Parametric Spectral Filters for Fast Converging, Scalable Convolutional Neural Networks

Luke Wood
Google
San Diego, CA USA
lukewood@google.com

Eric C. Larson
Southern Methodist University
Dallas, TX USA
eclarson@lyle.smu.edu

***Abstract*—Using spectral multiplication to compute convolution in neural networks has been investigated by a number of researchers because of its potential in speeding up computations for large images. However, previous methods require the learning of arbitrarily large convolution filters in the spectral domain, causing two untenable problems: an explosion in the number of trainable parameters per filter and an inability to reuse filters across images of differing sizes. To address this, we propose the usage of spectral approximation functions to approximate the massive Spectral domain filters with only a few trainable parameters. Our empirical analysis suggests that the proposed approximation maintains the benefits of arbitrarily large filters (such as improved rate of convergence in training, accuracy, and stability) while relying on significantly fewer trainable parameters.**

I. Introduction

Traditional convolutional neural networks (CNNs) are computed in the spatial domain and are typically of discrete sizes, such as 3x3xC when applied to images or activations with C channels [1]. These filters are often limited spatially to 3x3 for computational efficiency and because increasing complexity can be attained by cascading multiple 3x3 convolutions, as shown for image-based convolution by Szegedy et al. [2]. However, convolution applied in the spatial domain requires $O(N^2 \cdot k^2)$ operations with respect to image size NxN and filter size kxk in the spatial domain. Despite this complexity, efficient parallel implementations have been developed that exploit the massively data parallel processing of graphic processing units, GPUs [3].

It is well known that the Fourier Convolution Theorem allows for convolution with significantly fewer computations, with complexity of $O\left(N^2 \cdot \log(k^2)\right)$. It would seem straightforward, then, to replace the traditional convolutions applied in convolutional neural networks with equivalent Fourier domain operations—however, the process is not as simple as first glance would impart. Lin et al. investigated the use of Fourier operations for speeding up convolutional layers, with mixed results. They observed that some performance benefits [4], [5] can be achieved by applying a single Fast Fourier Transform (FFT), which runs in $O\left(N^2 \cdot \log(N^2)\right)$ complexity [6], and reusing the result through multiple convolutions—especially for large images. In addition to this, filters learned in the resulting space correspond to filters in the spatial domain bounded only by input size. Unfortunately, the process of learning the filter weights had little benefit from this operation because traditional back-propagation works directly on activation maps to update the spatial weights—the speedup in convolution was therefore only seen after training.

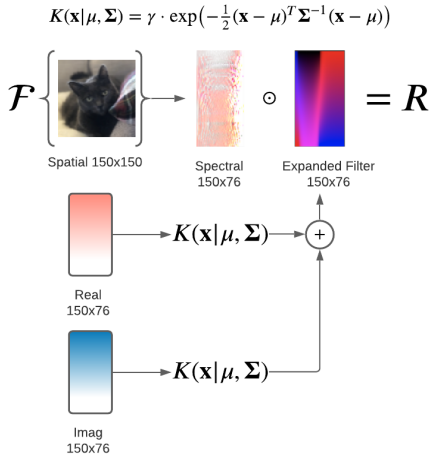

Fig. 1. Parametric Mapping Approach for Spectral Convolution

To help mitigate this, Pratt et al. [7] introduced a method that optimizes filter weights in the Fourier domain, which they called Fourier or Spectral Convolutional Networks. The key advantage of this approach is that convolutional filters are trained without ever being converted to their spatial representations. They implement spectral convolutional filters using trainable weight matrices with spatial dimensions identical to that of the input image. This allows for point-wise multiplication to serve as convolution (applied and optimized in the Fourier domain). Their experiments (reproduced in this paper also) show that these filters converge faster and yield higher accuracy than their fixed size spatial domain counterparts, but suffer from having a large number of trainable parameters.

Therefore, the optimization of Pratt et al. [7] introduces complexity through an explosion of the number of trainable parameters in the filter. That is, while these large spectral filters pose some benefits over the traditional spatially applied filters, each requires $M \cdot N$ trainable parameters, where M

and N correspond to the width and height of the spectral filter representations. This necessarily means that the size of the input image also influences the trainable parameters in the filter throughout the network, which is starkly different than when applying convolution in the spatial domain. This poses unacceptable memory consumption when working with large scale images compared to the constant 3x3 or 2x2 parameters in the spatial domain. Table I illustrates this, showing the number of trainable parameters required for spectral convolution layers when operating on images of varying sizes.

Image Dimensions	Params Per Filter	Per 64 Filter Layer
16x16	256	16k
32x32	1024	65k
150x150	22.5k	1.4m
640x320	204k	13.1m
860x600	516k	33m
1024x768	786k	50m

TABLE I
PARAMETERS PER SPECTRAL CONVOLUTION LAYER

To mitigate this explosion of trainable parameters, we propose the use of a parametric function that approximates the spectral representation of these filters (Figure 1). In doing so, the parameters of the function are optimized through back-propagated gradient descent, drastically reducing the number of trainable parameters while maintaining the increase in accuracy and improved convergence rates. Table II shows the proposed method's ability to maintain a constant size parameter set with respect to image size. By updating the parameters of the function, the total number of trainable parameters is *fewer* than spatially-applied convolution. Moreover, we define efficient algorithms for calculating two functions via the Keras API [8] for linear and 2D Gaussian Fourier filters.

	Spatial	Spectral	Linear	2D Gaussian
150x150	9	22.5k	4	12
860x600	9	516k	4	12
1024x768	9	786k	4	12

TABLE II
NUMBER OF PARAMETERS PER FILTER, BY APPROACH

We outline our contributions as follows: (1) we propose the use of the Fourier Convolution Theorem in the context of convolutional neural network optimization with parametric functions, (2) we propose a parametric function approximation for reducing the parameter explosion for learning directly from the learned parameters with fast calculations of the spectral representations, and (3) we evaluate the complexity of our approach as compared to a traditional convolutional neural network and the methods of Pratt et al. [7], showing improved convergence and accuracy on a simple image classification benchmark.

II. RELATED WORKS

As mentioned, Pratt et al. [7] proposed the concept of Fourier Convolutional Neural Networks. These networks rely on the *Fourier Convolution Layer*, which we call the *Spectral Convolution Layer*. These layers rely on the Fourier transform of input images and a weight matrix. The weight matrix size is equivalent to that of the input images. We replicate the results in our work, showing that, while more accurate, this method results in an explosion in the number of parameters. To mitigate this, we propose a parametric function that approximates the FFT of a filter and allows the network to learn directly from the parameters of the function, rather than the entire spectrum of the filter.

Recently, Lin et al. [4] focused on the potential speedup resulting from the Fourier Convolution Theorem. In their work they use pre-trained convolutional neural networks and adapt the feedforward convolutional operations to use the FFT. They show that, for large images, this can indeed cause a speedup in processing. However, for smaller images or smaller spatial layers (such as layers after large pooling has been applied) the advantages are not consistent. in contrast, our work focuses on the advantages of optimizing the spectral representation of a filter, rather than the computational benefit of the FFT.

Rippel et al. [9] proposed that the spectral domain is a powerful representation format for convolutional neural network training. In addition to boasting the aforementioned performance benefits of [7], the work presents an effective downsampling method in the spectral domain and demonstrates the effectiveness of Spectral CNNs for classification and approximation. They show that learning filters in the Spectral domain yields improved convergence rates by a factor of 2-5x. In our work, we manage to maintain the 2-5x increased convergence rate in a classification setting while also reducing the number of parameters from a factor of $Width * Height$ to a constant value.

III. METHODOLOGY

A. Spectral Filters

Spectral convolutional layers are used to generate a baseline in each of our benchmarks. Traditional convolutional layers follow the procedure shown in the following equation, where X is an input (either an image or activation from previous layer) with C channels, $f = [f_1, f_2, ..., f_F]$ is a list filters to be convolved with X, and r_i is a tensor of the output activation corresponding to f_i:

$$r_i = \text{Conv2D}(X, f_i)$$
$$\mathcal{F}(r_i) = R_i = \sum_{c=1}^{C} \mathcal{F}(X_c) \odot \mathcal{F}(f_i)$$

where $\odot$ denotes the Hadamard (or element-wise) product. To optimize the filters directly in the Fourier domain, we replace $\text{FFT}(f_i)$ with its Fourier representation F_i, a complex matrix with dimensions M, N corresponding the height and width of the input X. This is a naïve approach, that ignores parameter explosion and is given by the following equation, where X is the input image, and subsequent layers operations in the neural network are denoted by the superscript (L). For instance, $R_i^{(L)}$ is the Fourier representation of the input activation for channel i in layer L:

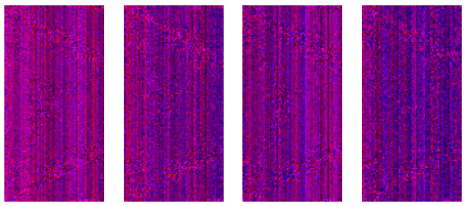

Fig. 2. Spectral Filters $h = 150, w = 76$, each pixel represent a complex number where the real value is given in blue and the imaginary value in red.

Fig. 3. Expanded Linear Filters $h = 150, w = 76$, each pixel represent a complex number where the real value is given in blue and the imaginary value in red.

$$R_i^{(1)} = \sum_{c=1}^{C} \mathcal{F}(X_c) \odot F_i^{(0)}$$
$$R_i^{(L+1)} = \sum_{c=1}^{C^{(L)}} R_c^{(L)} \odot F_i^{(L)}$$

Some example filters trained using this approach can be found in Figure 2. As the filters consist of complex numbers, blue represents the real portion of the filter and red the imaginary. Notice that no discernible structure is present in the filters that would be typically expected of common filters in the Fourier domain.

B. Parametric Filter Mapping

As noted, the biggest disadvantage of the naïve approach is that the spectral filters $F_i^{(L)}$ require $M \cdot N$ trainable parameters each. We remedy this by replacing F with $\hat{F}$, which is defined as the evaluation of a parametric function $K(\theta)$ for each point in the Fourier space. The parameters of the function, θ, are optimized using gradient descent.

$$R_i^{(1)} = \sum_{c=1}^{C} \mathcal{F}(X_c) \odot K_i^{(0)}(\theta)$$
$$R_i^{(L+1)} = \sum_{c=1}^{C^{(L)}} R_c^{(L)} \odot K_i^{(L)}(\theta)$$

We investigate two different parametric functions, K, linear and Gaussian. The linear parametric function is given by:

$$K(\mathbf{x}|s_1, s_2) = \mathbf{x} \cdot \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

where $\mathbf{x}$ spans the range of the 2-dimensional Fourier space. This function assumes that the filter is approximated well with only a slope in the x or y directions. The function $K(\mathbf{x}|s_1, s_2)$ is calculated for both the real and imaginary portions, resulting in four total trainable parameters. Some example filters taken from a trained network are shown in Figure 3.

The Fourier representation of convolutional filters often results in a high energy cluster around the origin, with decreasing energy in the higher frequencies. We theorized that the the symmetric 2D Gaussian function could efficiently approximate this behavior. The Gaussian parametric function is given by:

$$K(\mathbf{x}|\mu, \mathbf{\Sigma}) = \gamma \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu)\right)$$

During the optimization we allow $\mathbf{\Sigma}$ to be full covariance, which is three parameters for our 2D spectral representation (due to the symmetry of $\mathbf{\Sigma}$). With the two parameters in μ and the scaling parameter γ, there are six trainable parameters for the Gaussian function. The function $K(\mathbf{x}|\mu, \mathbf{\Sigma})$ is calculated for both the real and imaginary portions, resulting in twelve total trainable parameters. Some sample Gaussian 2D filters learned in the Cats Vs Dogs benchmark are visualized in Figure 4. We observed that that filters learned deviated from our hypothesis that they would have most energy at the DC offset (the origin of the 2D FFT plane). Instead, the learned filters have many sharp, kurtotic spikes and bands, or are gradually sloped. It was rare to find a Gaussian filter with center near the origin and reasonable variances. More investigation is needed to interpret why these kinds of functions were found–however it may be related to the "banding" learned from the unrestricted filters in Figure 2.

Brute force calculation of the functions for each (x, y) point is computationally costly. To ensure acceptable run times, the function must be computed with values dependent solely on the x or y coordinates in a single pass (resulting in two vectors) and the resulting vectors must be used to compute the entire matrix, K. Because both parametric functions can be separated by their horizontal and vertical portions (similar to separable convolution), each portions can be calculated independently, repeated along the rows or columns, and the results added together. This greatly speeds up the computation of each parametric function.

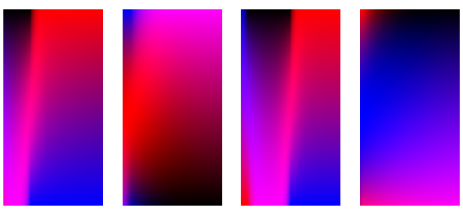

Fig. 4. 2D Gaussian Filters $150x76$, each pixel represent a complex number where the real value is given in blue and the imaginary value in red.

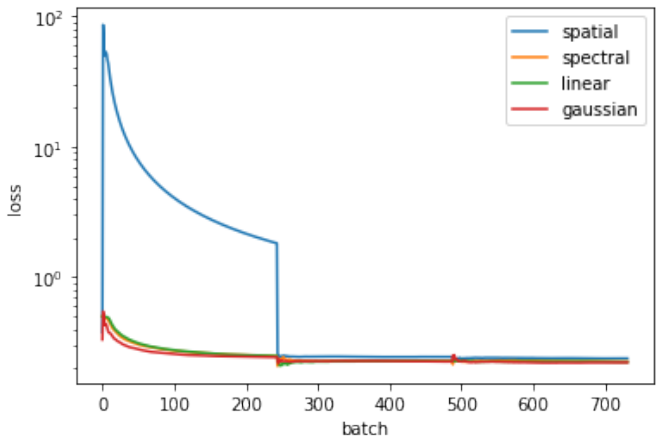

Fig. 5. Cats Vs Dogs Benchmark Learning Curves

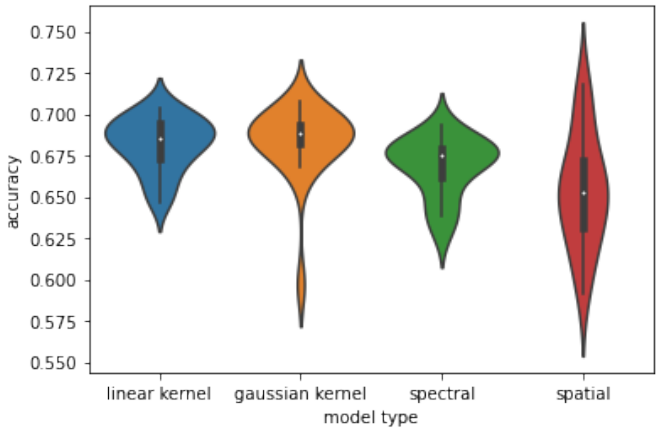

Fig. 6. Cats Vs Dogs Benchmark Accuracy Scores

IV. Benchmarks

We employ a simple benchmark in our evaluation, using the Cats versus Dogs dataset [10] provided by Microsoft. The aim of this dataset is to classify images as either a dog or cat. The data is preprocessed to be of the consistent shape 150x150 pixels and three color channels. We also applied some data augmentation in the form of random shifts and slight rotations (up to 5 pixel shifts and 5 degree rotations) We evaluate the performance of four models: a traditional CNN, a CNN using spectral convolutions, a CNN using linear spectral function approximation, and a CNN using Gaussian spectral function approximation. Because we desire to isolate the behavior of each convolution approach, no downsampling and regularization is performed.

Networks were implemented using the Keras API [8]. Each network consists of four convolutional layers, each followed by a rectified linear activation function. Each network also contains a fully connected layer at the end, with hyperbolic tangent activation to perform classification. Each network is trained 20 separate times (using consistent train and test separation for each model). That is, each training run consists of identical data between each network. Each network trained for 10 epochs before evaluation on a new set of test images. This means each network trained on 15k images 10 times per image.

Figure 5 shows one example of the loss versus batch iteration for each of the models employed. As can be seen, the spatial convolutional network converges more slowly than any of the spectral methods. The models with the best convergence are the raw spectral convolution network and the spectral convolution network with Gaussian filter map. In our experiments, this behavior was consistent, with the spectral networks converging earlier in the optimization.

Figure 6 shows a violin plot of the 20 benchmark runs for each algorithm. We observe that the two spectral approaches using parametric function approaches perform the best, followed by the spectral convolution network without parametric functions, and finally the spatial convolutional network. We note that the Spectral methods had considerably less variation.

Similarly, Table III shows that the spatial domain CNN has the lowest accuracy score and highest standard deviation. Both parametric filter methods perform exceptionally well on this task.

We theorize the performance of the parametric filters to be superior to that of the raw spectral filter due to implicit regularization—its difficult to over-fit with so few parameters. Due to the high number of parameters available to the spectral filter we suspect that the network is over-fitting. This is implicitly avoided by both the linear and Gaussian filter maps. Moreover, we also theorize that the Hadamard product helps to control over-fitting because it allows fewer connections between neurons in adjacent layers.

Mode	mean	max	median
Linear	0.681±0.016	0.704	0.684
2D Gaussian	0.683±.023	0.707	0.688
Spectral	0.668±.017	0.693	0.675
Spatial	0.653±.034	0.717	0.653

TABLE III
Cats Vs Dogs Benchmark Accuracy, number of trials= 20

V. Conclusion

We demonstrate that spectral parametric function mapped filters maintain the improved convergence and accuracy, improving on the results found by Rippel et al. [9]. We theorize based on our results that our parametric function mapped filters provide implicit regularization yielding accuracy and convergence benefits. In future work we anticipate integrating pooling methods to further improve the performance of our model, and perform evaluations outside of the image processing domain. We theorize that more sophisticated parametric functions would yield superior results to ours. Combined with the existing spectral domain downsampling proposed by Rippel et al. this proposes a powerful architecture for scalable Spectral domain deep learning.

REFERENCES

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[3] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and scalability of gpu-based convolutional neural networks," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*. IEEE, 2010, pp. 317–324.

[4] J. Lin, L. Ma, and Y. Yao, "A fourier domain acceleration framework for convolutional neural networks," vol. 364. Elsevier, 2019, pp. 254–268.

[5] ——, "A fourier domain training framework for convolutional neural networks based on the fourier domain pyramid pooling method and fourier domain exponential linear unit," *IEEE Access*, vol. 7, pp. 116 612–116 631, 2019.

[6] H. J. Nussbaumer, "The fast fourier transform," in *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, pp. 80–111.

[7] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "Fcnn: Fourier convolutional neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 786–798.

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

[9] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," 2015.

[10] "Dogs vs. cats." [Online]. Available: https://www.kaggle.com/c/dogs-vs-cats