Spectral Representations for Convolutional Neural Networks

E4040.2021Fall.CSGO.report
Hanshan Li hl3515, Yuning Zhou yz3922, Chongzhi Xu cx2273
Columbia University

Abstract

This project is our attempt to realize the ideas and reproduce the results from Rippel, Snoek and Adams' work in Spectral Representations For Convolutional Neural Network [1] that was published in 2015 at arXiv.org. The goal of our project is implementing Discrete Fourier Transform (DFT) in CNNs to improve the computation speed of convolutions in deep learning by performing DFT in the spectral representation, including spectral pooling and spectral filter parameterization, where the CNNs are under innovative design. This design is beneficial from reducing dimension by performing a truncation in frequency domain representation in spectral pooling which preserves more information per parameter, as well as facilitating the optimization through spectral filtering which assists to reduce training epochs. But this algorithm can also face challenges of accurately implementing the ideas of the paper, such as various Fourier transforms, to the TensorFlow with tensorflow keras module. Although we successfully implemented about 70% of the paper's novel proposals, we finally replicate a fair number of results with little or even zero improvement compared to the original authors' report, but the training accuracy of cifar-10 dataset is 3% higher than the original paper.

1. Introduction

CNNs, also known as Convolutional Neural Networks [2], have won much attention and reputation because they brought revolution to the computer vision field nearly a decade ago and promptly dominated the world of CV since the introduction of AlexNet [3] in 2012. As the name indicates, CNNs are neural networks with deep forward features due to the fact that they are composed of multiple successive layers including convolution layer, pooling layer, fully connected layer and receptive field. Firstly, the CNNs algorithm uses a bunch of learned filters (convolutional kernel) on the input image to compute convolution (in fact the correlation), then the nonlinear activation function (for example, Relu) is applied to the result of convolution, and ultimately perform pooling to make reduction of dimensionality before implementing next convolution. CNNs' main function in science and industry is processing images with large scale and classifying videos, which have been successfully applied for years. But one major challenge still exists - how to decrease the computation expenses to train the network. Particularly, effectively applying convolutional kernels could be the significant part of successfully implementing CNNs on large scale problems [3].

Discrete Fourier Transforms (DFT), as *Rippel, Snoek and Adams*as asserted, provide not only a significant speedup in the computation of convolutions in deep CNNs, but also elevates the effect of modeling and training in three different methods [4]. To be specific, DFT offers the map between convolution in the spatial domain and multiplication in the spectral domain. Therefore, the first plan is described as substituting the real numbers in convolutional kernels with complex numbers in the frequency domain. The second plan was proposed as using spectral pooling layers instead of max pooling layers, which reduces the dimensionality with much more information preservation when enduring the identical degree of parameter reduction. The last plan is a bold attempt, because it implements a revised edition of spectral pooling layer to perform regularization like dropout. Thus, the key thought of applying Fourier Transforms is that natural images typically follow an inverse power law, i.e., more information is stored in smaller frequencies [1]. Hence, our group came up with the idea of leveraging this property by abandoning the higher-frequency image components with significantly less information loss while processing. We will make comparisons and prove that this idea could overcome a crucial disadvantage of max pooling discussed by Krizhevsky et.al. [4] in this project.

Our major goals of this project were separated into two parts: first, triumphantly realizing the proposals of the referenced paper with TensorFlow tools and modeling a reasonable CNN to employ these proposals; second, reproducing the authors' elevated training results to the maximum extent. But there were numerous challenges and difficulties arising from the course of our work while embodying these original proposals. The first and even tortured challenge is related to concepts and terminology: since we tried not to misunderstand the conceptual description in the paper, we spent more than a week looking for all kinds of relevant materials, including but not limited to papers, journals, videos, and books. But we found only a few limited relevant materials. We have to start from the paper itself, searching and recording terms like spatial/frequency domains for both filters and images, or the function and display of Fourier Transforms. Second, the problems occurred when creating and implementing the spectral layers to perform spectral parameterization. Despite the operations being straightforward in conception, correct implementation of them requires much patience and concentration on details. The third difficulty we encountered was related to customizing the concise Keras model and Keras layers, because it's intractable to structure an intricate idea with a concise tool.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

It is proposed in the original paper that the parameterization with a spectral filter can be a substitution for the traditional methodology, for a CNN model to produce its weight matrix. The difference between the two types of filters is that the spectral filter is the DFT of the traditional one, which means the spectral filter has a mode complex expression in its coefficients. This filter has a compatible capacity with that of a traditional method, while maintaining an edge over its competitors on optimization in terms of the number of epochs it takes to converge.

Spectral pooling is another improvement proposed in the original paper. It is a substitution for the max pooling and performs better in reducing the image dimensionality. The advantage of this pooling technique is that it preserves more information than traditional pooling methods without producing coarse blocks in the transformed images. This result is presented in a quantitative format by comparing the loss of the two methods (spectral pooling and max pooling).

The overall performance of this CNN architecture is carried out on the CIFAR-10 and CIFAR-100 dataset, where the model achieved an impressive performance in terms of accuracy.

2.2 Key Results of the Original Paper

The key results for experiments are demonstrated in spectral pooling and spectral parameterization.

The results for spectral pooling are presented in 2 aspects: a better performance in information preservation, and a competitive advantage in classification with CNN models. Spectral pooling turns out to be a more advantageous approach in information preservation due to its allowance for any output dimensionality. This has overcome the flaw in the traditional method, (for example, max pooling), where the stride controls the granularity, as well as providing a superior solution in reconstruction for the same number of parameters. The results for this step are declared in a plot illustrating that this innovation can introduce a lower loss.

The classification of CNN models has achieved the error rates of 8.6% on CIFAR-10 and 31.6% on CIFAR-100 datasets, with the optimal hyperparameter configuration where $\gamma = 0.85$, $\alpha \approx 0.30$, $\beta \approx 0.15$, momentum of about 0.95 and initial learning rate 0.0088.

The result in spectral parameterization is the enhancement of the ability for convergence. It takes shorter time and fewer epochs for spectral parameterized CNN than the traditionally parameterized CNN to converge to the same error rate. This comparison is presented through a combination of plots and tables, where spectral parameterization is compared against the other two methods, namely deep and generic parameterization. All methods presented in this stage utilized Adam for parameter updates, while the speed-up factors are calculated to quantify the superiority of the spectral parameterization approach.

3. Methodology (of the Students' Project)

3.1. Objectives and Technical Challenges

The ultimate goal of this project is to deal with the primary challenge of CNN: reducing the computational expense of training, like that is mentioned in the original paper. This project attempts to improve the efficiency of CNNs by adopting discrete Fourier transform for faster convolution.

In our project, the objectives include conquering some technical challenges, spectral pooling, and spectral parameterization and making spectral convolution layers, which are the primary innovations introduced to CNN compared to traditional modeling methods. More detailed demonstration for these methods' applications of spectral representations is presented below.

3.1.1 Spectral parametrization

CNN parameterization, in a traditional way, is usually presented by transforming a 3D tensor with form like (C_{input}, H_{input} , W_{input}) to another 3D tensor in the form of (C_{output}, H_{output} , W_{output}) using the technique called cross-correlation computation which calculates the 3D volume of every x dimension and y dimension of the input tensor derived with C output number of filters. In this scenario, C represents the "channel", the number of the channels of the input tensor, while H represents the height and W represents the width of the tensor, which is a traditional representation for a tensor. As we know, a set of real numbers that composes the weights of the filter and a by side bias parameter, this set of real numbers constitutes the learner number in each one of C filters that involves with the transmission mentioned above.

In the meanwhile, if we want to implement CNN with spectral parameters, we should split the convolution (correlation) operations out of the whole process of model building. Although the convolution operation is the same as traditional CNN, the learned parameters including kernels, bias are complex numbers/coefficients of the filter under Discrete Fourier Transform, not the filter weights themselves. If the filter' spectral parameters are composed by an input tensor and a set of complex coefficients, we should use inverse DFT on the filter first. And then performing convolutions using new filters (after inverse DFT) with input tensor as what we usually do.

3.1.2 Spectral pooling

In neural networks, minimizing the shape of input tensor, concentrating the original information, the pooling layers are of significant importance. The idea behind spectral pooling stems from the observation that the frequency domain provides an ideal basis for inputs with spatial structure [1]. The key difference between spectral pooling and traditional pooling also lies in the pooling operation domain.

All traditional pooling methods used in neural networks operate the input tensor in the spatial domain, without exception. Take max pooling for instance, simple as it is, max pooling is one of the most widely implemented convolution processes. Its kernel extracts the maximum value from the spatial area composed by height and weight. In other words, only the maximums in certain spatial areas corresponding to kernels are "allowed" to pass through the further layers in the neural network and remain.

On the contrary, spectral pooling found a brand-new way out by truncating the frequency domain "height" and "width". It then applies inverse Discrete Fourier Transform on the frequency-truncated matrix back to the spatial domain. By doing this, the size of the original matrix is compressed in the spatial domain, the main meaningful, useful information is also conserved and condensed in a more smooth and more effective manner compared with traditional pooling, which is vividly demonstrated in the comparison of image compression between traditional pooling and spectral pooling in 5.1 and 5.2 section.

Suppose in the neural network, an input tensor has a shape of (batch size, height, width, channels), the explicit steps of spectral pooling is as follows.

* For every image from the batch size, for every specific channel from channels e.g., RGB, implementing discrete Fourier transform on the other two dimensions height and width.

* According to the frequency threshold for truncation, truncate every element below frequency low bound or above frequency high bound, maintain complex symmetry in the meantime.

* Applying the inverse Fourier transform on the truncated matrix in the frequency domain so that it's converted back to the spatial domain for each given channel and image.

Here for the sake of simplicity and comprehensibility, we also addressed a flowchart shown in Fig. 2 to better demonstrate the idea.

It is worth pointing out that we should pay careful attention to the complex symmetry of the image in the frequency domain, which ensures the success of inverse Fourier transform from truncated frequency matrix to a real-valued spatial image, otherwise the result would be incomprehensible to humans.

As what is mentioned in the original paper named Spectral Representations for Convolution Neural Networks that we try reproducing, the main secret behind this is the symmetry law in the discrete Fourier transform.

$$F(x)_{hw} = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{mn} e^{-2\pi i \left(\frac{mh}{M} + \frac{nw}{N}\right)}, \quad \forall h \in \{0, \ldots, M-1\}, \forall w \in \{0, \ldots, N-1\} \tag{1}$$

For a 2-dimensional discrete Fourier transform process as equation (1) depicts, it's obvious that the upper left corner of outcome matrix $F(x)_{00}$ is always DC current, a real number since it's the Fourier function value at h=w=0. Take even M and N for example, special case could also be found in $F(X)_{\frac{N}{2},0}$, $F(X)_{0,\frac{N}{2}}$ and $F(X)_{\frac{N}{2},\frac{N}{2}}$, area notated by these four points includes parameters unconstrained by the law of conjugate symmetry.

More generally, for any transform $F(x)$ of some input matrix x, the outcome should hold:

$$F(X)_{mn} = F(X)^* \, (M-m) mod M, (N-n) mod N, \quad \forall m \in \{0, \ldots, M-1\}, \forall n \in \{0, \ldots, N-1\} \tag{2}$$

According to conjugate symmetry, in the frequency domain we know that for a frequency matrix F, F [k, l] is the complex conjugate of F [-k, -l], and F [k, l] should have a real value format in the case it modulo the dimension of F. Therefore, whenever one truncates a frequency matrix, he or she should be aware that the special real value case in the "corner" must be shifted toward the center of the matrix at first. and implement truncation after that.

The original paper proposed the very idea while offering little information, leaving the code solution empty, while the previous graduate students Arshay Jain, Jared Samet and Alex Wainger in their previous attempt [6] to replicate the same paper [1], provided a basic implementation on the matrix shifting on TensorFlow version one. On top of that, we realized the matrix shift and frequency dropout by embedding them in the spectral pooling based on the custom TensorFlow Keras Model and custom TensorFlow Keras Layer by subclassing method in a more concise and comprehensible manner (see Subsection 3.2.2).

3.2 Problem Formulation and Design Description

3.2.1 Spectral Parametrization

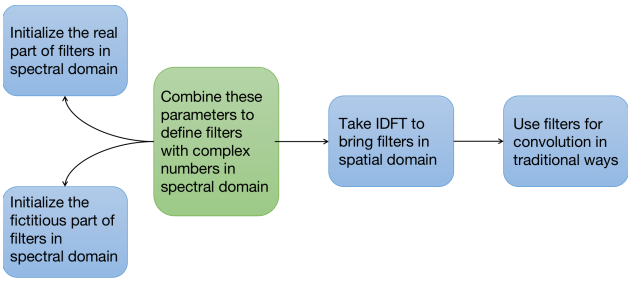

Fig.1. Process for Implementing Spectral Parameterization of Filters

This proposal was implemented in TensorFlow by making a comparison between the Keras model with our specially designed spectral convolution layer (variables are corresponding to the complex DFT coefficients instead of the filter weights) and the traditional CNN model. Vertically, we set up experiments between deep and generic model architecture as well as the kernel size. We combined these factors in order to compare the spatial and spectral performance when dealing with dataset cifar-10. Judging from the convergence time and accuracy, it is easy to conclude which method is more efficient and appropriate when implemented in the CNN with the same structure. To be more specific, there are 5 sets of comparison experiments: 1. CNN model with and without spectral convolution layers in generic CNN architecture (kernel size 3 and 6); 2. CNN model with and without spectral convolution layers in deep CNN architecture (kernel size 3 and 6); 3. CNN model with and without spectral pooling layers (no spectral convolution layers) in generic CNN architecture (kernel size 3). According to the results in section 5, we can safely arrive at the conclusion.

3.2.2 Spectral pooling

Architecture of how the custom spectral pooling layer is used in a custom spectral model is described in Fig.2. The TensorFlow network diagram is shown in Fig3. More details could be found in Section 4.2.2.

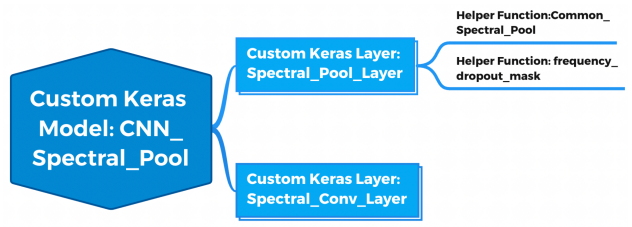

Fig.2. CNN with Spectral Pooling Architecture

4. Implementation

In this section, we introduce how our methodology in Section 3 is implemented in two subdivided sections, Data and Software Design.

4.1 Data

In this project we use CIFAR-10 and CIFAR-100, where CIFAR is short for Canadian Institute For Advanced Research [3].

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, which incorporate airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6000 images per class, while 50000 of the images are for training and 10000 images for testing. The testing set is randomly generated from each class.

The CIFAR-100 dataset is similar to the CIFAR-10 dataset, while the images belong to 100 classes. The size of the classes is 600, and there are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 “superclasses”, and thus each image has a coarser label and a finer label at the same time.

4.2 Software Design

On top of CIFAR-10 and CIFAR-100 dataset, we composed six Jupyter Notebooks i.e. .ipynb files and eight python files, all of the codes can be found on https://github.com/ecbme4040/e4040-2021fall-project-csgo.git. The overall Jupyter Notebook structure is listed in the mind mapping as follows, which is also covered in the README.txt file in the link:

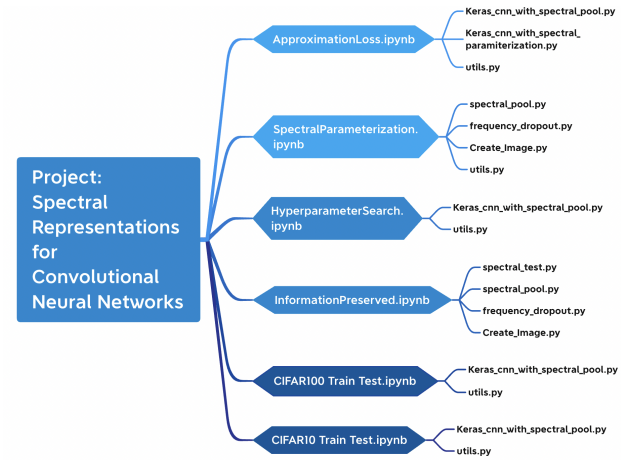

Fig. 3. Top Level Jupyter Notebook structure

4.2.1 Hyperparameter Search Pseudocode

The Python pseudocode of the hyperparameters search is as follows. This part will be discussed more in Section 5.1.2

```
SET parameters search range
Initialize variables to store parameters
FOR search_idx IN range(serch_times):
        SET start_time TO time.time()
        Randomly SELECT learning rate, L2_norm,
M, epsilon IN the range respectively in the range
        RECORD selected hyperparameters
l,L2,M,epsilon respectively
        RUN the custom keras spectral model
        LOG the best hyperparameters ever
LOG the overall best hyperparameters
```

4.2.2 Spectral_Pool_Layer Pseudocode

The Python pseudocode of the custom TensorFlow Keras Layer named Spectral_Pool_Layer corresponding to spectral pooling in the original paper is as follows.

```
# Init:  Initialize the custom layer by giving the number of
output channels, lower bound for frequency dropout and
upper bound for frequency dropout.
DEF Spectral_Pool_Layer(tf.keras.layers.Layer):
   DEF __init__(self, parameters):
      super(Spectral_Pool_Layer,
self).__init__(**kwargs)
      SET parameters TO self
# Call: Implement Fourier transform, Matrix shift,
Frequency Dropout, Inverse Fourier transform as a whole
spectral pooling process. The input tensor is a  four
dimensional matrix with shape (Batchsize, Height, Width,
Channels)
   DEF call(self, INPUT_tensor, activation=tf.nn.relu):
      SET img_fft TO tf.signal.fft2d (tf.cast
(INPUT_tensor, tf.complex64))
      # Image truncated always meet the required correct
complex symmetries so that its inverse Fourier transform
will be real-valued.
      SET img_truncated TO Common_Spectral_Pool
(img_fft, self.kernel_size)
     # Calculate output regarding the frequency drop out
bound
       IF (self.freq_dropout_lower_bound is not None)
and (self.freq_dropout_upper_bound is not None):
          SET tf_random_cutoff
          SET dropout_mask
          SET img_down_sampled TO img_truncated
[:,:,:,:] * dropout_mask
          SET output TO tf.math.real (tf.signal.ifft2d
(img_down_sampled))
      IF activation is not None:
         RETURN activation(output)
      ELSE:
         RETURN output
```

4.2.3 Spectral_Conv_Layer Pseudocode

The Python pseudocode of the custom TensorFlow Keras Layer named Spectral_Conv_Layer corresponding to spectral convolution in the original paper is as follows.

```
DEF Spectral_Conv_Layer (tf.keras.layers.Layer):
   DEF __init__(self, filters, kernel_size=3,
**kwargs):
      super(Spectral_Conv_Layer,
self).__init__(**kwargs)
      SET self.filters TO filters
      SET self.kernel_size TO kernel_size
      SET self.kernel TO None

      DEF build(self, INPUT_shape):
         SET self.sample_weight
         SET self.bias TO

      DEF call(self, INPUT_tensor):
         SET complex_sample_weight TO tf.cast
(self.sample_weight, dtype=tf.complex64)
         SET fft2d_sample_weight TO tf.signal. fft2d
(complex_sample_weight)
         SET real_init TO tf.math.real (fft2d_sample_
weight)
         SET imag_init TO tf.math.imag (fft2d_
sample_weight)
         SET spectral_weight TO tf.complex(real_init,
imag_init)
         SET complex_spatial_weight TO tf.signal.
ifft2d(spectral_weight)
         SET spatial_weight TO tf.math.real
(complex_spatial_weight)
         SET self.kernel TO spatial_weight + self.bias
         EXPAND self.kernel dimension with input
channels and output channels
         RETURN tf.nn.conv2d (INPUT= INPUT
_tensor, filters=self.kernel, strides=[1,1,1,1], padding=
'SAME')
```

4.2.4 CNN_Spectral_Pool Pseudocode

The Python pseudocode of the custom TensorFlow Keras Model named CNN_Spectral_Pool based on two Keras custom layers in Section 4.2.2 and 4.2.3 is as follows.

```
DEF CNN_Spectral_Pool(tf.keras.Model):
      DEF __init__(self, parameters):
           SET Layers to be used in this custom Keras
model TO self

      # Helper function
      DEF Num_of_Filters(self, m):
           RETURN min(self.max_num_filters, 96 +
32 * m)
```

```
    # Helper function
    DEF Freq_Dropout_Bounds(self, size, idx):
        RETURN freq_dropout_lower_bound,
freq_dropout_upper_bound

    DEF call(self, INPUT_tensor, training=True):
        FOR m IN range(self.M):
            IF m EQUALS 0:
                SET x TO
            Spectral_Conv_Layer0(INPUT_tensor)
                ELSE:
                SET x TO
            Spectral_Conv_Layerm(x)
            SET x TO
        self.Spectral_Pool_Layer_list[m](x)
            SET x TO self.flatten(INPUT_tensor)
            SET x TO self.dense1000(x)
            SET x TO self.dense100(x)
            SET x TO self.dense10(x)
            SET x TO self.softmax(x)
        RETURN x
```

5. Results

Results for this paper are presented in this section, following the order from project result replicated by our team to the comparison between the project result and the original paper [1]. The reasons behind the gap in results are elaborated further in the 5.3 section, where analysis is conducted. This section inspires and navigates the approaches for implementing future improvements.

5.1 Project Results

Project results are composed of 4 main parts. The information preservation results generated from the spectral pooling are presented with a series of images under pooling transformation; hyperparameter search result is shown with a combination of numerical search result and loss plots over time; spectral parameterization's impact on convergence is demonstrated with the variation between whether spectral parameterization is implemented; and the dataset's influence is measured through visualization.

5.1.1 Information Preservation Results

Below is the comparison between spectral pooling and max pooling in the information preservation in a grayscale image i.e., single channel compress case. The first line is the result of implementing max pooling on the original gray picture with kernel sizes respectively 64, 32, 16, 8, 4, 1, and the second line shows the result of implementing spectral pooling with cutoff frequencies threshold respectively 4, 8, 16, 32, 64, 128 corresponding to the frequency representation in the third line. By observing the comparison pictures though the two methods are capable for achieving similar compression effects, obviously spectral pooling has an overwhelming advantage compared with max pooling in the extent of information remaining. On the one hand, the variation in the max pooling result is abrupt and uncomfortable to humans, pale by comparison to the smooth, natural spectral pooling series. On the other hand, spectral pooling results are more cognizable, one can easily tell the picture is depicting a handsome boy even with a low frequency cutoff threshold 8, while it's almost impossible to distinguish with the result from a 16*16 kernel max pooling.

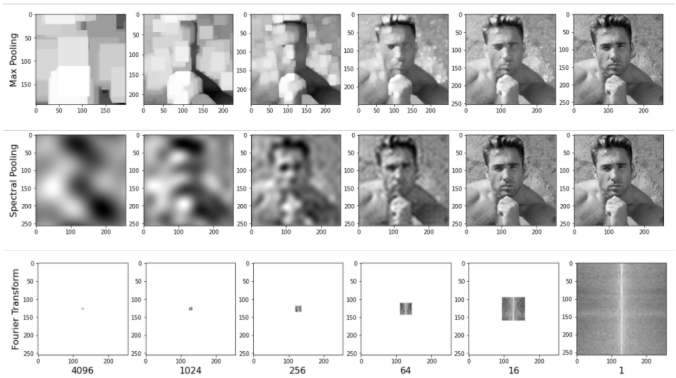

Fig. 4. Information preserved comparison between Spectral Pooling and Max Pooling in Gray image

Also, Fig.5 shows a plot of the average information loss for the CIFAR-10 dataset when implementing max pooling and spectral pooling. L2 error is shown in the vertical axis in terms of the standard value of the input image. The same plot is shown in the original paper. We believe that our results largely replicate the original paper's proposal that the spectral pooling well preserves the information in the original image files while making the input picture simpler to process.

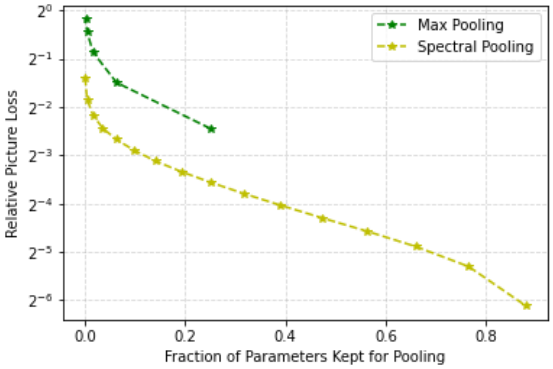

Fig. 5 Relative loss of spectral and max pooling

5.1.2 Hyperparameters Search

During the project experiments we found the performance of the model owes a close relationship with hyperparameters. Considering it's tedious and impossible to manually adjust the hyperparameters to the best, we

devised hyperparameter search so that it automatically gets the theoretically best hyperparameters for the convenience of the other parts of our project. Basically, the hyperparameter search is done on top of pseudocode in the part 4.2.1. And the result of hyperparameter search is shown below, optimal parameters turns out to be:

```
Hyperparameters Search Ended!!!
Best Train Accuracy Ever:  0.9688313603401184
Best Valid Accuracy Ever:  0.50439453125
Best Hyperparameters Ever:  {'learning_rate': 0.0001447809731081944, 'l2_norm':
0.00026955997339473444, 'M': 5, 'epsilon': 2.3074948215823896e-07}
```

Fig. 6. Optimal Hyperparameters through search

To verify the correctness and efficiency of hyperparameter search, we did a thorough comparison the model performance between manually tuned parameters and our optimal parameters in both CIFAR-10 and CIFAR-100:

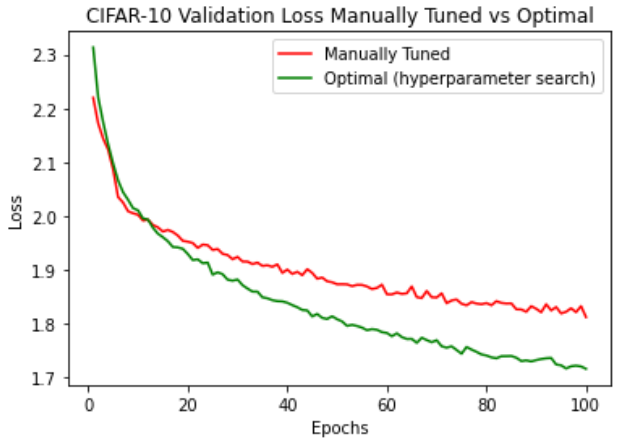

Fig.7. CIFAR-10 Manually Tuned vs Optimal

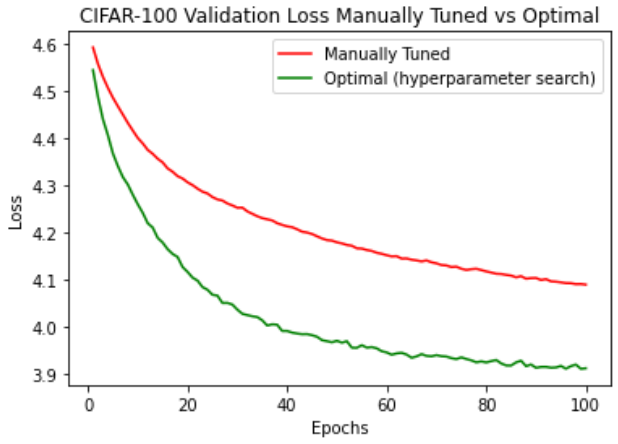

Fig.8. CIFAR-100 Manually Tuned vs Optimal

From the figures above (Fig.7 and Fig.8), it is presented that the loss from both manual-tuned model and parameter-searched model has a downward trend, while the loss decreases faster for the hyperparameter searched model on both CIFAR-10 and CIFAR-100 datasets. It also took fewer epochs for the hyperparameter-searched model to converge.

5.1.3 Spectral Parameterization

Whether spectral parameterization is embedded in the layers also influences the performance of the models. By controlling the variables in the experiment conditions, the behavior of the models with and without employing the spectral parameterization techniques (including spectral convolution and spectral pooling) can be compared, in terms of how their training and validation accuracy change over time and the convergence time each method needed.

Result is listed as pictures and explanations below (Note: green curves represent the scenario using spectral representation, blue curves not using it):

1. In generic CNN architecture, kernel size is 3, using max pooling comparison between using and not using spectral convolution layer, Fig.8 shows the validation accuracy of these two scenarios. We can derive some conclusions: in generic architecture, using a spectral convolution layer brings far better performance than not using it, judging from the convergence time and validation accuracy. When kernel size is smaller, such as 3, the spectral convolution layer is proved to have good "pass-ability", despite the applied complex kernel.

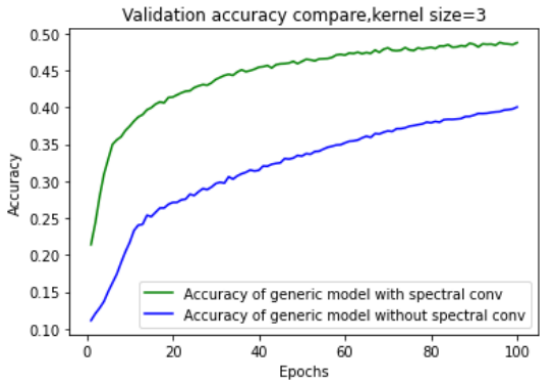

Fig.9. Generic, kernel size = 3, using spectral conv or not

2. In generic CNN architecture, kernel size is 6, using max pooling, comparison between using and not using spectral convolution layer: Fig.9 shows the validation accuracy of these two scenarios. We come to some conclusions: in generic architecture, using a spectral convolution layer brings not that better performance than not using it, judging from the convergence time and validation accuracy. Because the kernel size is bigger, such as 6, and the spectral convolution layer can't have a good "pass-ability", when complex kernels are implemented.

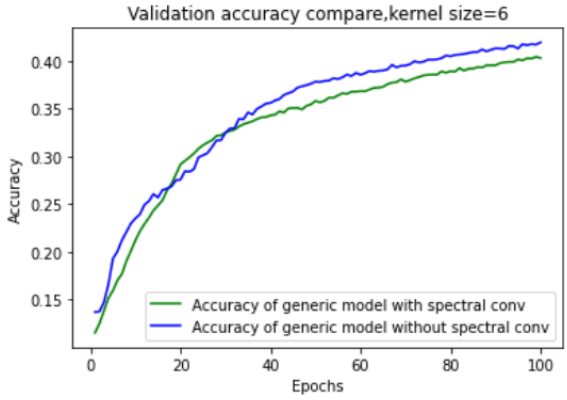

Fig.10. Generic, kernel size = 6, using spectral conv or not

3. In deep CNN architecture, kernel size is 3, using max pooling, comparison between using and not using spectral convolution layer: according to Fig.10, with smaller kernel size, the performance is still better.

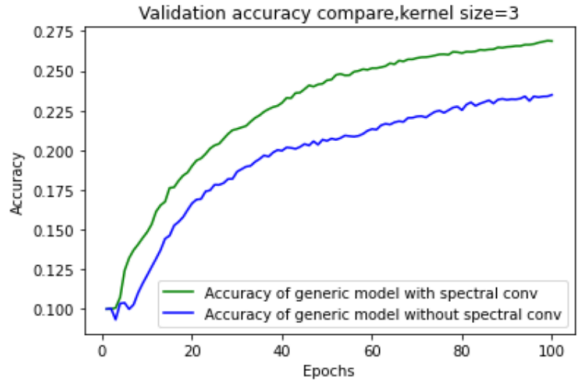

Fig.11. Deep, kernel size = 3, using spectral conv or not

4. Also, in deep CNN architecture, kernel size is 6, using max pooling, comparison between using and not using spectral convolution layer: the Fig.11 shows how superius the spectral representation is compared to traditional convolution layer when implemented in deep CNN network. It gave neural networks faster convergence speed and accuracy, in a 100-epoch training.

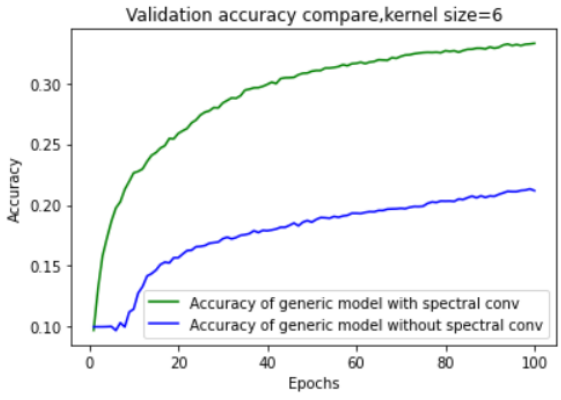

Fig.12. Deep, kernel size = 6, using spectral conv or not

5. In generic CNN architecture, kernel size is 6, using traditional convolution layers, comparison between using and not using spectral pooling layer: when spectral pooling and spectral convolution layers are all implemented in CNN to substitute traditional convolution layer and max pooling layers, we can conclude that the performance is not improved so much because these 2 spectral representations might block the flow of tensors.

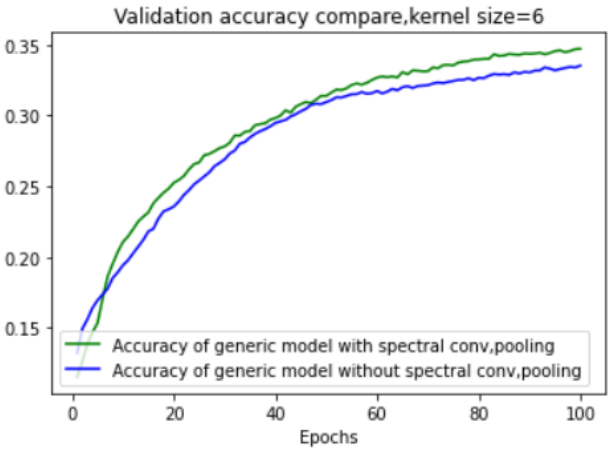

Fig.13 Generic, kernel size = 6, using spectral pooling or not

The plots demonstrating the differences between the model performance are shown above. All five experiments could reveal that the performance of CNN models can be improved in different aspects by implementing spectral parameterization to produce a lower loss, higher accuracy and faster convergence. But be aware that using both spectral pooling and spectral convolution layer will not be appropriate because of its blocking effect on tensors and slow down the convergence time.

Compared to the original paper, we can't estimate their convergence time for an epoch, but we indeed validate the ideas proposed in the paper that the spectral representation can improve the performance of CNNs, give faster convergence and higher training and validation accuracy. In the original paper they use error rate to represent the performance, here we consider the error rate plus training accuracy equals 1, so the training accuracy is higher than the validation accuracy presented in this part.

5.1.4 Datasets

As CIFAR-10 and CIFAR-100 are used in this project, the differences between the two datasets can further lead to a variation in the models' performance and experimental results.

A plot comparing the corresponding results of loss function is as below, where the loss in CIFAR-100 is significantly higher than that of CIFAR-10. This performance indicates that the convergence behavior is faster in CIFAR-10 than

that in CIFAR-100. Similarly, the accuracy of models on the two datasets presents a comparable pattern.

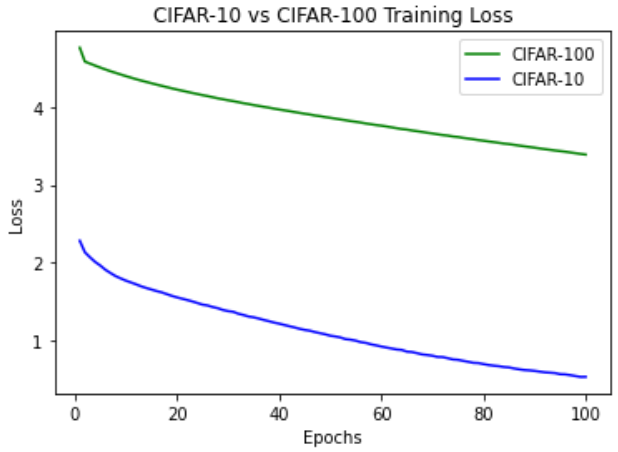

Fig.14. Loss for CIFAR-10 and CIFAR-100

5.2 Comparison of the Results Between the Original Paper and Students' Project

The overall comparison between the original paper's and our project's results are summarized in the table below. The accuracy for both datasets are lower than the products from the original paper, the analysis for this difference mentioned in Table.1 will be further elaborated in detail.

Table.1 Comparison with Original Paper in Training Accuracy

Accuracy	Original Paper	Students' Project
CIFAR-10	91.4%	94.93%
CIFAR-100	68.4%	46.7%

Meanwhile, the test accuracy for our research on the CIFAR-10 and CIFAR-100 datasets are 20% and 49% correspondingly.

5.2.1 Information Preservation Comparison Between the original paper and students' project

To go steps further than the original paper, we also managed to realize the comparison between max pooling and spectral pooling in the case of RGB 3 channels image compression. It further shows that the spectral pooling does well in preserving the information not only the outline but also the color. The uneven distribution of color blocks in the max pooling case makes its color different with the original one, for instance the result from a 64*64 kernel almost purely consists of white. However, though the image already becomes blurry with a 4 frequency drop off threshold, the brown, black, yellow, white colors are mostly preserved.

Fig.15 Information preserved comparison between Spectral Pooling and Max Pooling in RGB image

5.3 Discussion of Insights Gained

The first key reason for the difference from the results of the original paper lies in the limit of computer power and instable GCP connections. Which limits us to use smaller epoch numbers and therefore didn't reach the optimal model performance before epochs ended. When implementing, i.e., more than 200 epochs with batch size 512, it requires more than 20 minutes to see the final training and validation results. If tuning the parameters in a model, it requires another iteration, which is very time-consuming. So, we choose 100 epochs with batch size 512 to get as better results as we can.

The second reason can be concluded as the limit of the dataset. What we use is the CIFAR-10 and CIFAR-100, and the original paper used ImageNet pictures which are more explicit and contain much more features and information which make it convenient to do the analysis. ImageNet's images are much larger, with an average resolution of about 475x400 pixels, while CIFAR's images are only 32x32, we can't even distinguish some objects from the picture using our own eyes.

6. Future Work

In this project, it is achieved that the superiority of spectral representation is demonstrated. The innovative implementation of spectral pooling and spectral filter parameterization, whose benefits lie in retaining more comprehensive information of images and providing a faster convergence in optimization. The improvements in these two aspects have stimulated a better performance of the CNN model.

The future development of this research could be conducted in two directions: to quantify the comparison of results in computational cost for the spectral convolution layer, and to provide a quantitative way for the comparison in information preservation for images.

The first potential improvement quantitatively evaluates the dominance of the spectral pooling over the traditional pooling methods, for example, the max pooling. It is claimed that the pooling in spectral domain helps to preserve more information than purely spatial domain [1], and the authors of the original paper provided an intuitive comparison of figures under these two methods of transformation. However, an improvement can be made to further elaborate this result. One possible approach of providing quantitative metrics is to validate the accuracy of classifying these post-processing images with an additional supervised learning architecture. By providing a series of labels corresponding to each post-pooling image, the information preserved from pooling can be represented by the accuracy achieved by the supervised learning network, which quantifies the probability of how likely an image after pooling is recognizable. And thus, a comparison is made between the performance of traditional pooling methods and spectral pooling techniques.

The second improvement that can be made in future is to assess the computational cost of the spectral convolutional layer. According to the original paper, the authors declare that the computational cost has slightly increased by applying spectral transformation in CNN models. It is estimated with the computational time in this research, while a more comprehensive and accurate description of the computational cost can be obtained from a combination of time elapse and GPU memory consumption in a more quantifiable format.

7. Conclusion

With this attempt of implementing the DFT in CNNs, it is successfully proved that the spectral pooling and spectral parameterizing techniques are efficient in reducing the dimensionality of an image gradually while maintaining a relatively small loss of information, and accelerating the convergence of optimization. The model embedding these techniques is trained and tested on its performance on the datasets CIFAR-10 and CIFAR-100, on which a test accuracy of 55% and 28% is achieved correspondingly.

Although the accuracy as a measure of performance on the CIFAR datasets are relatively lower than the authors achieved [5], other conclusions are verified and proved to be consistent with the original paper. Based on the agreements and disagreements presented in this report, further developments in quantifying the computational cost and information preservation.

8. Acknowledgement

We are grateful for Professor Zoran Kostic for his lectures in Neural Networks, and this opportunity to apply our skills into this project. We would like to thank Zang Mingyang, Ahmad Feroz, Angus Alexander, Naik Rachita, and Deshmukh Adit for creating such an inclusive and supportive learning environment for us. We appreciate the contribution of the Google cloud and Tensorflow authors for the powerful tools which enable us to make use of GPUs and boost efficiency for complex implementations. We would like to express our sincere thankfulness to past graduate students Aarshay Jain, Jared Samet, Alex Wainger for their previous exploration in the reproduction of the original paper, enlightening us with basic concepts. We would also like to say thank you to the authors of the original paper and for providing us with a brilliant idea in research, and an opportunity to explore the world of CNN.

9.REFERENCES

[1] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.

[2] D. Kingma and J. Ba, "Adam: A method for stochastic optimiza- tion," *arXiv preprint arXiv:1412.6980*, 2014.

[3] A.Krizhevsky, V.Nair, and G.Hinton,"The cifar-10 dataset,"*online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classifi- cation with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[6] A. Jain, J. Samet and A. Wainger, "ECBM 4040 Final Project - Spectral Representations for Convolutional Neural Networks", 2015.

10. Appendix

10.1 Individual Student Contributions in Fractions

	hl3515	yz3922	cx2273
Last Name	Li	Zhou	Xu
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	Establish CNN spectral pool layer with tensorflow keras	Set up function to create images from dataset can be read and written	Establish tensorflow keras model to validate spectral parameterization's performance
What I did 2	Build callable model containing both spectral pool and spectral convolution layers	Train and test the model built the cifar-10 and cifar-100 dataset; visualize the performance	Write utils.py file to perform load dataset cigars and define function of strip and show the picture
What I did 3	Build function to perform frequency dropout on pictures	Conducted hyperparameter search; Arrange and write half the final report	Perform frequency dropout and calculate the loss of max and spectral pooling