Band-limited Training and Inference for Convolutional Neural Networks

Adam Dziedzic [*1] **John Paparrizos** [*1] **Sanjay Krishnan** [1] **Aaron Elmore** [1] **Michael Franklin** [1]

Abstract

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. We explore artificially constraining the frequency spectra of these filters and data, called band-limiting, during training. 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.

1. Introduction

Convolutional layers are an integral part of neural network architectures for computer vision, natural language processing, and time-series analysis (Krizhevsky et al., 2012; Kamper et al., 2016; Bińkowski et al., 2017). Convolutions are fundamental signal processing operations that amplify certain frequencies of the input and attenuate others. Recent results suggest that neural networks exhibit a *spectral bias* (Rahaman et al., 2018; Xu et al., 2018); 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 (Agrawal et al., 1993; Faloutsos et al., 1994; Torralba & Oliva, 2003). 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: training time, resource management, model compression, and robustness to noisy inputs.

Consider a frequency domain implementation of the convolution function that: (1) transforms the filter and the input into the frequency domain; (2) element-wise multiplies both frequency spectra; and (3) transforms the outcome product to the original domain. Let us assume that the final model is biased towards lower Fourier frequencies (Rahaman et al., 2018; Xu et al., 2018). 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 (Wang et al., 2018; Aberger et al.; Hubara et al., 2017), weight pruning (Han et al., 2015), or sparsification (Li et al., 2017), 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 instability (Wang et al., 2018; Aberger et al.). 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 (Mathieu et al., 2013), as well as new insights

[*]Equal contribution [1]Department of Computer Science, University of Chicago, Chicago, USA. Correspondence to: Adam Dziedzic <ady@uchicago.edu>, John Paparrizos <jopa@uchicago.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

about exploiting the conjugate symmetry of 2D FFTs, as suggested in (Rippel et al., 2015a). 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.

Band-limited training may additionally provide a new perspective on adversarial robustness (Papernot et al., 2015). Adversarial attacks on neural networks tend to involve high-frequency perturbations of input data (Huang et al., 2017; Madry et al., 2017; Papernot et al., 2015). Our experiments suggest that band-limited training produces models that can better reject noise than their full spectra counterparts.

Our experimental results over CNN training for time-series and image classification tasks lead to several interesting findings. First, band-limited models retain their predictive accuracy, even though the approximation error in the individual convolution operations can be relatively high. This indicates that models trained with band-limited spectra *learn to use low-frequency components*. Second, the amount of compression used during training should match the amount of compression used during inference to avoid significant losses in accuracy. Third, coefficient-based compression schemes (that discard a fixed number of Fourier coefficients) are more effective than ones that adaptively prune the frequency spectra (discard a fixed fraction of Fourier-domain mass). Finally, the test accuracy of the band-limited models gracefully degrades as a function of the compression rate.

In summary, we contribute:

1. **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.
2. **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.
3. **An extensive experimental evaluation across 1D and 2D CNN training tasks** that 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. 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) (He et al., 2018; Hinton et al., 2015; Sindhwani et al., 2015; Chen et al., 2015a). The ancillary benefits of compression and distillation, such as adversarial robustness, have also been noted in prior work (Huang et al., 2017; Madry et al., 2017; Papernot et al., 2015). One of the first approaches was called weight pruning (Han et al., 2015), but recently, the community is moving towards convolution-approximation methods (Liu et al., 2018; Chen et al., 2016). 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 (Hubara et al., 2017; Sato et al., 2017; Alistarh et al., 2018; De Sa et al., 2018). 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 (Rippel et al., 2015a; Rahaman et al., 2018; Xu et al., 2018), with applications to better initialization techniques, theoretical understanding 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 (Zlateski et al., 2018).

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 (Vasilache et al., 2015). Winograd domain was first explored in (Lavin & Gray, 2016) 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 (Gueguen et al., 2018), the JPEG quality used during encoding is 100%. The convolution via DCT (Reju et al., 2007) 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 (Lavin & Gray, 2016) 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 (Zlateski et al., 2018) suggest using FFTs on CPUs. (3) For ImageNet, both ResNet and DenseNet use 7x7 filters in their 1st layers (improvement via FFT noted by (Vasilache et al., 2015)),

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

which can be combined with spectral pooling (Rippel et al., 2015b). (4) The theoretical properties of the Fourier domain are well-understood, and this study elicits frequency domain properties of CNNs.

3. 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, 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 (Rahaman et al., 2018), 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.

3.1. 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:

$$x *_c y = F^{-1}\{(F_x[\omega] \cdot M_c[\omega]) \cdot (F_y[\omega] \cdot M_c[\omega])\} \quad (1)$$
$$= F^{-1}(S[\omega] \cdot M_c[\omega]) \quad (2)$$

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.

3.2. 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). 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.

3.2.1. 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 (Zlateski et al., 2018). 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 more practical for smaller filter sizes.

3.2.2. Band-limiting technique

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

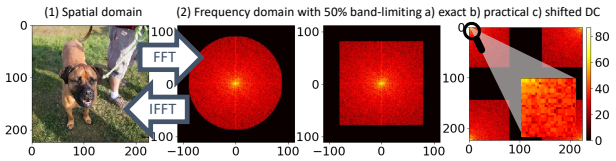

Figure 1. Transformations from input image to compressed FFT map. (1) Natural image in the spatial domain. (2) FFT transformation to frequency domain and a) exact band-limiting to 50%, b) practical band-limiting to 50%, c) lowest frequencies shifted to corners. The heat maps of magnitudes of Fourier 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.

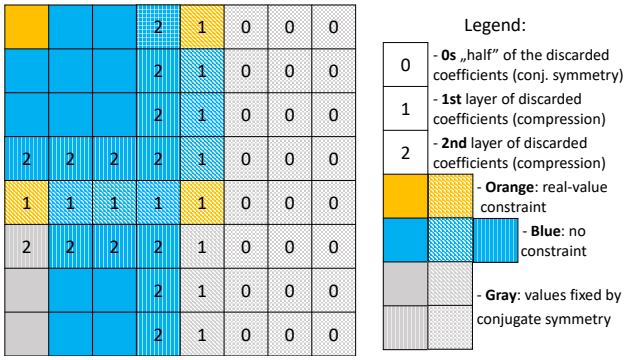

*Figure 2. An example of a square input map with marked conjugate symmetry (**Gray** cells). Almost half of the input cells marked with **0**s (zeros) are discarded first due to the conjugate symmetry. The remaining map is compressed layer by layer (we present how the first two layers: **1** and **2** 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.*

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 ($\lfloor \frac{N}{2} \rfloor + 1$) 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 real input map is of size $M \times N$, then its complex representation in the frequency domain is of size $M \times (\lfloor \frac{N}{2} \rfloor + 1)$. The real constraints for 2D inputs were explained in detail in Figure 2, similarly to (Rippel et al., 2015a). 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.

3.2.3. 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 $N \times (\lfloor \frac{N}{2} \rfloor + 1)$. 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* $(N + P) \times (\lfloor \frac{(N+P)}{2} \rfloor + 1)$ that is finally compressed.

3.3. 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 (Mathieu et al., 2013) (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

Table 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

CUDA the fast algorithm to multiply complex numbers with 3 real multiplications instead of 4 as described in (Lavin & Gray, 2016).

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).

4. 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.

4.1. 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 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 3, we explore two different compression schemes: (1) fixed compression, which discards the same percentage

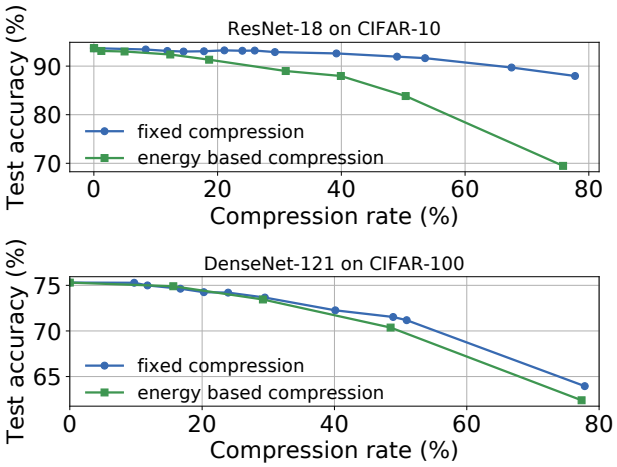

Figure 3. Test accuracy as a function of the compression rate for ResNet-18 on CIFAR-10 and DenseNet-121 on CIFAR-100. The fixed compression scheme that uses the same compression rate for each layer gives the highest test accuracy.

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 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 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.

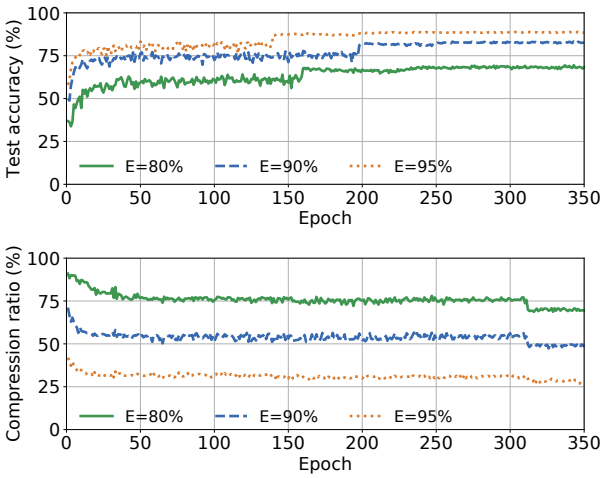

Figure 4. Compression changes during training with constant energy preserved: the longer we train the models with the same energy preserved, the smaller compression is applied. The compression rate (%) is calculated based on the size of the intermediate results for the FFT based convolution. E - is the amount of energy (in %) preserved in the spectral representation: 80, 90 and 95. We trained ResNet-18 models on CIFAR-10 for 350 epochs. The best test accuracy levels achieved by the models are: 69.47%, 83.37% and 88.99%, respectively.

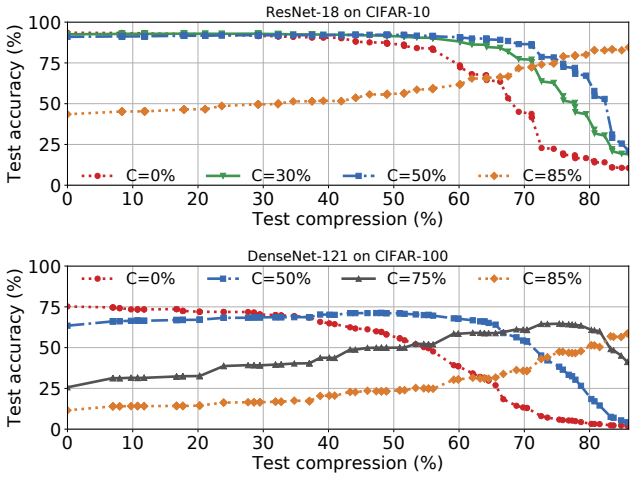

Figure 5. The highest accuracy during testing is for the same compression level as used for training and the test accuracy degrades smoothly for higher or lower levels of compression. First, we train models with different compression levels (e.g. DenseNet-121 on CIFAR-100 with compression rates: 0%, 50%, 75%, and 85%). Second, we test each model with compression levels ranging from 0% to 85%.

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, which is shown in Figure 4. 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).

4.2. Training Compression vs. Inference Compression

Having shown a smooth degradation in performance for various compression rates, we now study the effect of changing the compression rates during training and inference phases. This scenario is useful during dynamic resource allocation in model serving systems.

Figure 5 illustrates the test accuracy of ResNet-18 and DenseNet-121 models trained with specific coefficient compression rates (e.g., 0%, 50%, and 85%) while the compression rates are changed systematically during inference. We observe that the models achieve their best test accuracy when the same level of compression is used during training and inference. In addition, we performed the same experiment across 25 randomly chosen time-series datasets from the UCR archive (Chen et al., 2015b) using a 3-layer Fully Convolutional Network (FCN), which has achieved state-of-the-art results in prior work (Wang et al., 2017). We used the Friedman statistical test (Friedman, 1937) followed by the post-hoc Nemenyi test (Nemenyi, 1962) to assess the performance of multiple compression rates during inference over multiple datasets (see supplementary material for details). Our results suggest that the best test accuracy is achieved when the same compression rate is used during training and inference and, importantly, the difference in accuracy is statistically significantly better in comparison to the test accuracy achieved with different compression rate during inference.

Overall, our experiments show that band-limited CNNs learn the constrained spectrum and perform the best for similar constraining during inference. In addition, the smooth degradation in performance is a valuable property of band-limited training as it permits outer optimizations to tune the compression rate parameter without unexpected instabilities or performance cliffs.

4.3. Comparison Against Reduced Precision Method

Until now, we have demonstrated the performance of band-limited CNNs in comparison to the full spectra counterparts. It remains to show how the compression mechanism compares against a strong baseline. Specifically, we evaluate band-limited CNNs against CNNs using reduced precision

Table 2. Resource utilization (RES. in %) for a given precision and compression rate (SETUP). MEM. ALLOC. - the memory size allocated on the GPU device, MEM. UTIL. - percent of time when memory was read or written, GPU UTIL. - percent of time when one or more kernels was executing on the GPU. C - denotes the compression rate (%) applied, e.g., FP32-C=50% is model trained with 32 bit precision for floating point numbers and 50% compression applied.

RES(%) \ SETUP	FP32-C=0%	FP16-C=0%	FP32-C=50%	FP32-C=85%
AVG. MEM. ALLOC.	6.69	4.79	6.45	4.92
MAX. MEM. ALLOC.	16.36	11.69	14.98	10.75
AVG. MEM. UTIL.	9.97	5.46	5.54	3.50
MAX. MEM. UTIL.	41	22	24	20
AVG. GPU UTIL.	24.38	22.53	21.70	16.87
MAX. GPU UTIL.	89	81	74	70
TEST ACC.	93.69	91.53	92.32	85.4

arithmetic (RPA). RPA-based methods require specialized libraries [2] and are notoriously unstable. They require significant architectural modifications for precision levels under 16-bits–if not new training chipsets (Wang et al., 2018; Aberger et al.). From a resource perspective, band-limited CNNs are competitive with RPA-based CNNs–without requiring specialized libraries. To record the memory allocation, we run ResNet-18 on CIFAR-10 with batch size 32 and we query the VBIOS (via nvidia-smi every millisecond in the window of 7 seconds). Table 2 shows a set of basic statistics for resource utilization for RPA-based (fp16) and band-limited models. The more compression is applied or the lower the precision set (fp16), the lower the utilization of resources. In the supplementary material we show that it is possible to combine the two methods.

4.4. 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 6, 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 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 (Rauber et al., 2017). 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 (Carlini & Wagner, 2017) (band-limited convolutions return proper gradients). Fourier properties suggest further investigation of invariances under adversarial rotations and translations (Engstrom et al., 2017).

[2] https://devblogs.nvidia.com/apex-pytorch-easy-mixed-precision-training/

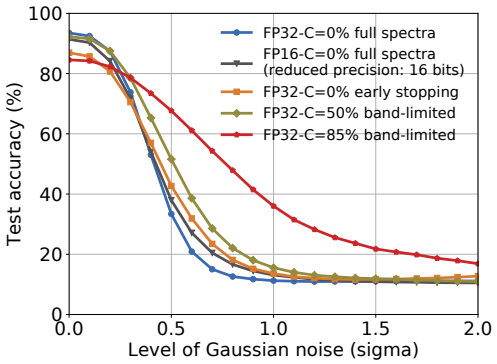

Figure 6. Input test images are perturbed with Gaussian noise, where the sigma parameter is changed from 0 to 2. The more band-limited model, the more robust it is to the introduced noise. We use ResNet-18 models trained on CIFAR-10.

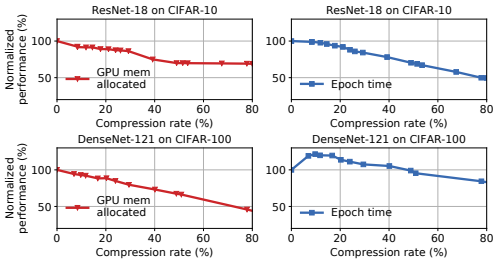

Figure 7. Normalized performance (%) between models trained with different FFT-compression rates.

4.5. Control of the GPU and Memory Usage

In Figure 7, 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: $\frac{\text{metric value for a compressed model}}{\text{metric value for the full spectra model}} \cdot 100\%$.

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

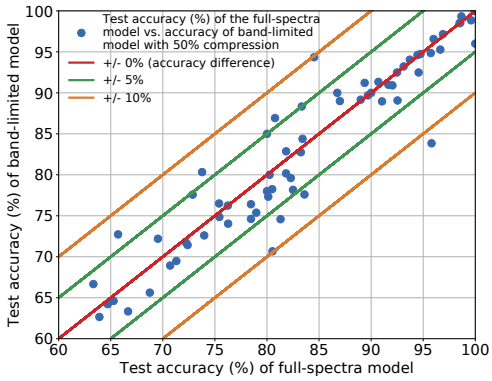

Figure 8. Comparing test accuracy (%) on a 3-layer FCN architecture between full-spectra models (100% energy preserved, no compression) and a band-limited models with 50% compression rate for time-series datasets from the UCR archive. The red line indicates no difference in accuracy between the models while green and orange margin lines show +/- 5% and +/- 10% differences.

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.

4.6. Generality of the Results

To show the applicability of band-limited training to different domains, we apply our technique using the FCN architecture discussed previously on the time-series datasets from the UCR archive. Figure 8 compares the test accuracy between full-spectra (no compression) and band-limited (with 50% compression) models with FFT-based 1D convolutions. As with the results for 2D convolutions, we find that not only is accuracy preserved but there are very significant savings in terms of GPU memory usage (Table 3).

5. Conclusion and Future Work

Our main finding is that compressing a model in the frequency domain, called band-limiting, gracefully degrades the predictive accuracy as a function of the compression rate. In this study, we also develop principled schemes to reduce the resource consumption of neural network training. Neural networks are heavily over-parametrized and modern compression techniques exploit this redundancy. Reducing this footprint during training is more challenging than during inference due to the sensitivity of gradient-based optimization to noise.

Table 3. Resource utilization and accuracy for the FCN network on a representative time-series dataset (see supplement for details).

ENERGY PRESERVED (%)	AVG. GPU MEM USAGE (MB)	MAX. TEST ACCURACY (%)
100	118	64.40
90	**25**	**63.52**
50	21	59.34
10	17	40.00

While implementing an efficient band-limited convolutional layer is not trivial, one has to exploit conjugate symmetry, cache locality, and fast complex arithmetic, no additional modification to the architecture or training procedure is needed. Band-limited training provides a continuous knob to trade-off resource utilization vs. predictive performance. But beyond computational performance, frequency restriction serves as a strong prior. If we know that our data has a biased frequency spectra or that the functions learned by the model should be smooth, band-limited training provides an efficient way to enforce those constraints.

There are several exciting avenues for future work. Trading off latency/memory for accuracy is a key challenge in streaming applications of CNNs, such as in video processing. Smooth tradeoffs allow an application to tune a model for its own Quality of Service requirements. One can also imagine a similar analysis with a cosine basis using a Discrete Cosine Transform rather than an FFT. There is some reason to believe that results will be similar as this has been applied to input compression (Gueguen et al., 2018) (as opposed to layer-wise compression in our work). Finally, we are interested in out-of-core neural network applications where intermediate results cannot fit in main-memory. Compression will be a key part for such applications. We believe that compression can make neural network architectures with larger filter sizes more practical to study.

We are also interested in the applications of Band-limited training to learned control and reinforcement learning problems. Control systems are often characterized by the impulse response of their frequency domains. We believe that a similar strategy to that presented in this paper can be used for more efficient system identification or reinforcement algorithms.

Acknowledgements

We thank the reviewers and our colleagues for their valuable feedback. This research was supported in part by the Center for Unstoppable Computing (CERES) at University of Chicago, NSF CISE Expeditions Award CCF-1139158, generous support from Google and NVIDIA, and the NSF Chameleon cloud project.

References

Aberger, C. R., De Sa, C., Leszczynski, M., Marzoev, A., Olukotun, K., Ré, C., and Zhang, J. High-accuracy low-precision training.

Agrawal, R., Faloutsos, C., and Swami, A. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pp. 69–84. Springer, 1993.

Alistarh, D., De Sa, C., and Konstantinov, N. The convergence of stochastic gradient descent in asynchronous shared memory. *arXiv preprint arXiv:1803.08841*, 2018.

Bińkowski, M., Marti, G., and Donnat, P. Autoregressive convolutional neural networks for asynchronous time series. *CoRR*, abs/1703.04122, 2017. URL http://arxiv.org/abs/1703.04122.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

Chen, W., Wilson, J., Tyree, S., Weinberger, K., and Chen, Y. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pp. 2285–2294, 2015a.

Chen, W., Wilson, J., Tyree, S., Weinberger, K. Q., and Chen, Y. Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1475–1484. ACM, 2016.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. The ucr time series classification archive, July 2015b. www.cs.ucr.edu/~eamonn/time_series_data/.

De Sa, C., Leszczynski, M., Zhang, J., Marzoev, A., Aberger, C. R., Olukotun, K., and Ré, C. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. A rotation and a translation suffice: Fooling cnns with simple transformations, 2017.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, pp. 419–429, New York, NY, USA, 1994. ACM. ISBN 0-89791-639-5. doi: 10.1145/191839.191925. URL http://doi.acm.org/10.1145/191839.191925.

Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

Gueguen, L., Sergeev, A., Kadlec, B., Liu, R., and Yosinski, J. Faster neural networks straight from jpeg. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3937–3948. Curran Associates, Inc., 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

Kamper, H., Wang, W., and Livescu, K. Deep convolutional acoustic word embeddings using word-pair side information. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4950–4954, 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

Lavin, A. and Gray, S. Fast algorithms for convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4013–4021, 2016.

Li, S. R., Park, J., and Tang, P. T. P. Enabling sparse winograd convolution by native pruning. *CoRR*, abs/1702.08597, 2017. URL http://arxiv.org/abs/1702.08597.

Liu, X., Pool, J., Han, S., and Dally, W. J. Efficient sparse-winograd convolutional neural networks. *arXiv preprint arXiv:1802.06367*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mathieu, M., Henaff, M., and LeCun, Y. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

Nemenyi, P. Distribution-free multiple comparisons. In *Biometrics*, volume 18, pp. 263. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

Rahaman, N., Arpit, D., Baratin, A., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the spectral bias of deep neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017. URL http://arxiv.org/abs/1707.04131.

Reju, V. G., Koh, S. N., and Soon, I. Y. Convolution using discrete sine and cosine transforms. *IEEE Signal Processing Letters*, 14(7):445–448, July 2007. ISSN 1070-9908.

Rippel, O., Snoek, J., and Adams, R. P. Spectral representations for convolutional neural networks. In *Advances in neural information processing systems*, pp. 2449–2457, 2015a.

Rippel, O., Snoek, J., and Adams, R. P. Spectral representations for convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2449–2457, Cambridge, MA, USA, 2015b. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969442.2969513.

Sato, K., Young, C., and Patterson, D. An in-depth look at googles first tensor processing unit (tpu). *Google Cloud Big Data and Machine Learning Blog*, 12, 2017.

Sindhwani, V., Sainath, T., and Kumar, S. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pp. 3088–3096, 2015.

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

Vasilache, N., Johnson, J., Mathieu, M., Chintala, S., Piantino, S., and LeCun, Y. Fast convolutional nets with fbfft: A GPU performance evaluation. *ICLR*, abs/1412.7580, 2015. URL http://arxiv.org/abs/1412.7580.

Wang, N., Choi, J., Brand, D., Chen, C.-Y., and Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, pp. 7686–7695, 2018.

Wang, Z., Yan, W., and Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017. doi: 10.1109/ijcnn.2017.7966039. URL http://dx.doi.org/10.1109/IJCNN.2017.7966039.

Xu, Z.-Q. J., Zhang, Y., and Xiao, Y. Training behavior of deep neural network in frequency domain. *arXiv preprint arXiv:1807.01251*, 2018.

Zlateski, A., Jia, Z., Li, K., and Durand, F. Fft convolutions are faster than winograd on modern cpus, here is why, 2018.