Fast Training of Convolutional Networks through FFTs

Vijay Aravindh Viswanathan (1219496837)

Rajesh Sathya Kumar (1219515401)

{ vviswa19, rsathyak }@asu.edu

Abstract:

Convolutional Neural Networks (CNN) have revolutionized the field of Computer Vision and pattern recognition. As the use of CNN's gets increasingly wide, the architectures that they employ also have become extremely complex over the last decade. The complex architecture and growing data requirements makes it computationally very expensive to run the algorithms, which may take weeks, if not months in several large-scale applications. In this project, we showcase an algorithm which significantly decreases the training time for any existing state-of-the-art CNN models. The algorithm involves computing convolutions as pointwise products in the fourier domain and also reusing the same map.

The algorithm will be implemented on a GPU architecture. We will be using the CIFAR-10 [2] dataset for demonstration of the running time analysis with and without using Fast Fourier Transform(FFT) in the convolutional layer.

This project is the work of Michael Mathieu, Mikael Henaff and Yann LeCun from the Courant Institute of Mathematical Sciences of NYU, cited in [1]. In this project, we do statistical analysis of how the algorithm scales for higher dimensional datasets.

Problem Statement:

Modern computer vision architectures predominantly employ the convolutional neural network. The training time of these networks increase as the architectures become more complex. The convolution operation is performed between a kernel and a 2-D matrix (input image/ output of previous layers etc). This becomes expensive as the number of layers and number of kernels in each layer increases. Modern problems incorporate millions of images as input to these networks. Modern networks use billions of parameters to train on the given data.

Any significant improvement in training time can bring about reductions in cost, emissions, compute resources etc.

Dataset:

We use the CIFAR-10 dataset. Below are the dataset specifications:

Training Data: 50000, *Test Data:* 10000, No Validation dataset

Number of classes: 10

Classes: 'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'

Citation:

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The

CIFAR-10 dataset. www.cs.toronto.edu/~kriz/cifar.html.

Source: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Baselines and Evaluation Metrics:

Model Summary:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [128, 6, 28, 28]             456
         MaxPool2d-2          [128, 6, 14, 14]               0
            Conv2d-3         [128, 16, 10, 10]           2,416
         MaxPool2d-4           [128, 16, 5, 5]               0
            Linear-5                [128, 120]          48,120
            Linear-6                 [128, 84]          10,164
            Linear-7                 [128, 10]             850
================================================================
```

Batch Size - 128

We use the above model as a baseline and get an accuracy of 61%. The baseline training time stats in milliseconds without FFT algorithm are as below for each epoch:

Mini Batch Training Time (in ms)

	Num of minibatches	Min(in ms)	Max(in ms)	Mean(in ms)	Median(in ms)	Std Dev(in ms)
Epoch 1	391.000	19.775	96.575	44.199	40.729	7.567
Epoch 2	391.000	20.708	99.861	44.424	40.718	8.248

Experiments and Results

The Goal of this project is to improve on the baseline training time with the same model using the FFT Convolution algorithm [1] in the CIFAR-10 dataset. In this project, we try to reduce the training time of the network while maintaining the accuracy.

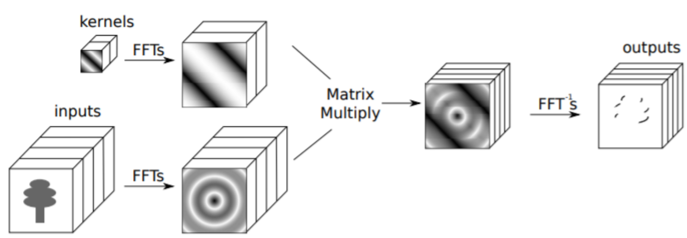

Figure 1: Illustration of the algorithm. Note that the matrix-multiplication involves multiplying all input feature maps by all corresponding kernels.

Illustration of the algorithm [1]

The need to compute FFT on each kernel and input increases the compute overhead. But the advantages become more pronounced as the input size, batch size and kernel size increase. We expect to see a decrease in the training time as the input and kernel size increase compared to the non-FFT convolutional neural network.

Citations:

1. Mathieu, M., Henaff, M., & LeCun, Y. (2014). Fast training of convolutional networks through FFTS: International Conference on Learning Representations (ICLR2014), CBLS, April 2014. Paper presented at 2nd International Conference on Learning Representations, ICLR 2014, Banff, Canada.
2. Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. , , 32--33.