a_7 + w_{22} \cdot a_8 \\ w_{11} \cdot a_5 + w_{12} \cdot a_6 + w_{21} \cdot a_8 + w_{22} \cdot a_9 \end{bmatrix} \end{aligned}$$

can be expressed as a matrix-vector product as follows:

$$\mathbf{e} = vec(A * W) = M\mathbf{r} \tag{A.1}$$

$$M = \begin{bmatrix} w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \tag{A.2}$$

$$\mathbf{r}^T = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9] \tag{A.3}$$

where $vec(\cdot)$ unrolls all elements of the input matrix with N_1 rows and N_2 columns into an output column vector of size N_1N_2, M is a circulant convolution matrix formed using the elements of W and $\mathbf{r} = vec(A)$.

Similarly, for $A \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{k\times k}$ such that w_{ij} is an element in row i and column j of W, one has $M \in \mathbb{R}^{(n-k+1)^2\times n^2}$, and $e \in \mathbb{R}^{(n-k+1)^2}$ is given by:

$$e = M\mathbf{r} \tag{A.4}$$

$$\|e\|_\infty = \|M\mathbf{r}\|_\infty = \max_{1\le i\le (n-k+1)^2} |\sum_{j=1}^{n^2} m_{ij} r_j| \tag{A.5}$$

$$\begin{aligned} &\le \max_{1\le i\le (n-k+1)^2} \sum_{j=1}^{n^2} |m_{ij}||r_j| \\ &\le \left(\max_{1\le i\le (n-k+1)^2} \sum_{j=1}^{n^2} |m_{ij}| \right) \max_{1\le j\le n^2} |r_j| \end{aligned}$$

$$\leq \left(\max_{1 \leq i \leq (n-k+1)^2} \sum_{j=1}^{n^2} |m_{ij}| \right) \|\mathbf{r}\|_\infty \tag{A.6}$$

where $\mathbf{r} = vec(A), \mathbf{r} \in \mathbb{R}^{n^2}$ such that r_j is the j^{th} element in the vector $\mathbf{r}$ and m_{ij} is the element in row i and column j of the matrix M.

From Equation A.2, $\sum_{j=1}^{n^2} |m_{ij}|$ is always equal to the ℓ_1-norm of the filter kernel weights $\|W\|_1 = \sum_{i'=1}^{k} \sum_{j'=1}^{k} |w_{i'j'}|$ for any row i, $1 \leq i \leq (n-k+1)^2$. Equation A.6, can now be rewritten as:

$$\|\mathbf{e}\|_\infty \leq \|W\|_1 \|\mathbf{r}\|_\infty \tag{A.7}$$

Since $\|\cdot\|_\infty \leq \|\cdot\|_1$ and $\|\cdot\|_\infty \leq \|\cdot\|_2$, one can derive the following inequality:

$$\|\mathbf{e}\|_\infty \leq \|W\|_1 \|\mathbf{r}\|_p \tag{A.8}$$

where $p = 1, 2, \infty$.

APPENDIX B

LIST OF PUBLICATIONS AND PATENT

- **T. Borkar**, F. Heide and L. Karam, "Defending Against Universal Attacks Through Selective Feature Regeneration," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 709-719.

- **T. S. Borkar** and L. J. Karam, "DeepCorrect: Correcting DNNs Against Image Distortions," *IEEE Transactions on Image Processing*, Volume 28, Issue 12, 2019, pp. 6022-6034.

- L. J. Karam and **T. Borkar**, "Systems and Methods for Feature Transformation, Correction and Regeneration for Robust Sensing, Transmission, Computer Vision, Recognition and Classification," US Patent Application No. 16/370,261

- L. Karam, **T. Borkar**. Y. Cao and J. Chae, "Generative Sensing: Transforming Unreliable Sensor Data for Reliable Recognition," *IEEE International Conference on Multimedia Information Processing and Retrieval*, April 2018, pp. 100-105.

- **T. S. Borkar** and L. J. Karam, "Automated Bird Plumage Coloration Quantification in Digital Images," *Advances in Visual Computing*, pp. 220-229. Springer 2014.