Spectral Representations for Convolutional Neural Networks

Vijay Kalmath(vsk2123), Arnav Saxena(as6456), Ayush Sinha(as6430)

***Abstract*—This project is an attempt at implementing the proposals made by Ripple et all [1] and achieving similar results. These proposals are aimed at improving the performance of Convolutional Neural Networks by exploiting the spectral domain of images (Fast Fourier Transforms are used to quicky convert images from spatial to spectral domain). Essentially the proposals can be summarized into two main ones. The first being, Spectral pooling where the pooling operation happens in the spectral domain of the input images. The second one being spectral parameterization where the image filters are initialized in the spectral domain; The corresponding parameters for this filter in the spectral domain are learnt instead.**
We were able to implement the proposal of the authors. However, we were able to partially achieve the results discussed by the authors due to shortage of compute time and GPU resources.

I. Introduction

Convolution Neural Networks (CNNs) have become the predominant method to address image classification tasks in recent days. The proposal and implementation of LeNet by Yann LeCun et al., showcasing CNNs beating the state of the art in 1998 [2] and the AlexNet winning the ImageNet competition by utilizing GPUs in 2012 [3] were pivotal in making CNNs as popular as they are today.
CNNs draw inspiration from how our brain processes images. They are similar to feedforward networks with the fundamental difference, that they assume the inputs are images [4]. CNN architectures characteristically have Convolution layers that convolve a learnable filter with the input images and Pooling layer which is a down sampling operation along the axis dimensions of the input.
CNNs are known to be computationally expensive to train. For faster Convolutions, the use of Discrete Fourier Transforms (DFT) has been shown to be integral [5]. This speedup arises from the property of operator duality between convolution in the spatial domain and element-wise multiplication in the frequency domain. Oren Rippel et al. in their paper acknowledge the usefulness of DFT and introduce two more applications of spectral representations of images. Firstly, they propose learning filter parameters directly in the Frequency domain, as they claim that defining the filter parameters of a CNN in the frequency domain optimizes the convolution operation since the spectral representation of filters is significantly sparser than their spatial counterpart. It is to be noted that this alternate parameterization technique doesn't impact the underlying CNN model in the way it works. Later on, the authors provide experimental results on how this technique helps the model reach convergence in less than 20-50% of the total number of epochs as compared to traditional CNNs. Secondly, they introduce spectral pooling which is essentially the truncation of the spectral representation of the image. They claim that spectral pooling saves more information for the same number of parameters compared to other pooling strategies. This is supported by the fact that higher frequencies in spectral representation of images tend to encode more noise [6] and removing them during spectral pooling leads to minimal loss of information. Additionally spectral pooling allows us to down sample the input to any arbitrary shape which is utilized by authors to perform regularization by stochastically changing the frequencies that are truncated. The goal of this project was to implement spectral pooling and parametrization and reproduce the results obtained in the paper. The goals of this project were:

1) Implement spectral Pooling Layer
2) Implement Spectral parametrization Convolution Layer
3) Use these layers on different CNN architectures to compare results

The technical challenges faced by the team included, understanding the concepts in 2D Fourier Transform to handle the edge cases presented in spectral pooling, successfully representing kernels in frequency domain while doing spectral parametrization and replicating the CNN architectures proposed in the paper for comparing results. The challenges were resolved by referring to the supplemental material offered along with the paper, understanding the underlying concepts duly by referring various resources available online and trial and error.

II. Summary of the Original Paper

A. Methodology of the Original Paper

The authors divide their work broadly into two parts; Spectral Pooling and Spectral Parametrization.

Spectral Pooling: The DFT of the input image $X \in \mathbb{R}^{M \times N}$ is computed assuming that the DC component is shifted to the center. The frequency representation of the image is cropped leaving only the desired $H \times W$ submatrix deonted by $y = \mathcal{F}(X) \in \mathbb{C}^{M \times N}$. This is converted back to spatial domain by taking the inverse DFT.

Spectral Parametrization: The filters used in the CNN are parametrized in the frequency domain itself. That is to learn a filter's parameters the filter is converted from spatial to

spectral domain by taking the filter's DFT. The filter is then converted back to its spatial representation taking the inverse DFT respectively.

The backpropagation through inverse DFT is done keeping in mind the conjugate symmetry constraints of DFT.

B. Key Results of the Original Paper

The following are the 3 keys results of the paper:

1) Information preservation by spectral pooling: Information retainment of images after pooling in spectral domain is measured by taking the l2 loss between the image and the pooled image. Significantly better reconstruction of the images are achieved for spectral pooling at same number of parameters compared to max pooling. Additionally max pooling limits the number of parameters (or dimensions) that can be retained in the pooled image (due to strides), whereas from spectral pooling we can get an output of any dimension. Both these observation are represented in the figure 1.

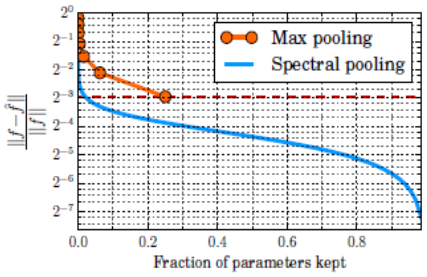

Fig. 1. Approximation Loss for ImageNet validation set

2) Spectral pooling is utilized with a CNN architecture to achieve classification rates of 8.6% on CIFAR-10 and 31.6% on CIFAR-100.
3) Spectral parametrization is applied on 3 different CNN architectures and the speedup in convergence measured. The speed up is calculated by using the number of epochs taken by the spectrally parametrized model to attain the same error rate as conventional CNN model

III. Methodology

The work in this project has been divided into two main components:

1) Spectral Filter Parameterization
2) Spectral Pooling

Both these components have been implemented as close to what has been described by the authors in the paper. The architectures developed are identical to what the authors have used to produce their results. We describe the methodology followed to implement these two components in the following subsections.

A. Spectral Filter Parameterization

In traditional CNN all the learnable parameters (weights of the filter as well as the associated bias terms) are defined in the real (spatial) domain. In this implementation however, the aim was to design a CNN with spectrally parameterized filters. We did this by initializing our parameter weights as complex-valued coefficients of the DFT of the filter weights rather than the filter weights themselves.

To do this we first created a custom "spectralConv2D" layer in Keras. Within this layer we built two separate kernels for representing the real and imaginary parts of our complex-valued kernel. This was done since Keras doesn't provide support for learning complex parameters directly. Next, we combine these two kernels into a single complex filter using tf.complex. We then take the inverse DFT of this filter so as to get its spatial representation which is ultimately used in the convolution operation with the input tensor (which was already in the spatial domain). Note that this convolution operation is identical to how it works in traditional CNN. Figure 2 demonstrates the steps as a flowchart.

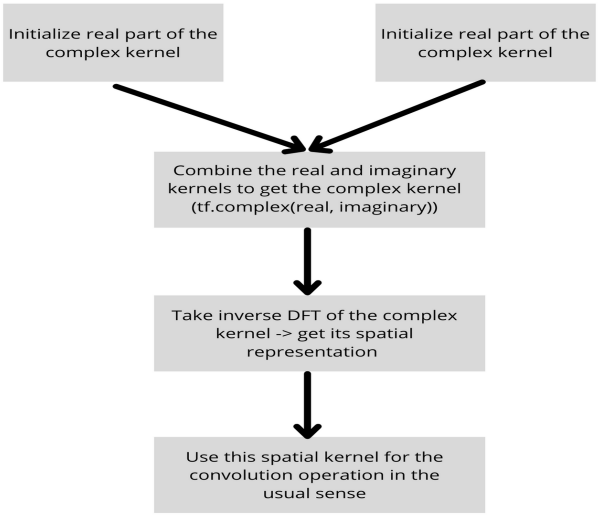

Fig. 2. Process for Spectral Filter Parameterization

B. Spectral Pooling

We have closely followed the Algorithm 1 for implementing spectral pooling. We define a class called Spectral Pool. This is used to help implement a custom "spectral pool layer" in Keras which takes as input an RGB image and the dimension to reduce this image to. For an input batch of RGB images we first convert the tensor to a "channels first" tensor. For each of the channels in the image we get its Fourier transform. While taking the Fourier transform of the image, we make sure to shift the dc component of image to the center. After

the image is in spectral domain, we reduce its dimensions by only filtering the low frequency values. The resulting image is also treated for corner cases according to algorithm 2. Finally, the inverse Fourier transform of the image is taken to get the down sampled image in the spatial domain. Figure 3 describes the same process.
Additionally as the authors have shown that stochastically choosing what frequencies to keep and what to terminate for an image in the spectral domain leads to a regularizing effect, another method called frequency dropout is implement which samples the filter size to be used in spectral pooling from a uniform distribution. The Algorithm3 explains it.

Algorithm 1 Spectral Pooling

Input: Image $X \in \mathbb{R}^{A \times B}$**, output size:** $C \times D$
Output: $\hat{X} \in \mathbb{R}^{C \times D}$

$y = \mathcal{F}(X)$ ▷ Taking Fourier Transform

if B is odd **then**
  Increase size of B by 1 to make it even
else if B is even **then**
  $top = \lfloor (A - C)/2 \rfloor$
  $bottom = top + C$
  $left = \lfloor (B - D)/2 \rfloor$
  $right = left + D$
end if

$\hat{y} = y[top : bottom, left : right]$
$y = TreatCornerCases(\hat{y})$
$\hat{x} = \mathcal{F}^{-1}(\hat{y})$

Algorithm 2 TreatCornerCases

Input: Image $Y \in \mathbb{C}^{A \times B}$
Output: $Image\hat{Y}$ **obeying conjugate symmetry**
$\hat{Y} = y$
$S = [0, 0]$ ▷ S contains indices whose value need to be made real
if A is even **then**
  S.Append(A/2,0)
else if B is even **then**
  S.Append(0,B/2)
else if A is even and B is even **then**
  S.Append(A/2,B/2)
end if
for $i \in S$ **do**
  $Imaginary\_part(S) = 0$
end for

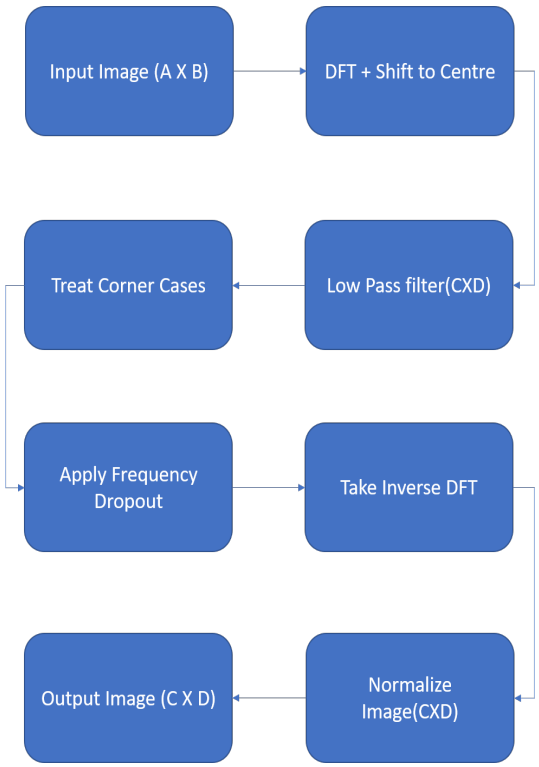

Fig. 3. Process for Spectral Pooling

IV. Implementation

The spectral pool and convolution layers were implemented using Tensorflow2 and NumPy libraries. This section first discusses the data that was used to test these layers. Afterwards we give an overview of the CNN architectures that were employed for performance experiments of the layers.

A. Data

The data sets that were used were CIFAR-10 dataset and CIFAR-100 dataset. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. [7] The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a ”fine” label (the class to which it belongs) and a ”coarse” label (the superclass to which it belongs) [7]. These two datasets were augmented two simulate real world conditions. Augmentations were done in the form of translations, horizontal reflections, hsv perturbations and dropout.

B. Deep Learning Network

There are 3 architectures being employed by the authors. We have utilized two to report our results. The first being the generic one used by many:

$C^{96}_{3\times3} \rightarrow MP^{2}_{3\times3} \rightarrow C^{192}_{3\times3} \rightarrow MP^{2}_{3\times3} \rightarrow FC^{1024} \rightarrow SoftMax$

The second being a deep one claimed by authors to be competitive:

$C^{96}_{3\times3} \rightarrow C^{96}_{3\times3} \rightarrow MP^{2}_{3\times3} \rightarrow C^{192}_{3\times3} \rightarrow C^{192}_{3\times3} \rightarrow C^{192}_{3\times3} \rightarrow MP^{2}_{3\times3} \rightarrow C^{192}_{1\times1} \rightarrow C^{10/100}_{1\times1} \rightarrow GA \rightarrow SoftMax$

Here C^{F}_{S} denotes Convolution layer with F filter each of size S. GA denotes global averaging layer. MP denotes max pooling layer.

V. Results

We attempted to reproduce the three implementations and their and results achieved in the original paper.The error rates that they achieved involved high epoch training of the model and multiple iterations of Bayesian hyperparameterization. While we did implement hyperparameterization using Keras Tuner we could not run it for a huge number of max trials and epochs per trial due to lack of computational resources.Nevertheless for the majority of the

A. Information preservation

The authors of the original paper claim that spectral pooling is much better at preserving information than other traditional pooling methods with similar amount of dimension reduction.

For this purpose, we implemented the spectral pooling and created the images subplots in the fashion similar to what was reported by the authors in their paper in Figure 2.

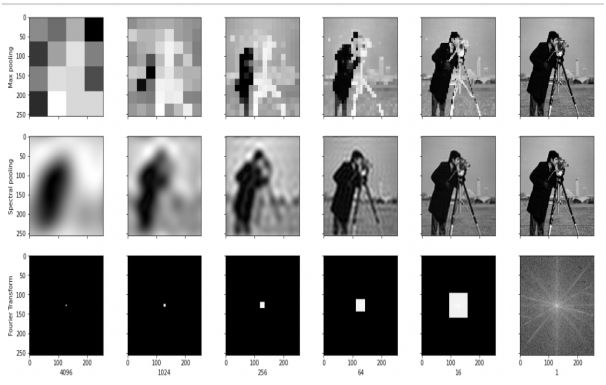

Fig. 4. Higher Information preservation in Spectral Pooling than Max Pooling

Here we take a gray scale 256x256 image and we compare the outputs of multiple stages of max pooling and spectral pooling. The amount of pooling done increasing exponentially from the right to left.The third row of images indicates the square within which the data was kept and the rest removed. We can clearly see that the spectral pooling is much better at keeping the overall structure of the image intact through the extreme amounts of pooling.

We can see that in the most extreme case of dimension reduction(by a factor of 4096 , the max pooling output has lost all resemblances to the original image but in the spectral pooling with just 8 frequencies , we can still see the silhouette of the man taking the photo.

Furthermore we worked on performing similar check on an RGB image , we see that spectral pooling outperforms Max pooling with multiple channels as well.

Fig. 5. Higher Information preservation in Spectral Pooling than Max Pooling in RGB images

While we can comfortably say that we spectral pooling performs better than max pooling , we further tried to find mathematical evidence to show that spectral pooling performs better than Max pooling .

Using L2 Norm we were able mathematically prove that more information is preserved in spectral pooling than max pooling. In Figure 6, the X-axis shows the amount of parameters kept in the image after the corresponding pooling application and the Y-Axis is the relative loss of information between the original image and the processed image.

We used CIFAR-10 dataset wherein we subset 1000 images from the training dataset as validation and calculated the approximation loss for all the 1000 images.

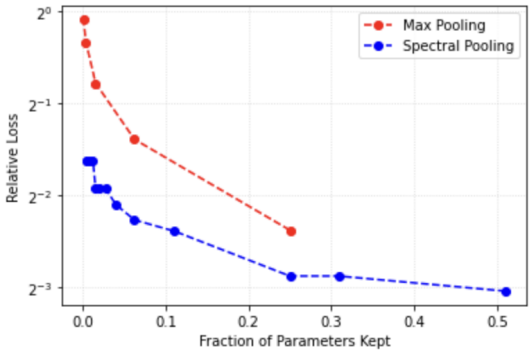

Fig. 6. Relative loss of the two Pooling methodologies with respect to amount of original information preserved.

As the images we used were a lot smaller than the Imagenet validation dataset used by the authors, the x-axis had to limited to 0.5 in order to keep the graph easy to read and infer from.

B. Spectral Pooling and frequency dropout

With the spectral pooling layer confirmed to be working as expected , we implemented a CNN architecture specified in section 5.1 of the original paper.
The architecture uses M pairs of layers of Convolutional Layers followed by Spectral Pooling frequency dropout layers whose filter size is a function of the output of the convolutional layer. The last part of the model were 1x1 CNNs with filters equal to the number of output classes followed by Global Averaging and Soft max layers.

The classification error rates we obtained for the CIFAR10 and CIFAR-100 datasets with just spectral pooling and No other form of regularization was 21% test error rate and a very low value for CIFAR100 , this value does not indicate the capability of the layer as we were unable to train it completely over multiple epochs without having the VM stall or the GPU run out of memory. With a better GPU and a more powerful VM , we believe we should be able to achieve the results achieved by the team for CIFAR100 as well.
In order to replicate the work done by the authors , we performed Bayesian hyperparameter search over the gamma value , the frequency dropout alpha and beta values and the number of convolutional pairs variable.

Once again due to the highly computation intensive operations involved in hyperparameter search , we were unable to run it for the required number of epochs and the required number of trials , therefore we were only able to achieve a validation accuracy of 82% after the Bayesian hyperparamater search .

C. Optimization Convergence Speed

In the original paper, the authors claimed that spectral parameterization consistently improved the number of epochs it took to reach convergence by a factor of 2.2 to 5.1, depending on the architecture. Due to time constraints we couldn't run our experiments to absolute convergence and hence can't reproduce the results in terms of speed-up factors, but we can clearly notice the optimization benefits of using spectral parameterization in our experiments. In figure X, we have plotted the training errors across epochs for the following two CNNs: a) traditional spatial CNN and b) CNN with spectral filters. We test these CNNs across four different settings:

1) Deep architecture with 3x3 sized convolution filters
2) Deep architecture with 5x5 sized convolution filters
3) Generic architecture with 3x3 sized convolution filters
4) Generic architecture with 5x5 sized convolution filters

As suggested in the original paper all the models were supplemented with considerable data augmentation in the form of translations, horizontal reflections, and HSV perturbations. The first thing we observe is that across all the 4 settings, the traditional spatial CNN model struggled to converge in the first 50 epochs whereas the spectrally parameterized CNN pretty much converges within the first 50 epochs. Note that both the spatial and the spectral CNNs were trained in identical experimental settings and hence the difference in convergence rates is pretty drastic. This difference is even more prominent when looking at the deep model where our spectral CNN converged within 10 and 20 epochs for 3x3 and 5x5 sized filters respectively. The original paper quoted pretty much the same results as well.

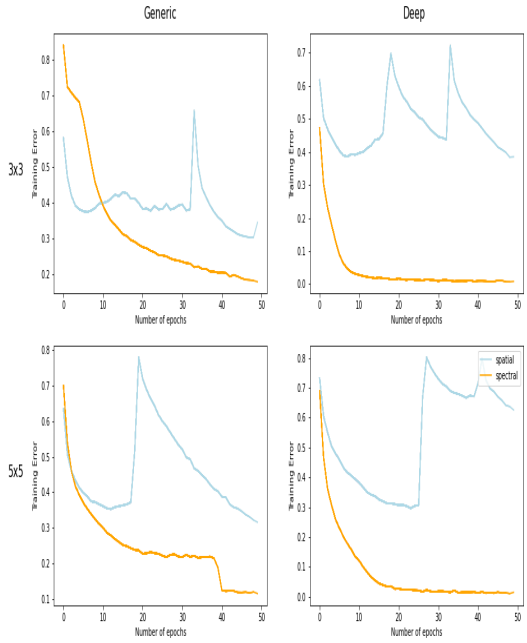

Fig. 7. Training rates for various experiments with y axis denoting error rate

VI. Future Work

In the future work the original paper discusses the opportunities of building an end-to-end component wherein the convolution also takes place in the spectral space. We were also able to take a step forward in this direction and have implemented a custom spectral convolution layer in Keras. To achieve this, we accept our kernels and input in the usual spatial domain and convert them both in the frequency domain by taking their FFT. Next, we conduct element wise multiplication on them instead of usual convolution operation and ultimately convert the result we get from this multiplication back to spatial form using inverse FFT. Flowchart 8 articulates this process easily.

Due to time constraints we had to skip certain experiments in this work and hence as our first priority we would like to conduct those experiments and expand upon the insights provided in this work. Lastly, this project helped us have all the components necessary to build an end-to-end convolution network in the spectral domain. Hence that seems to be a

natural extension of our work in the future.

Furthermore , since the paper requires complex weights being initialized and there is considerable future work present, we can try to revive the feature request Feature request: complex support in initializers with tensorflow and try to get complex initializers be part of the standard tensorflow package.

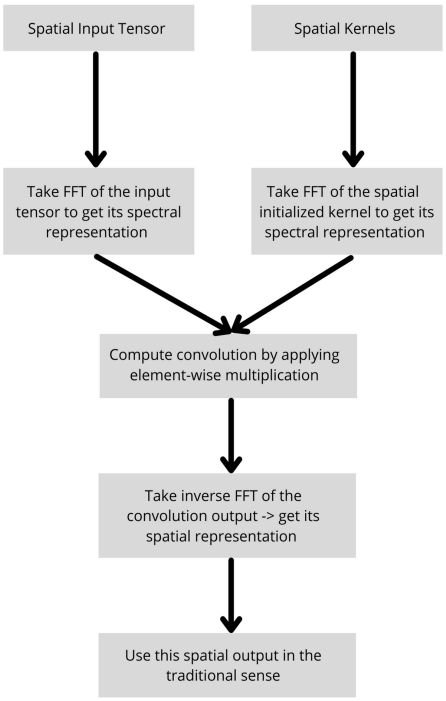

Fig. 8. Modification to Spectral Convolutions

VII. Conclusion

In this project we implemented Spectral pooling layers, Spectral Parameterization layers and Frequency dropout for Spectral pooling. In Spectral parametrization

Moreover to aid the function of these proposals we also implemented modules for data augmentation and Bayesian hyper parameter search.
Toward the end we also implemented spectral convolution wherein the convolution operation takes place in the frequency domain itself.
This way we have all the components necessary to build the end to end spectral CNN architecture as was proposed by the authors of the original papers. We expect to work on it in our future endeavors.

Acknowledgment

We would extend our gratitude to Professor Shree K. Nayar for the insightful image processing videos put out by him online. We are grateful to all the contributors of TensorFlow and Keras Documentation. We extend our thanks to the TAs and Prof Kostic of ECBME4040 course for helping us understand the prerequisites to implementing this paper.

References

[1] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," 2015.
[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[4] M. LLC. (2021) Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition cs231n. [Online]. Available: https://cs231n.github.io/
[5] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards ai," 2007.
[6] A. Torralba and A. Oliva, "Statistics of natural images categories," *Network (Bristol, England)*, vol. 14, pp. 391–412, 09 2003.
[7] V. N. Alex Krizhevsky and G. Hinton. Cifar-10 and cifar-100. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

VIII. Individual Student Contributions in Fractions

	vsk2123	as6456	as6430
Last Name	Kalmath	Saxena	Sinha
Fraction of contribution	1/3	1/3	1/3
What I did 1	Implemented Spectral Pooling	Implemented spectral convolution	Designed and aided in spectral pooling
What I did 2	Implemented Spectral Convolution	Designed deep learning architectures as well as data augmentation code for experiments	Implemented Bayesian Hyper Parameter Search
What I did 3	Implemented Frequency Dropout	Conducted experiments on optimization convergence and spectral vs spatial convolution	Researched Spectral Convolution and frequency dropout

Every participant contributed to the write up.