ECBM 4040 Final Project - Spectral Representations for Convolutional Neural Networks

Aarshay Jain *(aj2713)*, Jared Samet *(jss2272)*, Alex Wainger *(atw2131)*

Abstract—This project is an attempt to implement the ideas and replicate the results of [Rippel, Snoek and Adams 2015]. The paper introduces two ideas to improve training of CNNs: spectral pooling, a novel approach to dimensionality reduction, and spectral filter parameterization, which represents convolutional filters in the spectral domain to help train the network in fewer epochs. The main technical challenge was to **not screw up the Fourier transforms, need to write this better**. Although we successfully implemented the paper's novel ideas in TensorFlow, we were unable to replicate the improved training results that the original authors report.

1 Introduction

The impressive achievements of Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision in recent years. CNNs have won the annual ImageNet competition every year since the introduction of AlexNet [paper reference] in 2012 and have rapidly come to dominate the research landscape for computer vision. As their name suggests, CNNs are deep feedforward neural networks mostly composed of successive layers that first compute the convolution (actually the cross-correlation) of the input image with a set of learned filters, then apply a nonlinear activation function to the result of the convolution, and finally reduce the dimensionality of the resulting activations through an operation such as max-pooling before repeating the process.

The Fourier transform, in both its continuous and discrete forms, maps convolution in the spatial domain to multiplication in the spectral domain and is often used internally at the hardware or software level by libraries such as NVIDIA's cuDNN that provide APIs to efficiently compute convolutions. Rippel, Snoek and Adams claim that the Discrete Fourier Transform (DFT) can be used not only to improve the efficiency of computing the convolution itself, but also to improve the training of CNNs in three distinct ways. Their first proposal is to represent the learned CNN filters as complex numbers in the frequency domain instead of as real numbers in the spatial domain. The second proposal is to replace the max-pooling layer that is typically found in CNNs with a spectral pooling layer, which shrinks the dimensionality with significantly less information loss for the same degree of parameter reduction. The third proposal is to use a modified version of the spectral pooling operation as a regularization technique similar to dropout.

Our objectives in this project were, first, to successfully implement the proposals of original paper in TensorFlow and implement a CNN employing these proposals, and, second, to replicate the improved training results reported by the authors. There were a number of challenges we encountered while implementing these novel proposals. The first was conceptual: making sure we properly understood the intricacies involved in shifting back and forth between the spatial and frequency domains for both the filters (in the case of spectral filter parameterization) and the images (for spectral pooling and frequency dropout). The second involved actually implementing the spectral pooling and frequency dropout operations. Although these operations are conceptually straightforward, implementing them correctly required careful attention to detail. The third challenge we faced involved backpropagating the gradient to the the convolution filters when they were parameterized as complex numbers in the frequency domain. The final challenge was faithfully replicating the details of the authors' multiple network architectures, which are described tersely with many different parameters.

2 Original Paper

2.1 Methodology

The original paper proposes spectral filter parameterization as an alternate formulation for the learned weight matrices of CNNs. In this formulation, which has the equivalent expressive power and capacity as the traditional formulation, the parameters are represented as the complex coefficients of the DFT of the traditional convolution filters. The authors claim that this formulation is more suitable for optimization than the traditional formulation and compare the convergence time (measured in optimization epochs, not in wall-clock time) of CNNs parameterized with spatial to CNNs parameterized with spectral filters.

The authors also propose spectral pooling as an alternative to max-pooling for reducing the dimensionality of the image as it is transformed throughout the neural network. They argue that this pooling operation preserves information better than max-pooling and quantify this by comparing the approximation loss of the two pooling techniques. They propose a network architecture consisting of alternating convolutional and spectral-pooling layers in which the only form regularization is frequency dropout and weight decay and achieve impressive accuracy rates on the CIFAR-10 and CIFAR-100 datasets by optimizing the hyperparameters of their architecture.

2.2 Key Results

The potential explanatory variables we included in our models were: The first key result of the original paper is that spectral pooling preserves information better than max pooling, which they demonstrate both visually, by showing a series of images whose dimensionality has been reduced by the two methods, and quantitatively, by computing the approximation loss, measured as the L2 norm of the difference of the pooled and unpooled images. The second key result is the error rates of 8.6% and 31.6%, respectively, that they achieve on the CIFAR-10 and CIFAR-100 datasets with their proposed CNN architecture. The third key result is the improved convergence time they achieve using the Adam optimizer. They measure this time by comparing the number of training epochs for the traditionally parameterized CNN and the spectrally parameterized CNN to converge to the same error rate.

3 Methodology

3.1 Implementation

We first implemented the paper's three proposals in TensorFlow. Since the concepts proposed were not immediately obvious upon reading the original paper, we have included a description here of precisely how these were implemented; we trust that these proposals accurately reflect the authors' intentions.

3.1.1 Spectral Filter Parameterization

In a traditionally parameterized CNN, a single convolutional layer transforms a three-dimensional tensor of shape $(C_{input}, H_{input}, W_{input})$ to a three-dimensional tensor of shape $(C_{output}, H_{output}, W_{output})$ by computing the cross-correlation of the three-dimensional volume at every x- and y-dimension of the input tensor with C_{output} separate filters. Here, C denotes the number of channels of the tensor, H denotes the height of the tensor and W denotes the width of the tensor. For each of the C_{output} filters, the learned parameters involved in this transformation consist of a set of real numbers that constitute the weights of the filter and a bias term. In a spectrally parameterized CNN, the convolution operation itself is identical, but the learned parameters are the complex coefficients of the DFT of the filter as opposed to the filter weights themselves. Given an input tensor and the complex numbers that constitute the spectral parameters for the filter, the output tensor is computed by first computing the inverse DFT of the filter and then convolving the resulting filter with the input tensor as usual.

We implemented this proposal in TensorFlow by writing a custom convolutional layer class where the Variables (TensorFlow's learned parameters) correspond to the complex DFT coefficients instead of the filter weights.

3.1.2 Spectral Pooling

In most CNN architectures, the operation of the network gradually transforms an input image with 3 channels (R, G, and B values) and a large height and weight dimension (e.g. 3 x 32 x 32 or 3 x 224 x 224) to an output "image" consisting of a large (e.g., 512) number of channels and a small (e.g. 5x5, 3x3, or even 1x1) number of height and weight "pixels". Different CNN architectures achieve this in different ways, but one common approach is to use max-pooling to downsample the image periodically; for example, after every convolutional layer, or after every other convolutional layer, max-pooling may be used to cut the height and weight of the image roughly in half. With spectral pooling, the dimensionality reduction is instead performed by:

- Computing the two-dimensional DFT of the input tensor for each input channel
- Truncating the resulting frequency matrix to the desired output dimensionality
- Computing the inverse DFT of the truncated frequency matrix to obtain the output tensor for the given channel

The truncation operation requires careful attention to detail in order to produce a real-valued output tensor. The DFT of a real-valued signal in the spatial domain obeys certain complex symmetries in the frequency domain; specifically, F[s, t] is the complex conjugate of F[-s, -t]; and, for a finite frequency matrix, if (s,t) and (-s,t) coincide modulo the dimension of the matrix, then F[s,t] must be real-valued; for example, the DC component, F[0,0] must be real valued, and for a 16 x 16 frequency matrix, F[8,8], F[0,8] and F[8,0] must also be real-valued. As a result, when an even-numbered output dimension is desired, the input frequency matrix cannot simply be truncated; the complex symmetries governing the original frequency matrix are not the same as the ones governing the truncated matrix. The paper refers to the need to handle these corner cases carefully but does not include the algorithm itself.

We implemented this proposal in TensorFlow by writing a custom spectral pooling function and layer class that always produces a truncated frequency matrix obeying the required complex symmetries. As with max-pooling, this layer has no learned parameters.

3.1.3 Frequency Dropout

This proposal is related to spectral pooling and is implemented in the same class, but serves a different purpose. The frequency matrix is computed for each input channel as before, but instead of being truncated to a smaller dimensionality, a brick-wall low-pass filter is instead applied: the higher frequencies are simply set to zero. The cutoff frequency for the filter is chosen randomly for each minibatch as a method of regularization. In the paper, this regularization is applied as part of the spectral pooling operation.

We implemented this proposal in TensorFlow by writing a function to perform the required filtering and incorporating the function into the spectral pooling layer.

3.2 Result Replication

We attempted to reproduce the three key results of the original paper. The impressive classification rates in the original paper required an intense hyperparameter optimization over six separate hyperparameter dimensions that would have required more computational resources than we had available for this project, but the authors report most of the results of their optimization. For the parameters where they reported optimal results, we used those and attempted to search over the others. For the other two key results, we followed their methodology as closely as possible.

3.2.1 Information Preservation

3.2.2 Spectral Pooling and Frequency Dropout for Traditionally Parameterized CNNs

We implemented a CNN class implementing the architecture specified in section 5.1 of the original paper. This architecture begins with M pairs of alternating layers of convolutional and spectral-pooling/frequency dropout layers, followed by a 1x1 convolutional layer with multiple filters, a 1x1 convolutional layer outputting the number of target classes, finally followed by a global averaging layer over which the softmax was computed for classification. The optimal hyperparameters which were not specified in the original paper were the number of layers and the weight decay rate. We performed hyperparameter search over a subsample of CIFAR-10 dataset to identify the optimal parameters. As we used the Adam optimizer instead of SGD with momentum, we added the initial learning rate and the dimensionality decay rate gamma to the set of hyperparameters over which we searched. We anneal the learning rate by 50% after every ten epochs. We randomly chose parameters on a log scale for the L2 norm and the learning rate and on a uniform scale for gamma and the number of layers M. We ran this search in parallel across multiple GPU instances and trained the network on the full datasets once the optimal hyperparameters had been identified. In addition to the random hyperparameter search, we also attempted to identify good hyperparameters by manually tuning.

3.2.3 Optimization Convergence Speed

To test the convergence speed of spectral parameterization, we created two new CNN architectures ? deep and generic. The deep architecture is defined in the paper as two 96-filter convolutional layers, a 3x3 stride 2 max pool, three 192-filter convolutional layers, a 3x3 stride 2 max pool, a 192-filter 1x1 convolutional layer, a 10-filter 1x1 convolutional layer, and a global averaging layer. The generic architecture is a more standard CNN, with a 96-filter convolutional layer, 3x3 stride 2 max pool, 192-filter convolutional layer, 3x3 stride 2 max pool, then fully connected layers with sizes of 1024 and 512. Both architectures end with a softmax to compute the probabilities of each sample belonging to each of the 10 classes. We also tested with the spectral pooling architecture described in section 3.2.2. For each of these three architectures, we trained 4 different configurations ? 3x3 kernels with spectrally parameterized conv weights, 3x3 kernels with standard conv weights, and the same thing with 5x5 kernels.

With all of our architectures defined, we used a standard training procedure, feeding mini-batches of 128 images through the network before computing and applying the gradient updates in an effort to minimize the loss (which was the sum of the cross entropy and l2 losses). We added vertical translations and horizontal flips after each epoch to increase the difficulty of training, as the paper indicates. For the models with spectral parameterization, we used the spectral convolution layer that was described in section 3.1.1. The paper mentioned that they used a Bayesian optimizer to find the best hyperparameter values, but they did not actually state what those values were for the generic and deep architectures, so we hand-tuned the networks on small samples of the training data before training on the larger dataset. Due to the number of models involved with this test and the computational intensity that spectral pooling and spectral parameterization add, we opted to train them on just one of the five batches of CIFAR10 data (i.e. 10,000 of the 50,000 samples in the training set), and even that took 12 hours on a Google Cloud GPU. The results of this test are discussed in section 5.1.3.

4 IMPLEMENTATION

4.1 Deep Learning Network

4.2 Software Design

5 RESULTS

5.1 Project Results and Comparison

5.1.1 Information Preservation

A FIGURE GOES HERE

We present a plot of the average information dissipation for the CIFAR-10 ¡¡¡test¿¿¿ set. The y-axis displays the L2 error normalized by the norm of the input images. The red line indicates the error rate achieved by max-pooling 2x2 regions with a stride of 2, which is the minimum downsampling that max-pooling can perform.

The figure from the original paper is similar. Our plot was produced using the CIFAR-10 dataset instead of the ImageNet validation set, which is most likely responsible for the differences here. The ImageNet images are much larger, with an average resolution of approximately 475x400 pixels (depending on the year of the dataset in question) whereas the CIFAR images are only 32x32. We believe that our results substantially validate the claim of the original paper that spectral pooling preserves information in the original image files well.

5.1.2 Spectral Pooling and Frequency Dropout for Traditionally Parameterized CNNs

The best classification error rates we obtained for the CIFAR-10 and CIFAR-100 datasets were 18.76% and 51.83%, respectively. These were obtained with the following hyperparameters:

M (number of spectral pooling layers): 6 for CIFAR 10, 4 for CIFAR 100 L2 Norm: 3e-4 for CIFAR 10, 1e-4 for CIFAR 100 Gamma (dimensionality reduction at each spectral pooling layer: 0.79 Initial learning rate: 1e-3

Both of these results were improvements on the results of our hyperparameter search; the best results obtained by random search were 27.63% for CIFAR-10 and 66.68% for CIFAR-100, respectively.

The original paper reports results of 8.6% for CIFAR 10 and 31.6% for CIFAR 100, respectively; therefore, our results were significantly worse than the results reported by the paper?s authors.

Optimization Convergence Speed (Results - Section 5.1.3)

Alex - please complete

5.2 Insights Gained

Implementing the proposals of this paper and attempting to replicate their results was an extremely enjoyable and educational project. None of us had a background in signal processing before attempting this project and we are grateful to have had the opportunity to learn about the Fourier transform and its applications in image processing. It was disappointing that we were not able to replicate the impressive results obtained by the authors. However, after thinking about the proposals and their implications, we have certain theories for why our results with these techniques were essentially equivalent to the results obtained with standard CNNs.

When we were first discussing the paper before we began to implement the proposals, our theory for why the spectrally parameterized filters would be easier to optimize than the traditionally parameters filters was the DFT was a mysterious nonlinear operation and that the resulting parameters had ?nicer" derivatives than the standard parameters. As an analogy, even though the tanh function and the logistic sigmoid are nearly identical, the difference in the range of the two functions makes it much easier for stochastic gradient descent to optimize networks using tanh for the nonlinearity than those with employing the sigmoid function. It is now obvious to us that the Fourier transform is a linear transform and the DFT coefficients are linear combinations of the filter weights, so this theory about the derivatives does not seem plausible.

DEPENDING ON ALEX'S RESULTS

Our conjecture is that in order to achieve their superior results on convergence times, the authors used a particular form of initialization for the filter weights which outperformed the standard Glorot initialization that we used. **OR**

Despite this, we did indeed find that the spectrally parameterized networks converge faster in practice. Although we don't have a particularly satisfying explanation for this, we assume, as do the original authors, that this is a result of the of the form of the parameter updates used by the Adam optimizer. As the original authors write, ?[T]his parametrization corresponds to a rotation to a more meaningful axis alignment, where the number of relevant elements has been significantly reduced. Since modern optimizers implement update rules that consist of adaptive element-wise rescaling, they are able to leverage this axis alignment by making large updates to a small number of elements."

Another insight, related to the spectral pooling operation, involves this choice of downsampling operation and its effects on the image content. As the images in both the original paper and ours demonstrate, it is clearly true that we can still easily understand an image after a fairly drastic dimensionality reduction through spectral pooling. However, there is certain important information that is lost by discarding the high-frequency content of an image. For example, edge detection, which is typically one of the first filters learned by a CNN, is essentially equivalent to discarding all of the low-frequency content of an image! As a result, it is not so clear why the spectral pooling operation would be the best way of downsampling the image, particularly as it is transformed through the network. In fact, it is somewhat akin to average-pooling, which has empirically not proved as successful in most CNN architectures as max-pooling.

Finally, it is not clear to us why the frequency dropout technique should have been as successful as a regularization technique as the authors found. The traditional dropout operation has a clear interpretation as a way of forcing the network to act as an ensemble with shared weights and the statistical properties of ensembles are well known to produce superior results. Although there may be a sound theoretical explanation for why frequency dropout is successful at regularizing a CNN, we were not able to come up with one.

6 Conclusion

Spectral pooling is a downsampling technique that can gradually reduce the dimensionality of an image while maintaining a close approximation to the original image. However, we were not able to replicate the authors? finding that spectral pooling and the related technique of frequency dropout were able to sufficiently regularize the network on their own in order to achieve classification rates on the CIFAR-10 and CIFAR-100 that were substantially superior to previously reported techniques. Finally, **we did not find that that the Adam optimizer converged in substantially fewer epochs when optimizing a network parameterized by the spectral coefficients of the convolutional filters as opposed to the filter weights themselves, regardless of network architecture - THIS MAY NEED TO BE CHANGED BASED ON ALEX'S RESULTS**

7 Acknowledgements

We are grateful to Stanford Professor Brad Osgood and to Stanford University for the outstanding series of lecture videos, The Fourier Transform and its Applications, which are available on YouTube. We are also grateful to Google and the TensorFlow authors and contributors, who have developed and open-sourced a library that makes it possible to implement these complex proposals in an extraordinarily efficient way that correctly and automatically implements differentiation and backpropagation for us while allowing us to take advantage of GPUs. Finally, we would like to thank Professor Kostic for making this fascinating paper available one of the options for our final project!

References

[1] REPLACE THIS OF COURSE Almanie, Tahani, Rsha Mirza, and Elizabeth Lor. "Crime Prediction Based On Crime Types And Using Spatial And Temporal Criminal Hotspots." arXiv preprint arXiv:1508.02050 (2015).

8 Appendix

8.1 Individual Student Contributions