THE UNIVERSITY OF CHICAGO

INPUT AND MODEL COMPRESSION FOR ADAPTIVE AND ROBUST CONVOLUTIONAL NEURAL NETWORKS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCE
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
ADAM DZIEDZIC

CHICAGO, ILLINOIS
DECEMBER 2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Most existing model compression techniques optimize for the inference. However, deep convolutional neural networks take GPU-days of computation to train on large data sets and need substantial memory workspace for input data and intermediate results. We propose a method for acceleration and compression of the convolution operation in the Fourier domain, which we call band-limiting. The band-limited training can control resource usage (GPU and memory) while retaining high accuracy. Our method with 50% compression in the frequency domain results in only 1.5% drop in accuracy for ReNet-18 model trained on CIFAR-10 data while reducing the GPU memory usage by 40% and the computation time by 30% in comparison to their full-spectra counterparts. The added-value of the band-limited models is that they provide adversarial robustness to black-box and non-adaptive white-box attacks. We investigate when the compression-based and random perturbation defenses work and why they lead to similar gains in robustness. We observe that all the perturbation methods have a very similar underlying mechanism, where the key is to ensure that the error introduced by the perturbations is big enough to dominate the adversarial noise but small enough to generate valid predictions. We plan on further investigating compression methods and incorporating internal perturbation layers during model training.

CHAPTER 1

INTRODUCTION

Convolutions are fundamental signal processing operations that amplify certain frequencies of the input and attenuate others. Moreover, the convolutional layers are an integral part of neural network architectures for computer vision, natural language processing, and time-series analysis [36, 33, 7]. Recent results suggest that neural networks exhibit a *spectral bias* [49, 65]. They ultimately learn filters with a strong bias towards lower frequencies. Most input data, such as time-series and images, are also naturally biased towards lower frequencies [2, 21, 59]. This begs the question—does a convolutional neural network (CNN) need to explicitly represent the high-frequency components of its convolutional layers? We show that the answer to the question leads to some surprising new perspectives on resource management in terms of the training time and model compression as well as on robustness of models to noisy inputs.

Consider a frequency domain implementation of the convolution function executed in the following steps: (1) transform the filter and the input to the frequency domain; (2) element-wise multiply both frequency spectra; and (3) transform the outcome product to the original domain. Let us assume that the final model is biased towards lower Fourier frequencies [49, 65]. Then, it follows that discarding a significant number of the Fourier coefficients from high frequencies after step (1) should have a minimal effect. A smaller intermediate array size after step (1) reduces the number of multiplications in step (2) as well as the memory usage. This gives us a knob to tune the resource utilization, namely, memory and computation, as a function of how much of the high frequency spectrum we choose to represent. Our primary research question is whether we can train CNNs using such *band-limited* convolutional layers, which only exploit a subset of the frequency spectra of the filter and input data.

While there are several competing compression techniques, such as reduced precision arithmetic [62, 1, 31, 45], weight pruning [26, 66], or sparsification [39], these techniques can be hard to operationalize. CNN optimization algorithms can be sensitive to the noise introduced during the training process, and training-time compression can require specialized libraries to avoid insta-

bility [62, 1]. Furthermore, pruning and sparsification techniques only reduce resource utilization during inference. In our experiments, surprisingly, band-limited training does not seem to suffer the same problems and gracefully degrades predictive performance as a function of compression rate. Band-limited CNNs can be trained with any gradient-based algorithm, where layer's gradient is projected onto the set of allowed frequencies.

We implement an FFT-based convolutional layer that selectively constrains the Fourier spectrum utilized during both forward and backward passes. In addition, we apply standard techniques to improve the efficiency of FFT-based convolution [44], as well as new insights about exploiting the conjugate symmetry of 2D FFTs, as suggested in [52]. With this FFT-based implementation, we find competitive reductions in memory usage and floating point operations to reduced precision arithmetic (RPA) but with the added advantage of training stability and a continuum of compression rates.

In addition to the improvement in performance, the Band-limited training provides a new perspective on adversarial robustness [47]. Adversarial attacks on neural networks tend to involve high-frequency perturbations of input data [30, 43, 47]. Our experiments show that band-limited training produces models that can better reject noise than their full spectra counterparts. This is a direct consequence of our compression methods, where we discard the high frequency coefficients that usually correspond to noise. This observation leads us to exploration of compression techniques for adversarial robustness.

The robustness of neural networks to adversarial examples is one of the crucial requirements to safe deployments of neural networks in natural environments. Surprisingly, the adversarial inputs themselves are not robust and FFT-based compression of the attacking input often recovers the desired prediction. The recovery techniques expand beyond the regime of compression techniques and are also successful when using random perturbations, such as, Gaussian or Uniform noise.

The attacks on Convolutional Neural Networks, such as Carlini & Wanger [10] or PGD [43], generate strategically placed modifications that can be easily dominated by different types of perturbations resulting in correct predictions [19, 54, 29]. This suggests that the standard adversarial

examples are not robust. Many defense techniques explicitly leverage this property and can be retrospectively interpreted as perturbations of the input images. However, a detailed understanding of this phenomenon is lacking from the research literature including: (1) what types of perturbations work and what is their underlying mechanism, (2) whether all attacks exhibit this property, and (3) possible counter-measures attackers can employ to defeat perturbation defenses.

We can interpret a large number of recent defenses as a type of input perturbations, for example, feature squeezing [64], frequency or JPEG compression [19], randomized smoothing [14], and perturbation of network structure or the inputs randomly [32, 67, 25]. The defense techniques exhibit very similar gains in robustness. To show it, we start with a simple model where every example is passed through a lossy channel (stochastic or deterministic) prior to model inference. This channel induces a small perturbation to the input. We optimize the perturbation to be small enough as not to affect the prediction accuracy on clean examples but large enough to dominate any adversarial attack. We find that this trade-off is surprisingly consistent across very different families of input perturbations, where the relationship between channel distortion (the L_2 distance between channel input and output) and robustness is very similar.

Why are some state-of-the-art attacks are sensitive to perturbation-based defenses? We find that many attacks execute an optimization procedure that finds an adversarial image that is very close to the original image in terms of of L_1, L_2, or L_∞ norm. The resultant optimum, i.e., the adversarial image, tends to exhibit a higher level of instability than natural examples, which we demonstrate from the perspective of a first-order and second-order analysis. By instability we mean that small perturbations of the example can affect the prediction confidences.

The unification of perturbation-based defense also gives us insight into how an attacker might avoid them.

Our experiments suggest that all the perturbation based defenses are vulnerable to the same types of attack strategies. We argue that the optimization procedure in the attacker should find the smallest distance from the original image that closes the recovery window. In fact, we can devise a generic attacker that attacks a particularly strong lossy channel, based on the additive Laplace

noise, and adaptive attacks designed on this channel are often successful against other defenses. This result implies that for many input perturbation defenses the attacker need not be fully adaptive, i.e., they do not need to know exactly what kind of transformation is used to defend the network.

CHAPTER 2

BAND-LIMITED TRAINING AND INFERENCE

We propose a novel methodology for band-limited training and inference of CNNs that constrains the Fourier spectrum utilized during both forward and backward passes. Our approach requires no modification of the existing training algorithms or neural network architecture, unlike other compression schemes. An efficient FFT-based implementation of the band-limited convolutional layer for 1D and 2D data that exploits conjugate symmetry, fast complex multiplication, and frequency map reuse. An extensive experimental evaluation across 1D and 2D CNN training tasks illustrates: (1) band-limited training can effectively control the resource usage (GPU and memory) and (2) models trained with band-limited layers retain high prediction accuracy.

2.1 Related work

Model Compression: The idea of model compression to reduce the memory footprint or feed-forward (inference) latency has been extensively studied (also related to distillation) [27, 28, 57, 11]. The ancillary benefits of compression and distillation, such as adversarial robustness, have also been noted in prior work [30, 43, 47]. One of the first approaches was called weight pruning [26], but recently, the community is moving towards convolution-approximation methods [40, 12]. We see an opportunity for a detailed study of the training dynamics with both filter and signal compression in convolutional networks. We carefully control this approximation by tracking the spectral energy level preserved.

Reduced Precision Training: We see band-limited neural network training as a form of reduced-precision training [31, 56, 3, 17]. Our focus is to understand how a spectral-domain approximation affects model training, and hypothesize that such compression is more stable and gracefully degrades compared to harsher alternatives.

Spectral Properties of CNNs: There is substantial recent interest in studying the spectral properties of CNNs [52, 49, 65], with applications to better initialization techniques, theoretical under-

standing of CNN capacity, and eventually, better training methodologies. More practically, FFT-based convolution implementations have been long supported in popular deep learning frameworks (especially in cases where filters are large in size). Recent work further suggests that FFT-based convolutions might be useful on smaller filters as well on CPU architectures [68].

Data transformations: Input data and filters can be represented in Winograd, FFT, DCT, Wavelet or other domains. In our work we investigate the most popular FFT-based frequency representation that is natively supported in many deep learning frameworks (e.g., PyTorch) and highly optimized [61]. Winograd domain was first explored in [37] for faster convolution but this domain does not expose the notion of frequencies. An alternative DCT representation is commonly used for image compression. It can be extracted from JPEG images and provided as an input to a model. However, for the method proposed in [24], the JPEG quality used during encoding is 100%. The convolution via DCT [51] is also more expensive than via FFT.

Small vs Large Filters: FFT-based convolution is a standard algorithm included in popular libraries, such as cuDNN[1]. While alternative convolutional algorithms [37] are more efficient for small filter sizes (e.g., 3x3), the larger filters are also significant. (1) During the backward pass, the gradient acts as a large convolutional filter. (2) The trade-offs are chipset-dependent and [68] suggest using FFTs on CPUs. (3) For ImageNet, both ResNet and DenseNet use 7x7 filters in their 1st layers (improvement via FFT noted by [61]), which can be combined with spectral pooling [53]. (4) The theoretical properties of the Fourier domain are well-understood, and this study elicits frequency domain properties of CNNs.

2.2 Band-Limited Convolution

Let x be an input tensor (e.g., a signal) and y be another tensor representing the filter. We denote the convolution operation as $x * y$. Both x and y can be thought of as discrete functions (mapping tensor index positions $\mathbf{n}$ to values $x[\mathbf{n}]$). Accordingly, they have a corresponding Fourier representation,

1. https://developer.nvidia.com/cudnn

which re-indexes each tensor in the spectral (or frequency) domain:

$$F_x[\omega] = F(x[\mathbf{n}]) \qquad F_y[\omega] = F(y[\mathbf{n}])$$

This mapping is invertible $x = F^{-1}(F(x))$. Convolutions in the spectral domain correspond to element-wise multiplications:

$$x * y = F^{-1}(F_x[\omega] \cdot F_y[\omega])$$

The intermediate quantity $S[\omega] = F_x[\omega] \cdot F_y[\omega]$ is called the *spectrum* of the convolution. We start with the modeling assumption that for a substantial portion of the high-frequency domain, $|S[\omega]|$ is close to 0. This assumption is substantiated by the recent work by Rahman et al. studying the inductive biases of CNNs [49], with experimental results suggesting that CNNs are biased towards learning low-frequency filters (i.e., smooth functions). We take this a step further and consider the joint spectra of both the filter and the signal to understand the memory and computation implications of this insight.

2.2.1 Compression

Let $M_c[\omega]$ be a discrete indicator function defined as follows:

$$M_c[\omega] = \begin{cases} 1, \omega \leq c \\ 0, \omega > c \end{cases}$$

$M_c[\omega]$ is a mask that limits the $S[\omega]$ to a certain *band* of frequencies. The *band-limited* spectrum is defined as, $S[\omega] \cdot M_c[\omega]$, and the band-limited convolution operation is defined as:

$$x *_c y = F^{-1}\{(F_x[\omega] \cdot M_c[\omega]) \cdot (F_y[\omega] \cdot M_c[\omega])\} \tag{1}$$

$$= F^{-1}(S[\omega] \cdot M_c[\omega]) \tag{2}$$

The operation $*_c$ is compatible with automatic differentiation as implemented in popular deep learning frameworks such as PyTorch and TensorFlow. The mask $M_c[\omega]$ is applied to both the signal $F_x[\omega]$ and filter $F_y[\omega]$ (in equation 1) to indicate the compression of both arguments and fewer number of element-wise multiplications in the frequency domain.

2.2.2 FFT Implementation

We implement band-limited convolution with the Fast Fourier Transform. FFT-based convolution is supported by many Deep Learning libraries (e.g., cuDNN). It is most effective for larger filter-sizes where it significantly reduces the amount of floating point operations. While convolutions can be implemented by many algorithms, including matrix multiplication and the Winograd minimal filtering algorithm, the use of an FFT is actually important (as explained above in section 2.1). The compression mask $M_c[\omega]$ is sparse in the Fourier domain. $F^{-1}(M_c)$ is, however, dense in the spatial or temporal domains. If the algorithm does not operate in the Fourier domain, it cannot take advantage of the sparsity in the frequency domain.

The Expense of FFT-based Convolution

It is worth noting that pre-processing steps are crucial for a correct implementation of convolution via FFT. The filter is usually much smaller (than the input) and has to be padded with zeros to the final length of the input signal. The input signal has to be padded on one end with as many zeros as the size of the filter to prevent the effects of wrapped-around filter data (for example, the last values of convolution should be calculated only from the final overlap of the filter with the input signal and should not be polluted with values from the beginning of the input signal).

Due to this padding and expansion, FFT-based convolution implementations are often expensive in terms of memory usage. Such an approach is typically avoided on GPU architecture, but recent results suggest improvements on CPU architecture [68]. The compression mask $M_c[\omega]$ reduces the size of the expanded spectra; we need not compute the product for those values that are masked out. Therefore, a band-limiting approach has the potential to make FFT-based convolution

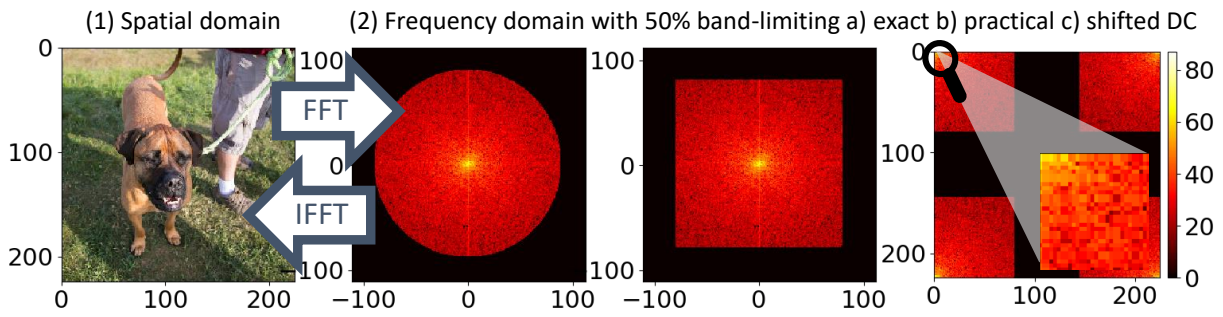

Figure 2.1: *Transformations from input image to compressed FFT map. (1) Natural image in the spatial domain. (2) FFT transformation to frequency domain and a) exact band-limiting to 50%, b) practical band-limiting to 50%, c) lowest frequencies shifted to corners. The heat maps of magnitudes of Fourier coefficients are plotted for a single channel (0-th) in a logarithmic scale (dB) with linear interpolation and the max value is colored with white while the min value is colored with black.*

more practical for smaller filter sizes.

Band-limiting technique

We present the transformations from a natural image to a band-limited FFT map in Figure 2.1.

The FFT domain cannot be arbitrarily manipulated as we must preserve *conjugate symmetry*. For a 1D signal this is straight-forward. $F[-\omega] = F^*[\omega]$, where the sign of the imaginary part is opposite when $\omega < 0$. The compression is applied by discarding the high frequencies in the first half of the signal. We have to do the same to the filter, and then, the element-wise multiplication in the frequency domain is performed between the compressed signal (input map) and the compressed filter. We use zero padding to align the sizes of the signal and filter. We execute the inverse FFT (IFFT) of the output of this multiplication to return to the original spatial or time domain.

In addition to the conjugate symmetry there are certain values that are constrained to be real. For example, the first coefficient is real (the average value) for the odd and even length signals and the middle element ($\left\lfloor \frac{N}{2} \right\rfloor + 1$) is also real for the even-length signals. We do not violate these constraints and keep the coefficients real, for instance, by replacing the middle value with zero during compression or padding the output with zeros.

The conjugate symmetry for a 2D signal $F[-\omega, -\theta] = F^*[\omega, \theta]$ is more complicated. If the

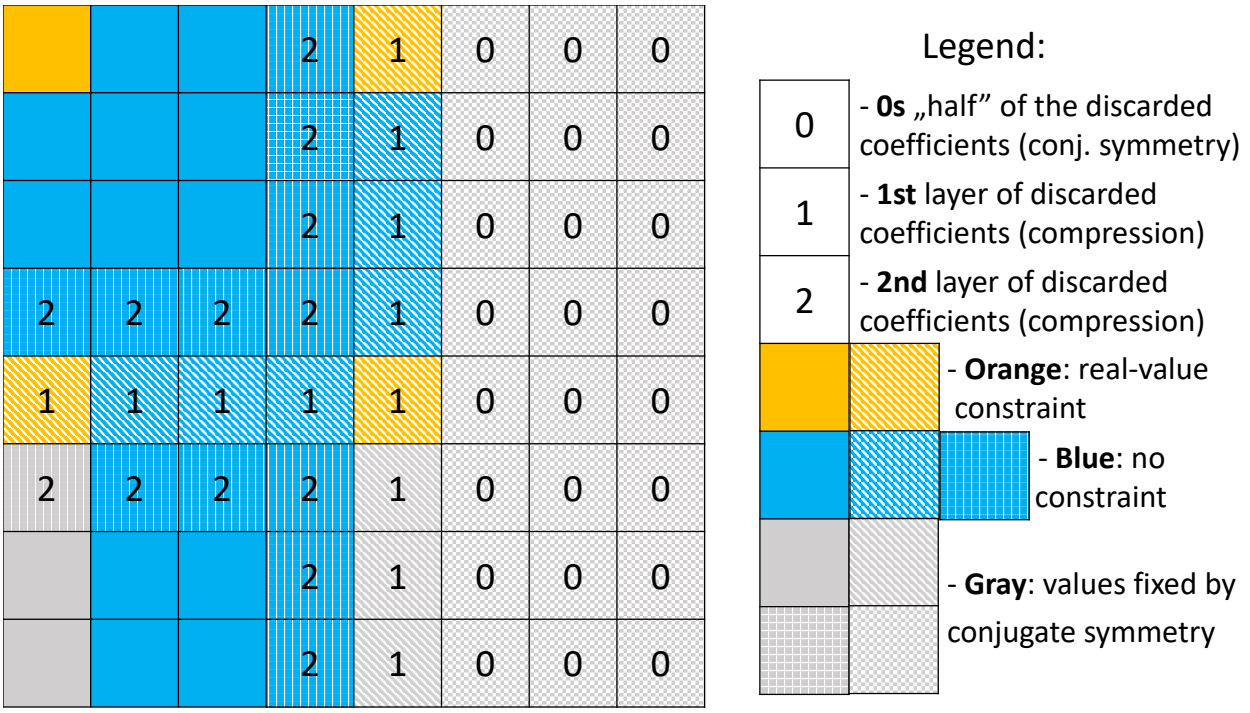

Figure 2.2: *An example of a square input map with marked conjugate symmetry (**Gray** cells). Almost half of the input cells marked with **0**s (zeros) are discarded first due to the conjugate symmetry. The remaining map is compressed layer by layer (we present how the first two layers: **1** and **2** are selected). **Blue** and **Orange** cells represent a minimal number of coefficients that have to be preserved in the frequency domain to fully reconstruct the initial spatial input. Additionally, the **Orange** cells represent real-valued coefficients.*

real input map is of size $M \times N$, then its complex representation in the frequency domain is of size $M \times (\left\lfloor \frac{N}{2} \right\rfloor + 1)$. The real constraints for 2D inputs were explained in detail in Figure 2.2, similarly to [52]. For the most interesting and most common case of even height and width of the input, there are always four real coefficients in the spectral representation (depicted as **Orange** cells: top-left corner, middle value in top row, middle value in most-left column and the value in the center). The DC component is located in the top-left corner. The largest values are placed in the corners and decrease towards the center. This trend is our guideline in the design of the compression pattern, in which for the *left half* of the input, we discard coefficients from the center in *L-like* shapes towards the top-left and bottom-left corners.

Map Reuse

The FFT computations of the tensors: input map, filter, and the gradient of the output as well as the IFFT of the final output tensors are one of the most expensive operations in the FFT-based convolution. We avoid re-computation of the FFT for the input map and the filter by saving their frequency representations at the end of the forward pass and reusing them in the corresponding backward pass. The memory footprint for the input map in the spatial and frequency domains is almost the same. We retain only half of the frequency coefficients but they are represented as complex numbers. Further on, we assume square input maps and filters (the most common case). For an $N \times N$ real input map, the initial *complex-size* is $N \times (\left\lfloor \frac{N}{2} \right\rfloor + 1)$. The filter (also called kernel) is of size $K \times K$. The FFT-ed input map has to be convolved with the gradient of size $G \times G$ in the backward pass and usually $G > K$. Thus, to reuse the FFT-ed input map and avoid wrapped-around values, the required padding is of size: $P = \max(K-1, G-1)$. This gives us the final full spatial size of tensors used in FFT operations $(N+P) \times (N+P)$ and the corresponding full *complex-size* $(N+P) \times (\left\lfloor \frac{(N+P)}{2} \right\rfloor + 1)$ that is finally compressed.

2.2.3 Implementation in PyTorch and CUDA

Our compression in the frequency domain is implemented as a module in PyTorch that can be plugged into any architecture as a convolutional layer. The code is written in Python with extensions in C++ and CUDA for the main bottleneck of the algorithm. The most expensive computationally and memory-wise component is the Hadamard product in the frequency domain. The complexity analysis of the FFT-based convolution is described in [44] (section 2.3, page 3). The complex multiplications for the convolution in the frequency domain require $3Sf'fn^2$ real multiplications and $5Sf'fn^2$ real additions, where S is the mini-batch size, f' is the number of filter banks (i.e., kernels or output channels), f is the number of input channels, and n is the height and width of the inputs. In comparison, the cost of the FFT of the input map is $Sfn^2 2logn$, and usually $f' >> 2logn$. We implemented in CUDA the fast algorithm to multiply complex numbers with 3 real multiplications instead of 4 as described in [37].

Our approach to convolution in the frequency domain aims at saving memory and utilizing as many GPU threads as possible. In our CUDA code, we fuse the element-wise complex multiplication (which in a standalone version is an injective one-to-one map operator) with the summation along an input channel (a reduction operator) in a thread execution path to limit the memory size from $2Sff'n^2$, where 2 represents the real and imaginary parts, to the size of the output $2Sf'n^2$, and avoid any additional synchronization by focusing on computation of a single output cell: (x, y) coordinates in the output map. We also implemented another variant of convolution in the frequency domain by using tensor transpositions and replacing the complex tensor multiplication (CGEMM) with three real tensor multiplications (SGEMM).

Table 2.1: Test accuracies for ResNet-18 on CIFAR-10 and DenseNet-121 on CIFAR-100 with the same compression rate across all layers. We vary compression from 0% (full-spectra model) to 50% (band-limited model).

CIFAR	0%	10%	20%	30%	40%	50%
10	93.69	93.42	93.24	92.89	92.61	92.32
100	75.30	75.28	74.25	73.66	72.26	71.18

2.3 Experimental Results

We run our experiments on single GPU deployments with NVidia P-100 GPUs and 16 GBs of total memory. The objective of our experiments is to demonstrate the robustness and explore the properties of band-limited training and inference for CNNs.

2.3.1 Effects of Band-limited Training on Inference

First, we study how band-limiting training effects the final test accuracy of two popular deep neural networks, namely, ResNet-18 and DenseNet-121, on CIFAR-10 and CIFAR-100 datasets, respectively. Specifically, we vary the compression rate between 0% and 50% for each convolutional layer (i.e., the percentage of coefficients discarded) and we train the two models for 350 epochs. Then, we measure the final test accuracy using the same compression rate as the one used during training. Our results in Table 2.1 show a smooth degradation in accuracy despite the aggressive compression applied during band-limiting training.

To better understand the effects of band-limiting training, in Figure 2.3, we explore two different compression schemes: (1) fixed compression, which discards the same percentage of spectral coefficients in each layer and (2) energy compression, which discards coefficients in an adaptive manner based on the specified energy retention in the frequency spectrum. By Parseval's theorem, the energy of an input tensor x is preserved in the Fourier domain and defined as: $E(x) = \sum_{n=0}^{N-1} |x[n]|^2 = \sum_{\omega=0}^{2\pi} |F_x[\omega]|$ (for normalized FFT transformation). For example, for two convolutional layers of the same size, a fixed compression of 50% discards 50% of coefficients in

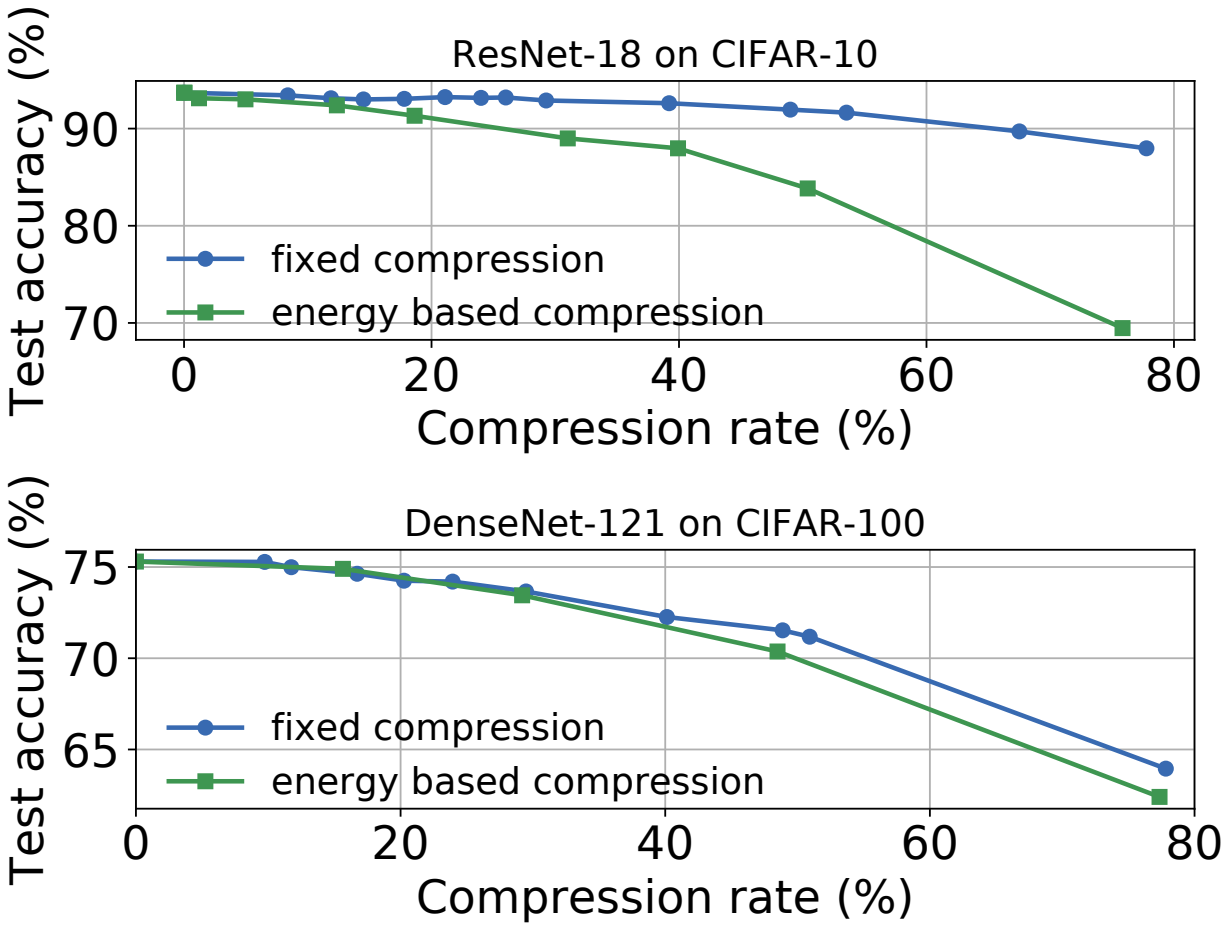

Figure 2.3: *Test accuracy as a function of the compression rate for ResNet-18 on CIFAR-10 and DenseNet-121 on CIFAR-100. The fixed compression scheme that uses the same compression rate for each layer gives the highest test accuracy.*

each layer. On the other hand, the energy approach may find that 90% of the energy is preserved in the 40% of the low frequency coefficients in the first convolutional layer while for the second convolutional layer, 90% of energy is preserved in 60% of the low frequency coefficients.

For more than 50% of compression rate for both techniques, the fixed compression method achieves the max test accuracy of 92.32% (only about 1% worse than the best test accuracy for the full model) whereas the preserved energy method results in significant losses (e.g., ResNet-18 reaches 83.37% on CIFAR-10). Our findings suggest that altering the compression rate during model training may affect the dynamics of SGD. The worse accuracy of the models trained with any form of dynamic compression is result of the higher noise incurred by frequent changes to the

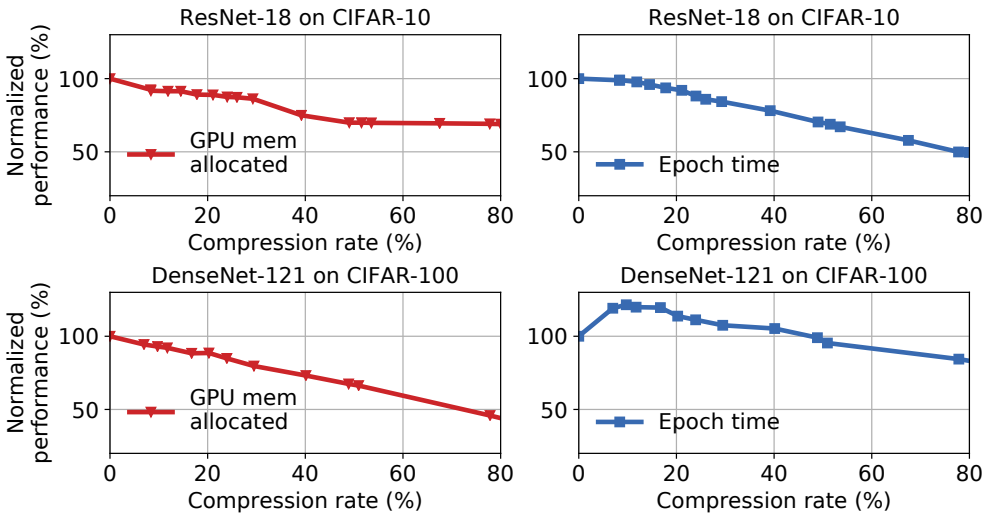

Figure 2.4: *Normalized performance (%) between models trained with different FFT-compression rates.*

number of coefficients that are considered during training. The test accuracy for energy-based compression follows the coefficient one for DenseNet-121 while they markedly diverge for ResNet-18. ResNet combines outputs from L and $L+1$ layers by summation. In the adaptive scheme, this means adding maps produced from different spectral bands. In contrast, DenseNet concatenates the layers.

To dive deeper into the effects on SGD, we performed an experiment where we keep the same energy preserved in each layer and for every epoch. Every epoch we record what is the physical compression (number of discarded coefficients) for each layer. The dynamic compression based on the energy preserved shows that at the beginning of the training the network is focused on the low frequency coefficients and as the training unfolds, more and more coefficients are taken into account. The compression based on preserved energy does not steadily converge to a certain compression rate but can decrease significantly and abruptly even at the end of the training process (especially, for the initial layers).

2.3.2 Control of the GPU and Memory Usage

In Figure 2.4, we compare two metrics: maximum GPU memory allocated (during training) and time per epoch, as we increase the compression rate. The points in the graph with 100% performance for 0% of compression rate correspond to the values of the metrics for the full spectra (uncompressed) model. We normalize the values for the compressed models as: $\frac{\text{metric value for a compressed model}}{\text{metric value for the full spectra model}}$. 100%.

For the ResNet-18 architecture, a small drop in accuracy can save a significant amount of computation time. However, for more convolutional layers in DenseNet-121, the overhead of compression (for small compression rate) is no longer amortized by fewer multiplications (between compressed tensors). The overhead is due to the modifications of tensors to compress them in the frequency domain and their decompression (restoration to the initial size) before going back to the spatial domain (to preserve the same frequencies for the inverse FFT). FFT-ed tensors in PyTorch place the lowest frequency coefficients in the corners. For compression, we extract parts of a tensor from its top-left and bottom-left corners. For the decompression part, we pad the discarded parts with zeros and concatenate the top and bottom slices.

DenseNet-121 shows a significant drop in GPU memory usage with relatively low decrease in accuracy. On the other hand, ResNet-18 is a smaller network and after about 50% of the compression rate, other than convolutional layers start dominate the memory usage. The convolution operation remains the major computation bottleneck and we decrease it with higher compression.

2.3.3 Robustness to Noise

Next, we evaluate the robustness of band-limited CNNs. Specifically, models trained with more compression discard part of the noise by removing the high frequency Fourier coefficients. In Figure 2.5, we show the test accuracy for input images perturbed with different levels of Gaussian noise, which is controlled systematically by the sigma parameter, fed into models trained with different compression levels (i.e., 0%, 50%, and 85%) and methods (i.e., band-limited vs. RPA-based). Our results demonstrate that models trained with higher compression are more robust to

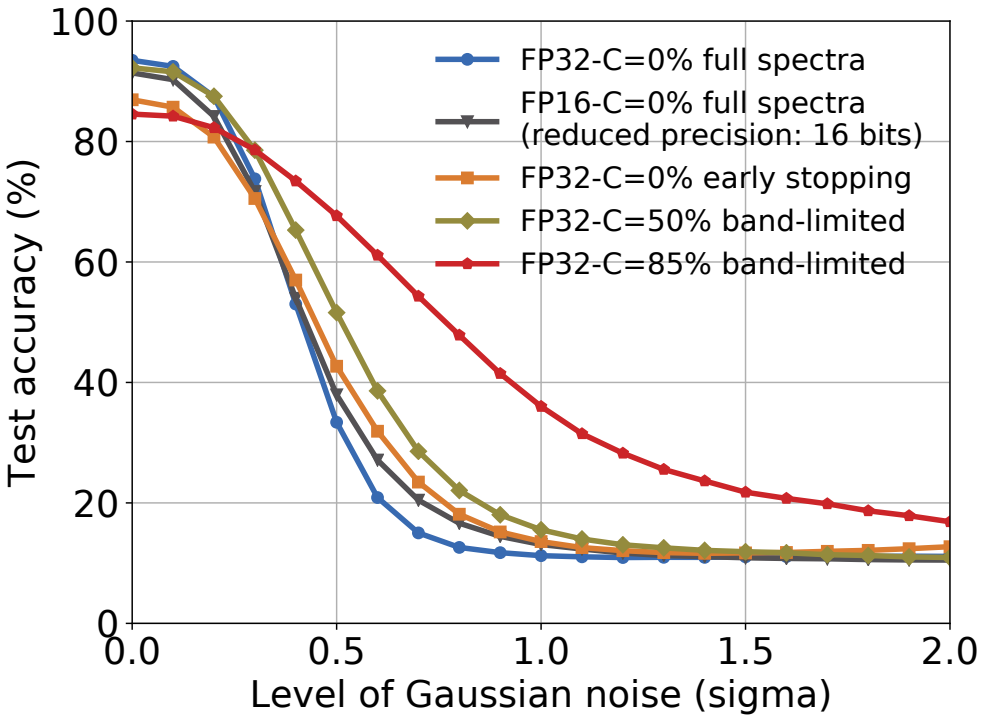

Figure 2.5: *Input test images are perturbed with Gaussian noise, where the sigma parameter is changed from 0 to 2. The more band-limited model, the more robust it is to the introduced noise. We use ResNet-18 models trained on CIFAR-10.*

the inserted noise. Interestingly, band-limited CNNs also outperform the RPA-based method and under-fitted models (e.g., via early stopping), which do not exhibit the robustness to noise.

We additionally run experiments using Foolbox [50]. Our method is robust to decision-based and score-based (black-box) attacks (e.g., the band-limited model is better than the full-spectra model in 70% of cases for the additive uniform noise, and in about 99% cases for the pixel perturbations attacks) but not to the gradient-based (white-box and adaptive) attacks, e.g., Carlini-Wagner [9] (band-limited convolutions return proper gradients). Fourier properties suggest further investigation of invariances under adversarial rotations and translations [20].

CHAPTER 3

PERTURBATION DEFENSES FOR ADVERSARIAL ATTACKS

The non-adaptive attacks are not robust since small changes to the adversarial input often recover the original label. This is an obvious corollary to the very existence of adversarial examples that by definition are relatively close to correctly predicted examples in the input space. Random perturbations of the input can dominate the strategically placed perturbations synthesized by an attack. In fact, the results are consistent across both deterministic and stochastic channels that degrade the fidelity of the input example. From the perspective of the attacker, the recovery window can be closed to make the perturbation based recovery techniques ineffective.

3.1 Related work

Much of the community's current understanding of adversarial sensitivity in neural networks is based on the seminal work by [58]. Multiple contemporaneous works also studied different aspects of this problem, postulating linearity and over-parametrization as possible explanations [23, 6]. Since the beginning of this line of work, the connection between compression and adversarial robustness has been recognized. The main defense strategies include: the idea of defensive network distillation[1] [47], quantizing inputs using feature squeezing [64], the thermometer encoding as another form of quantization [8], JPEG compression harnessed by [19, 25, 15, 16, 5, 42]. Other line of research leveraged connection between randomization and adversarial robustness: Pixel Deflection [48], random resizing and padding of the input image [63], and total variance minimization [25]. In our work we unify the methods based on compression and randomization that are applied to the input images.

While all of the aforementioned defenses were later broken [10, 4], it is important to understand why these approaches afforded any form of robustness. The community actually lacks consensus on this point: [58] suggest that neural networks have *blind spots*, [64] suggest that quantization

1. The distillation is a form of compression, however, the defensive distillation does not result in smaller models.

makes the adversarial search space smaller, [8] suggest the linearity is the main culprit and quantized inputs break up linearity.

[67] inject random Gaussian noise into an image and then discretize it. This method fits into the noisy channel framework with different definitions of $C(x)$. They show improved performance of this combined model in the non-adaptive setting. Our experiments find that simply injecting Gaussian noise (or even Uniform or Laplace noise) is an equally effective defense method. We also show that the discretization is not essential to good performance if the level of noise is appropriately tuned. The imprecise channel defense in a neural network is also related to the idea of gradient masking or gradient obfuscation, i.e., a hard to differentiate layer [46]. In this work, the backward pass computation is perturbed to make it difficult for a gradient-based attack to synthesize an adversarial image (while the forward pass is kept the same). We implement both non-adaptive attacks and adaptive that can observe the channel and take an approximate gradient through it. We further focus our study on the families of white-box attacks proposed by [9], and their adaptive variants.

Noise injection can be much more powerful than regularization or a dataset augmentation method. The dropout algorithm can be seen as applying noise to the hidden units. The dropout randomization [22] was used to create a defense that was not completely broken and required a high distortion added to the adversarial examples [10]. Many new defenses propose randomization through noise injection without considering the adversarial training [67, 14]. The work on injection of noise into inputs and each of the layers of neural networks by [41] is a strong heuristic that led to defenses with theoretical guarantees. The random smoothing provides a certified robustness up to a certain threshold of input distortion (and is not designed to be robust beyond the threshold) by utilizing inequalities from the differential privacy literature [38]. [14] improve the theoretical bounds of methods that randomly smooth the input examples. Recent work focuses on combination of randomized smoothing with adversarial training and achieves state of the art in terms of the provable robustness [55].

3.2 Lossy Channel Model

We consider convolutional neural networks that take $w \times h$ (width times height) RGB digital images as input, giving an example space of $\mathcal{X} \in (255)^{w \times h \times 3}$, where (z) denotes the integer numbers from 0 to z. We consider a discrete label space of k classes represented as a confidence value $\mathcal{Y} \in [0,1]^k$. Neural networks are parametrized functions (by a weight vector θ) between the example and label spaces $f(x;\theta) : \mathcal{X} \mapsto \mathcal{Y}$.

An adversarial input x_{adv} is a perturbation of a correctly predicted example x that is incorrectly predicted by f.

$$f(x) \neq f(x_{adv})$$

The *distortion* is the ℓ_2 error between the original example and the adversarial one:

$$\delta_{adv} = \|x - x_{adv}\|_2$$

3.2.1 Model

Approximating $f(\cdot)$ with a less precise version $\bar{f}(\cdot)$ can counter-intuitively make it more robust [19]:

$$f(x) = \bar{f}(x_{adv})$$

Intuitively, a lossy version of f introduces noise into a prediction which dominates the strategic perturbations found by an adversarial attack procedure. It turns out that we can characterize a number of popular defense methodologies with this basic framework.

Let x be an example and f be a trained neural network. Precise evaluation means running $f(x)$ and observing the predicted label. Imprecise evaluation involves first transforming x through a deterministic or stochastic noise process $C(x) = C[x' \mid x]$, and then evaluating the neural network

$$y = f(x') \quad x' \sim C(x)$$

We can think of $C(x)$ as a noisy channel (as in signal processing). The *distortion* of a $C(x)$ is the expected ℓ_2 reconstruction error:

$$\delta_c = \mathbf{E}[\|C(x) - x\|_2],$$

which is a measure of how much information is lost passing the example through a channel.

This paper shows that there is a subtle trade-off between δ_c and δ_{adv}. In particular, we can find δ_c such that $\delta_c >> \delta_{adv}$ and $f(x) = f(C(x_{adv}))$. We show that compression and randomization based techniques exhibit this property.

Example of Deterministic Channel

When $C(x)$ is deterministic it can be thought of as a lossy compression technique. Essentially, we run the following operation on each input example:

$$x' = \texttt{compress}(x)$$

One form of compression for CNNs is color-depth compression. Most common image classification neural network architectures convert the integer valued inputs into floating point numbers. We abstract this process with the `norm` function that for each pixel $n \in (255)$ maps it to a real number $v \in [0,1]$ by normalizing the value and the corresponding `denorm` function that retrieves the original integer value (where $\lfloor \rceil$ denotes the nearest integer function) [2]:

$$\texttt{norm}(n) := \frac{n}{255} \qquad \texttt{denorm}(v) := \lfloor 255 * v \rceil$$

This process is normally reversible $v = \texttt{norm}(\texttt{denorm}(v))$, but we can artificially make this process lossy. Consider a parametrized $C(\cdot)$ version of the color-depth compression function:

$$C(v,b) := \frac{1}{2^b - 1} \cdot \lfloor (2^b - 1) * v \rceil$$

2. More complex normalization schemes exist but for ease of exposition we focus on this simple process.

By decreasing b by Δb we reduce the fidelity of representing v by a factor of $2^{\Delta b}$ (for the b bits of precision).

Example of Stochastic Perturbation

The channel model is particularly interesting when $C(x)$ is stochastic. Randomization has also been noted to play a big role in strong defenses in prior work [43, 67, 14]. For example, we could add independent random noise to each input pixel:

$$x' = x + \varepsilon$$

We consider two schemes, Gaussian $\varepsilon \sim N(0, \sigma)$ and additive Uniform noise $\varepsilon \sim U(-B, B)$ which add independent noise to each pixel.

One of the advantages of randomization is that an adversary cannot anticipate how the particular channel C will transform an input before prediction. However, there is another subtle advantage to randomization. Randomized approaches can partially recover their loss in accuracy due to imprecision by averaging over multiple instances of the perturbation. In classification problems, we can take the most frequent label seen after T perturbation trials:

$$\bar{f}(x) = \arg\max_{1\ldots k} \sum_{i}^{T} f(x + \varepsilon)$$

3.2.2 Perturbation Analysis

While the intuition is that the channel's perturbations dominate strategically placed distortions in an adversarial example, the underlying mathematical mechanism of why recovery is possible is not clear. We start with the hypothesis that synthesized adversarial examples have *unstable* predictions–meaning that small perturbations to the input space can change confidence values drastically. How do we quantify instability?

Let $f(x)$ be a function that maps an image to a single class confidence value (i.e., a scalar

output). We want to understand how $f(x)$ changes if x is perturbed by ε. We can apply a Taylor expansion of f around the given example x:

$$f(x+\varepsilon) \approx f(x) + \varepsilon^T \nabla_x f(x) + \frac{1}{2}\varepsilon^T \nabla_x^2 f(x)\varepsilon + ...$$

where $\nabla_x f(x)$ denotes the gradient of the function f with respect to x and $\nabla_x^2 f(x)$ denotes the Hessian of the function f with respect to x. The magnitude of the change in confidence is governed by the Taylor series terms in factorially decreasing importance. $\|\varepsilon\|_2$ is exactly the distortion measure δ_c described at the beginning of Section 3.2.1. Thus, the expression is bounded in terms of the *operator* norm, or the maximal change in norm that could be induced, of each of the terms:

$$\varepsilon^T \nabla_x f(x) + \frac{1}{2}\varepsilon^T \nabla_x^2 f(x)\varepsilon + ... \leq \delta_c \, M_1(x) + \frac{1}{2}\delta_c^2 \, M_2(x) + ...$$

As $\nabla_x f(x)$ is a vector, this is simply the familiar ℓ_2 norm, and for the second order term this is the maximal eigenvalue:

$$M_1(x) = \|\nabla_x f(x)\|_2 \quad M_2(x) = \lambda_{max}(\nabla_x^2 f(x))$$

When M_1 and M_2 are larger this means there is a greater propensity to change the prediction for small perturbations. We will show experimentally that for certain types of attacks the M_1 and M_2 values around adversarial examples exhibit signs of instability compared to those around natural examples–suggesting a mathematical mechanism of why recovery is possible.

3.3 Experimental results

Our experiments evaluate the efficacy of imprecision based defenses in a number of different adversarial problem settings. The main objective is to show that perturbation defenses give similar gains in robustness.

We use ResNet50 trained on ImageNet and ResNet18 trained on CIFAR-10 datasets. For each image we generate an adversarial attack with popular white-box gradient-based methods: PGD L_∞ [43] and C&W L_2 [9]. Then, we run the adversarial images through different channels: (1) compression based: Frequency Compression (FC), Color Depth reduction (CD), SVD-based compression (SVD), and (2) randomized channels: Uniform noise (Unif), Gaussian Noise (Gauss), Laplace Noise (Laplace). We measure the accuracy of $f(C(x))$, which indicates the ability of the imprecise channels to recover the original label. We present the results in Figure 3.1.

When there is an identity channel, the adversarial attack is always successful. However, each of the imprecise channels is able to recover a substantial portion of original labels from the adversarial examples. It is important to note that we are evaluating these attacks in the setting, where any misclassification is considered a success and the adversary is not aware of the defense.

Importantly, all the channels can be tuned to recover very similar maximum accuracy after attacks. For example, all the channels can achieve about 85% accuracy for CIFAR-10 after non-adaptive PGD or C&W attacks. This suggests that any form of imprecision with the right error magnitude is effective at defending against these types of attacks. We observe that the attacks that incur higher distortions such as FGSM or LBFGS decrease the accuracy more than the iterative attacks such as C&W L_2 or PGD L_∞. This is because the iterative attacks find the adversarial images that are closer to the original images in terms of the corresponding distance measure that they optimize for in the input space. *The key is to ensure that the error introduced by the imprecise channels is big enough to dominate the adversarial perturbations but small enough to generate valid predictions.* For the different imprecise channels, we plot the channel distortion against the accuracy. The curves are qualitatively similar in the *low* noise regime, but they show more differences when higher distortions are incurred. The lowest accuracy across the datasets for high

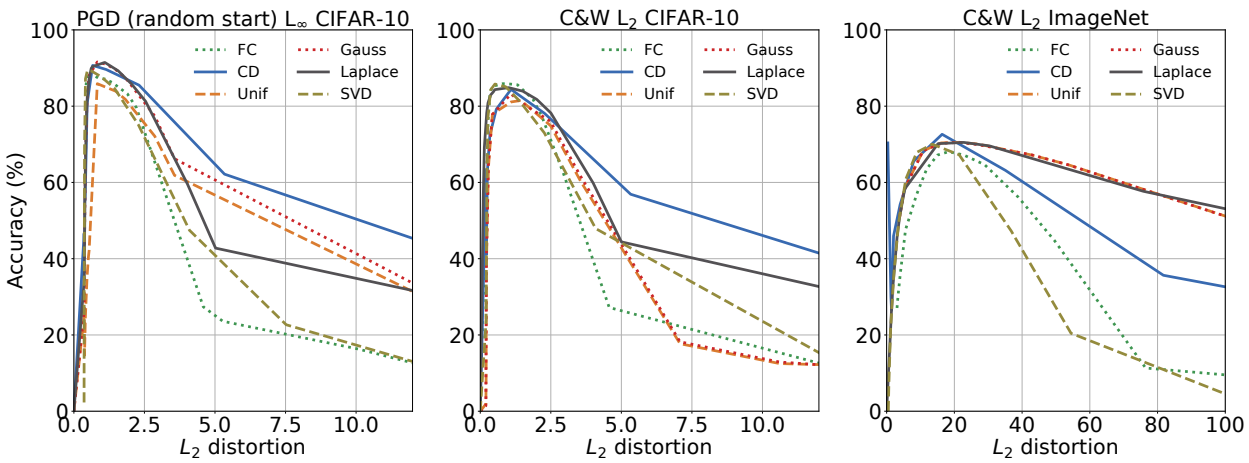

Figure 3.1: We plot the *channel distortion* against accuracy (%). The experiment is run for the PGD attack with 40 iterations and C&W L_2 attack with 100 iterations on all images from the test CIFAR-10 and the dev ImageNet datasets. The adversary is not aware of the defense. The test accuracy on the full clean data is 93.56% and 76.13% for CIFAR-10 and ImageNet, respectively. Interestingly enough, we observe that the C&W attack is on the sub-pixel level for ImageNet and rounding to the nearest 8 bit integers in the CD channel has high recovery rate, while for 7 bit integer the accuracy drops slightly due to imprecision.

distortions is observed for content-preserving FC and SVD compression channels. The Gaussian and Uniform channels have very similar trends. They outperform other channels on ImageNet for large distortions but are less performant on low-resolution CIFAR-10 images, where the CD channel achieves higher accuracy.

CHAPTER 4

RESEARCH PROGRESS AND PLAN

The thesis will include three parts of work as follows:

1. Band-limited training and inference for Convolutional Neural Networks

    - Status: **Completed**
    - Problem statement: The convolutional layers are core building blocks of neural network architectures. In general, a convolutional filter applies to the entire frequency spectrum of the input data. FFT-based convolution, implemented in the NVIDIA cuDNN library [1], is efficient for big filters but requires very large memory workspace. We explore how to artificially constrain the frequency spectra of filters and input maps, called band-limiting, during training and inference. The frequency domain constraints apply to both the feed-forward and back-propagation steps. Experimentally, we observe that Convolutional Neural Networks (CNNs) are resilient to this compression scheme and results suggest that CNNs learn to leverage lower-frequency components. In particular, we found: (1) band-limited training can effectively control the resource usage (GPU and memory); (2) models trained with band-limited layers retain high prediction accuracy; and (3) requires no modification to existing training algorithms or neural network architectures to use unlike other compression schemes.
    - Accomplishments:

        a) Our paper on *Band-limited Training and Inference For Convolutional Nerual Networks* published at ICML 2019 [18].

        b) We also have a paper published on *Artificial Intelligence in Resource-Constrained and Shared Environments* at ACM SIGOPS Operating Systems Review [35].

1. `https://developer.nvidia.com/cudnn`

c) We also used the techniques in our paper on *Machine Learning based detection of multiple Wi-Fi BSSs for LTE-U CSAT* published at International Conference on Computing, Networking and Communications (ICNC 2020) [60].

2. A Perturbation Analysis of Input Transformations for Adversarial Attacks

   - Status: **In-progress**
   - Problem statement: The existence of adversarial examples, or intentional mis-predictions constructed from small changes to correctly predicted examples, is one of the most significant challenges in neural network research today. Many new defenses are based on a simple observation - the adversarial inputs themselves are not robust and small perturbations to the attacking input often recover the desired prediction. While the intuition is clear, a detailed understanding of this phenomenon is missing from the research literature. We carry out a comprehensive experimental analysis of when and why perturbation defenses work and potential mechanisms that could explain their effectiveness (or ineffectiveness) in different settings.
   - Expected completion date: May 2020

3. Training-time compression methods for the Convolutional Neural Networks

   - Status: **In-progress**
   - Problem statement: Most existing model compression techniques optimize for the inference time and cater to the low latency requirement, such as, detection of pedestrians for self-driving cars. We argue that model training time should also be optimized since it consumes big amount of resources that contribute to substantial cost in the cloud-oriented Machine Learning as a Service environment. We aim to explore neural networks from the perspective of different domains for efficient compression during training. Our primary focus is on harnessing the FFT-based processing to limit resource usage while retaining high accuracy. Moreover, we investigate additional properties of

different transformations. The FFT, DCT, or SVD domains are interpretable, where we can reason about the expected properties based on preserved frequencies or singular values. On the other hand, Winograd algorithm is numerically most efficient for small filters but there is no interpretation of the data in the Winograd domain. This limits its adaptation to the resource control during the training time.

We will investigate how the SVD transformation can be leveraged for transition from 2D to less expensive 1D convolution, by using singular vectors as separate channels. It was observed that the first few layers of a neural network transform the image to a domain similar to DCT [24]. However, execution of convolution operation in this domain is substantially more expensive than in the FFT domain.

Our idea is to combine the DCT representation proposed by [24] and the multigrid neural architecture proposed in [34], which special case was introduced for spatial-frequency split by [13]. The DCT representation in *YCbCr* color space uses larger spatial representation for the *Y* part than for the *CbCr* parts. We want to keep them in separate channels as done in [13]. Please, see Figure 4.1.

We also plan on translating the existing model compression techniques from the inference-time to the training-time.

- Expected completion date: May 2020

Detailed research plan

1. Design and implement:
    - Information exchange between the Y, Cb, and Cr DCT blocks
    - Compression of internal representations in the DCT domain
    - Data augmentation (rotate, resize, crop, gaussian noise) in the DCT domain
2. Find faster and/or more accurate:

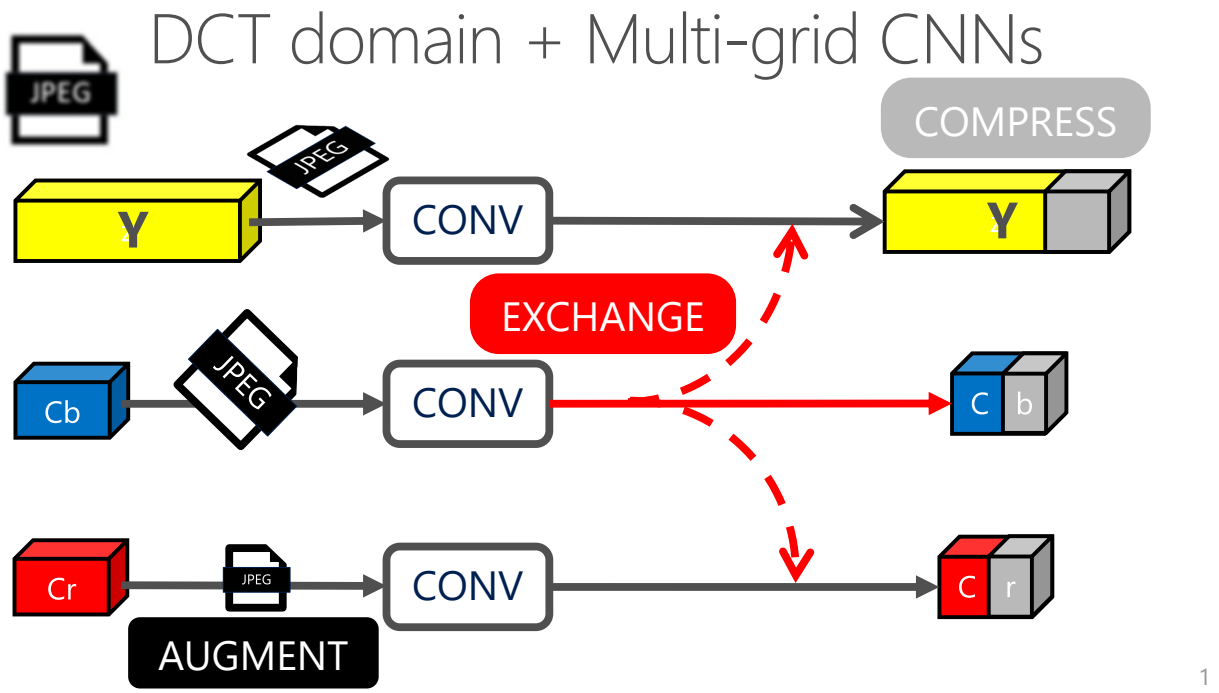

Figure 4.1: The schema of the DCT domain representation combined with the Multigrid neural architecture.

- Processing paths (conv blocks, pooling, etc.) of the Y, Cb, and Cr channels
- Size of the DCT blocks (default is 8 x 8 pixels)

3. Datasets and Base Architectures

- CIFAR-10 on ResNet-18
- CIFAR-100 on DenseNet-121
- ImageNet on ResNet-50

4. Ablation study: identify from where improvements arise

CHAPTER 5

SUMMARY

In this proposal, our main goal is to make neural networks more adaptive and robust. Our primary method relies on different transformation techniques within neural networks. The FFT-based transformation is successful in terms of performance for big filters and its additional properties make it very attractive in terms of adversarial robustness or interpretability of the internal mechanisms of neural networks. In the Winograd domain, data cannot be interpreted in the same way as in FFT or SVD domains. Currently, models learned using Winograd convolution are compressed and accelerated only after training by zeroing-out values close to zero (using a tuned threshold).

We show that different compression techniques, for example, FFT, SVD or bit-level compression, as well as random perturbations (Laplace, Gauss, or Uniform noise) lead to similar gains in robustness. Their underlying mechanism is very similar and can be summarized as inducing enough distortion to cover specific adversarial noise but not too high so that the model accuracy stays relatively high. Most of this methods are not differentiable and the gradient is hard to approximate, especially if the perturbation techniques are applied in many internal layers of neural networks. The perturbation techniques can be applied during training and then inference or exclusively during inference time. The model accuracy on clean data is usually a few percent higher and the the attack is less successful (including its adaptive version) when we include the perturbation layers during model training. However, the training time is longer since the gradients are less accurate.

REFERENCES

[1] Christopher R Aberger, Christopher De Sa, Megan Leszczynski, Alana Marzoev, Kunle Olukotun, Christopher Ré, and Jian Zhang. High-accuracy low-precision training.

[2] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer, 1993.

[3] Dan Alistarh, Christopher De Sa, and Nikola Konstantinov. The convergence of stochastic gradient descent in asynchronous shared memory. *arXiv preprint arXiv:1803.08841*, 2018.

[4] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

[5] Ayse Elvan Aydemir, Alptekin Temizel, and Tugba Taskaya-Temizel. The effects of JPEG and JPEG2000 compression on attacks using adversarial examples. *CoRR*, abs/1803.10418, 2018.

[6] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[7] Mikolaj Bińkowski, Gautier Marti, and Philippe Donnat. Autoregressive convolutional neural networks for asynchronous time series. *CoRR*, abs/1703.04122, 2017.

[8] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[9] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, May 2017.

[10] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[11] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[12] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1475–1484. ACM, 2016.

[13] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *CoRR*, abs/1904.05049, 2019.

[14] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[15] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with JPEG compression. *CoRR*, abs/1705.02900, 2017.

[16] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using JPEG compression. *CoRR*, abs/1802.06816, 2018.

[17] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

[18] Adam Dziedzic, John Paparrizos, Sanjay Krishnan, Aaron Elmore, and Michael Franklin. Band-limited training and inference for convolutional neural networks. *ICML*, 2019.

[19] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[20] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations, 2017.

[21] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, pages 419–429, New York, NY, USA, 1994. ACM.

[22] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[23] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[24] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3937–3948. Curran Associates, Inc., 2018.

[25] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. *arXiv e-prints*, page arXiv:1711.00117, Oct 2017.

[26] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[27] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

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

[29] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A new defense against adversarial images: Turning a weakness into a strength. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1633–1644. Curran Associates, Inc., 2019.

[30] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[31] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[32] Mehdi Jafarnia-Jahromi, Tasmin Chowdhury, Hsin-Tai Wu, and Sayandev Mukherjee. Ppd: Permutation phase defense against adversarial examples in deep learning. *arXiv preprint arXiv:1812.10049*, 2018.

[33] Herman Kamper, Weiran Wang, and Karen Livescu. Deep convolutional acoustic word embeddings using word-pair side information. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4950–4954, 2016.

[34] Tsung-Wei Ke, Michael Maire, and Stella X. Yu. Multigrid neural architectures. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[35] Sanjay Krishnan, Aaron J. Elmore, Michael Franklin, John Paparrizos, Zechao Shang, Adam Dziedzic, and Rui Liu. Artificial intelligence in resource-constrained and shared environments. *SIGOPS Oper. Syst. Rev.*, 53(1):1–6, July 2019.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[37] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4021, 2016.

[38] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified Robustness to Adversarial Examples with Differential Privacy. *arXiv e-prints*, page arXiv:1802.03471, Feb 2018.

[39] Sheng R. Li, Jongsoo Park, and Ping Tak Peter Tang. Enabling sparse winograd convolution by native pruning. *CoRR*, abs/1702.08597, 2017.

[40] Xingyu Liu, Jeff Pool, Song Han, and William J Dally. Efficient sparse-winograd convolutional neural networks. *arXiv preprint arXiv:1802.06367*, 2018.

[41] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 381–397, Cham, 2018. Springer International Publishing.

[42] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[43] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[44] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

[45] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. *CoRR*, abs/1710.03740, 2017.

[46] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[47] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

[48] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[49] Nasim Rahaman, Devansh Arpit, Aristide Baratin, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of deep neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

[50] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[51] V. G. Reju, S. N. Koh, and I. Y. Soon. Convolution using discrete sine and cosine transforms. *IEEE Signal Processing Letters*, 14(7):445–448, July 2007.

[52] Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. In *Advances in neural information processing systems*, pages 2449–2457, 2015.

[53] Oren Rippel, Jasper Snoek, and Ryan P. Adams. Spectral representations for convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2449–2457, Cambridge, MA, USA, 2015. MIT Press.

[54] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5498–5507, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[55] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya P. Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *CoRR*, abs/1906.04584, 2019.

[56] Kaz Sato, Cliff Young, and David Patterson. An in-depth look at google's first tensor processing unit (tpu). *Google Cloud Big Data and Machine Learning Blog*, 12, 2017.

[57] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3088–3096, 2015.

[58] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[59] Antonio Torralba and Aude Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems*, 14(3):391–412, 2003.

[60] Sathya Vanlin, Adam Dziedzic, Monisha Ghosh, and Sanjay Krishnan. Machine learning based detection of multiple wi-fi bsss for lte-u csat. *International Conference on Computing, Networking and Communications (ICNC)*, 2019.

[61] Nicolas Vasilache, Jeff Johnson, Michaël Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast convolutional nets with fbfft: A GPU performance evaluation. *ICLR*, abs/1412.7580, 2015.

[62] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, pages 7686–7695, 2018.

[63] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[64] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[65] Zhi-Qin J Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. *arXiv preprint arXiv:1807.01251*, 2018.

[66] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 2130–2141. Curran Associates, Inc., 2019.

[67] Yuchen Zhang and Percy Liang. Defending against whitebox adversarial attacks via randomized discretization. *AISTATS*, 2019.

[68] Aleksandar Zlateski, Zhen Jia, Kai Li, and Fredo Durand. Fft convolutions are faster than winograd on modern cpus, here is why, 2018.