# Design and Acceleration of Machine-Learning back-ends on modern Architectures

Students: Alex Gessinger(SRU), Sihan Chen(CUHK)
Mentors: Dr. Stanimire Tomov(UTK), Dr. Kwai Wong(UTK)

## Abstract

Convolutional Neural Networks are extremely useful in computer vision and many other related fields, but the computation of them tends to be extremely expensive in many cases. The aim of this research project is to accelerate Convolutional Neural Networks, while it is divided into two directions:

- To design a machine-learning back-end on GPU using the MAGMA library to using efficient algorithms;
- To analyze the performance of various machine learning back-ends. Utilizing magmaDNN which uses the magma back-end and Keras, a high-level Python wrapper for which one of three machine learning back-ends can be specified, TensorFlow, Theano, or CNTK. Using well known CNN benchmarks such as MNIST, CIFAR-10, and ImageNet.

## Goals

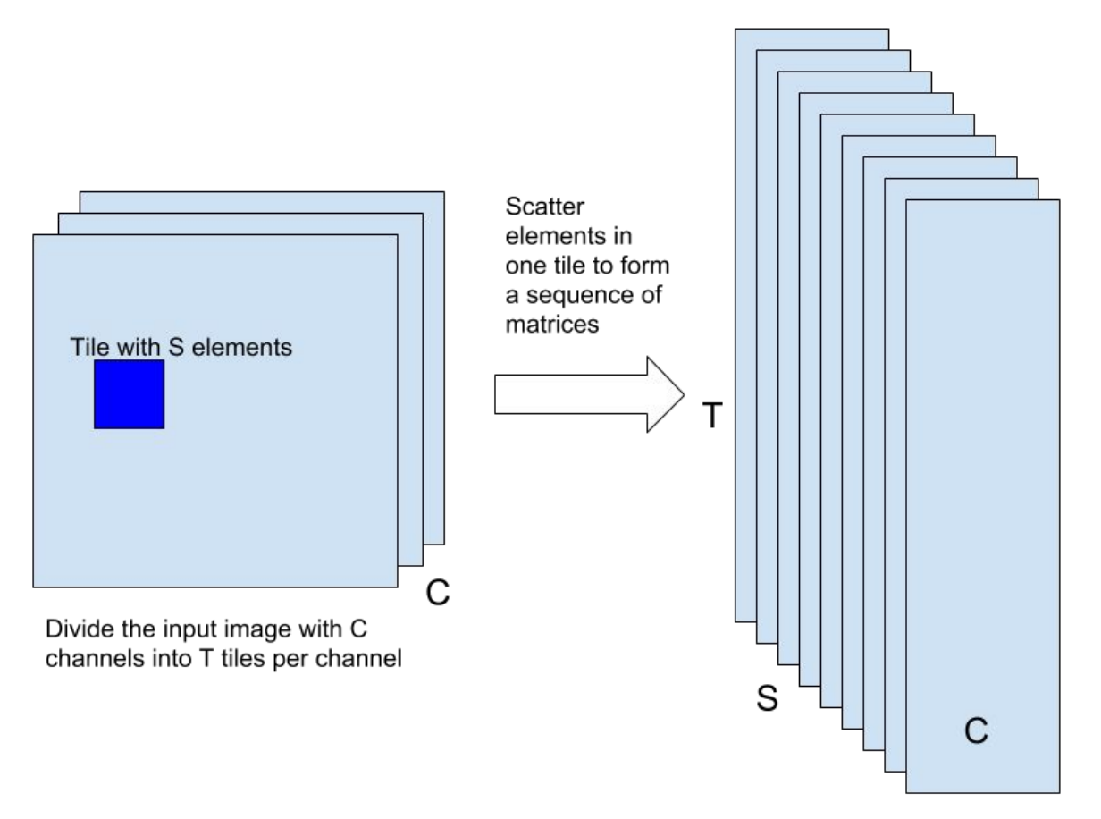
- To design an algorithm to efficiently compute the convolutional layers of CNN. Try it on CPU first to compare the efficiency and verify the accuracy of the algorithm, then implement it on GPU using C with MAGMA library. Do experiments on benchmarks with classic CNN architectures ((e.g., CIFAR-10 on LeNet-5), compare the results with other implementations of CNN on GPU, such as cuDNN.
- Add a pruning step in the convolutional layer, compare the performance (both accuracy and speed) between vanilla CNN (without pruning) and with pruning in spatial domain and Winograd domain respectively.
- Analyze Convolutional Neural Network hyperparameters and their impact on performance related to training time and accuracy on various architectures such as Lenet-5 and VGG-16.
- Create various CNN architectures using magmaDNN and Keras catered toward specific dataset benchmarks e.g., Images of cats and dogs, MNIST dataset, ImageNet dataset.

## Winograd Algorithm

The Winograd minimal filtering Algorithm transforms the element-wise matrix-matirx multiplication into GEMM, which can have better efficiency on GPU with MAGMA library. For any data $d \in \mathbb{M}_{m\times m}$ and filter $g \in \mathbb{M}_{r\times r}$ where $m, r \in \mathbb{N}$, we can always find constant matrices $A, G, B$ such that we can represent their convolution $Y$ as

$$Y = A^T[[GgG^T] \odot [B^T dB]]A$$

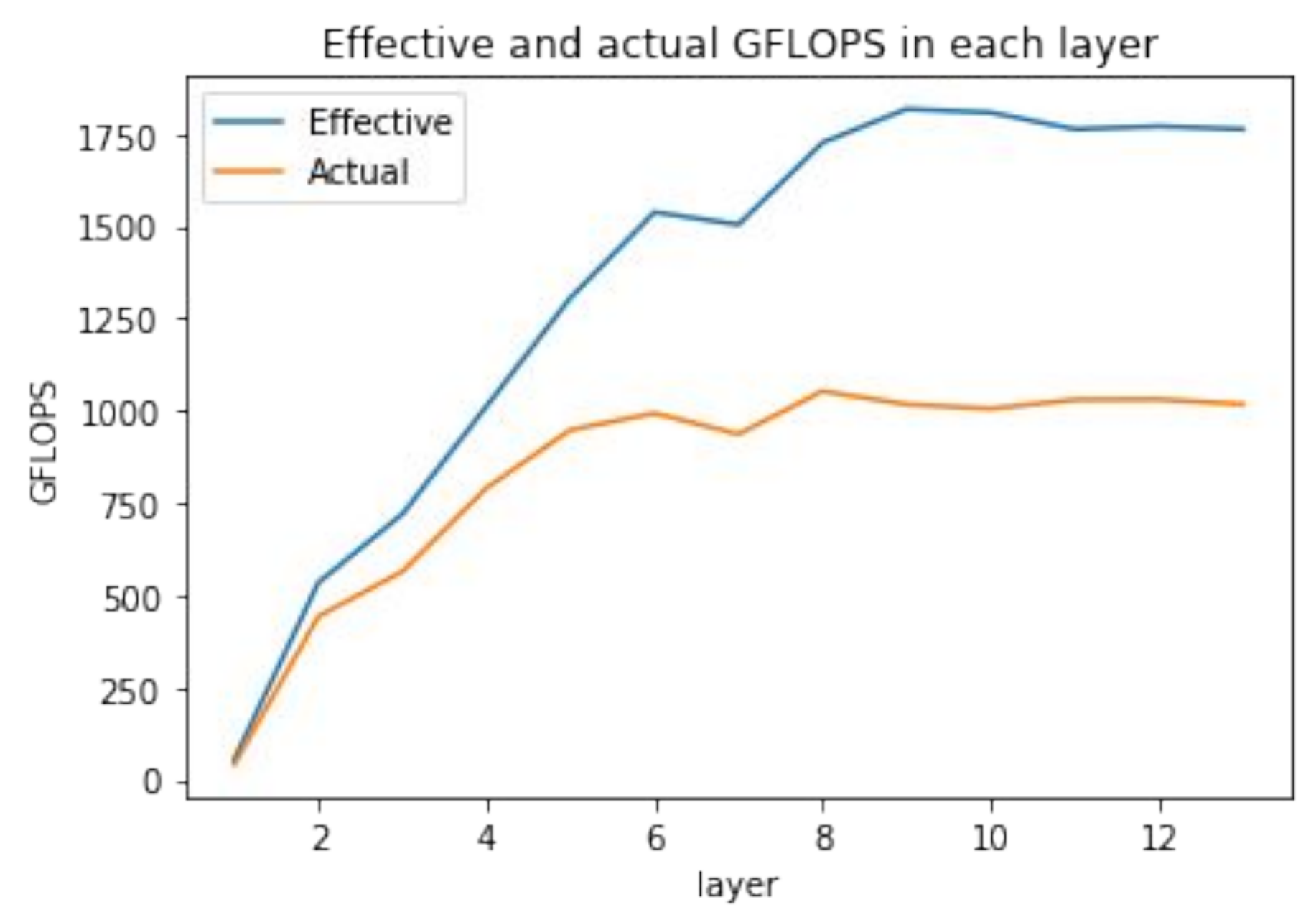
therefore we have reduced the time of element-wise multiplications from $(m-r+1)^2 * r^2$ down to 1. Moreover, we can transform the remaining one into GEMM by dividing the transformed images $(B^T dB)$ into small tiles, scattering it into a sequence of matrices along the elements of each tile, and do similar things to transformed filters. Let $T$ denote the number tiles, $S$ being the number of elements in each tile, $C$ being the number of channels, $K$ denoting the number of filters, then we can transform $[GgG^T] \odot [B^T dB]$ into the summation of $S$ times of GEMM with matrices of size $T \times C$ and $C \times K$.


Tile with S elements
C
Divide the input image with C channels into T tiles per channel
Scatter elements in one tile to form a sequence of matrices
T
S
C

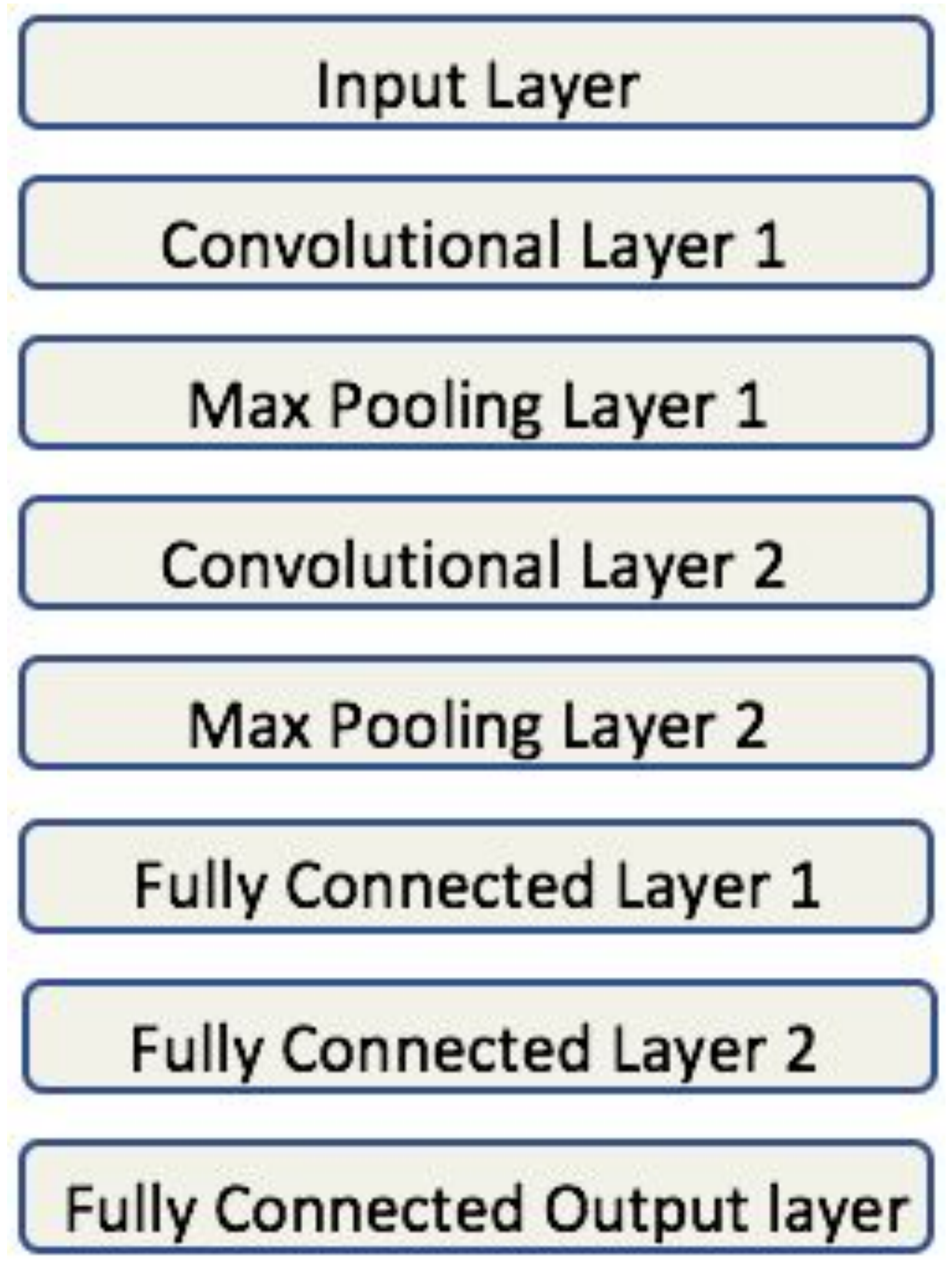
A simple illustration on how to scatter an input image with C channels. We divide it into T tiles (with overlap) of S elements.

## Experiment on CPU

We have tested the algorithm described above on CPU, while the GPU work is still in process. In each layer we conduct the convolution twice, with conventional computation and Winograd algorithm respectively, to verify the accuracy of Winograd algorithm. As for performance, we use GFlops to measure that. In the experiment, we calculate the actual GFlops (representing the actual computations needed in Winograd algorithm) and the effective GFlops (representing the computations need to complete the same work using conventional method to calculate convolution). VGG-16 network with input images of size 226*226*3 was used in this experiment.



## LeNet-5 Architecture

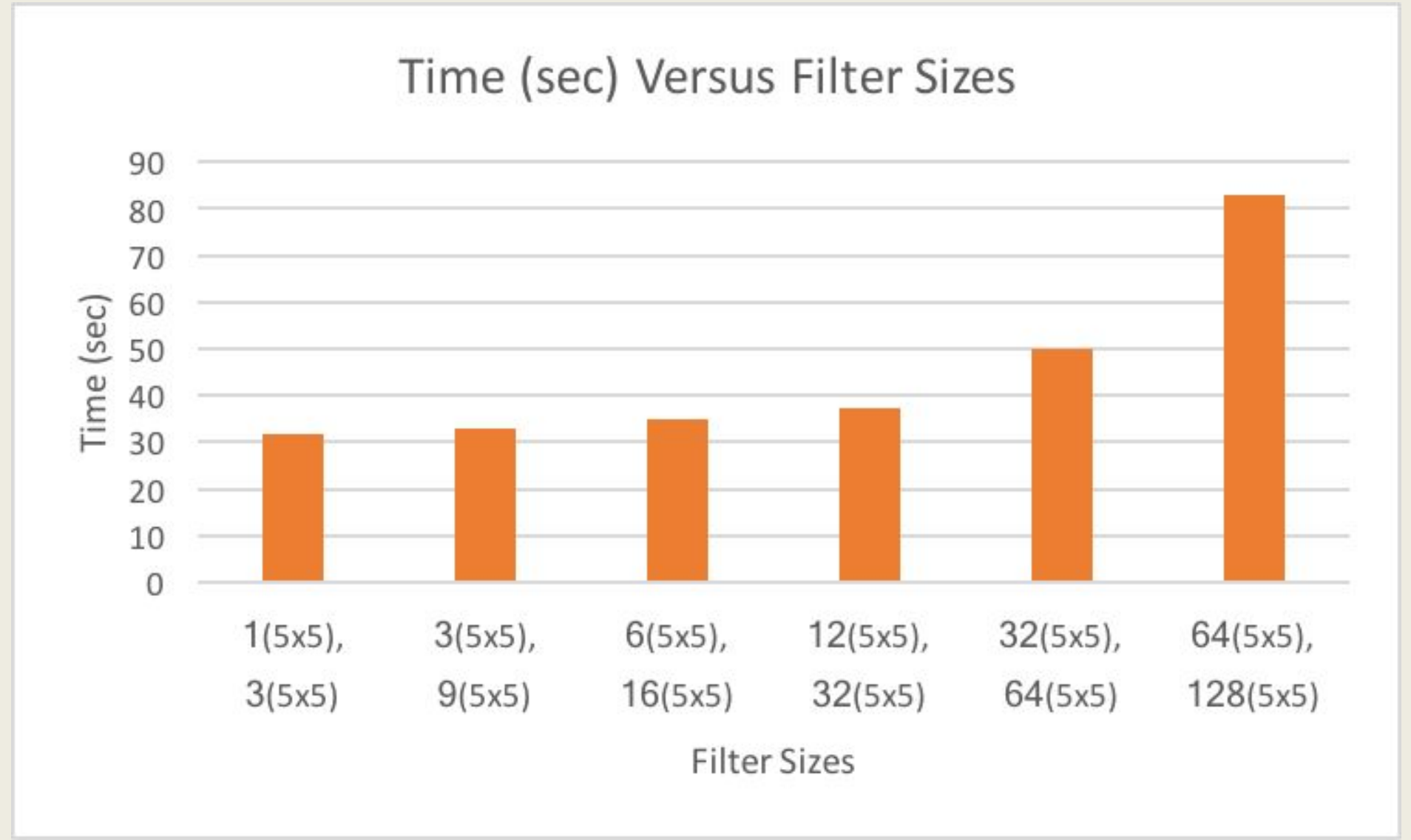


| Hyperparameters | |
|---|---|
| Learning Rate | 0.01 |
| Activation Functions | Rectified Linear Unit(ReLU) Sigmoid for output layer |
| Batch size | 64 |
| Epochs | 15 |
| Conv 1 | 6 5x5 Filters |
| Max Pooling 1 | 2x2, Stride of 2 |
| Conv 2 | 16 5x5 Filters |
| Max Pooling 2 | 2x2, Stride of 2 |
| Fully Connected 1 | 120 Neurons |
| Fully Connected 2 | 84 Neurons |

The LeNet-5 architecture is the most simplistic classic CNN architecture. We implemented Lenet-5 in magmaDNN and Keras to quickly evaluate various CNN hyperparameters and architectures.

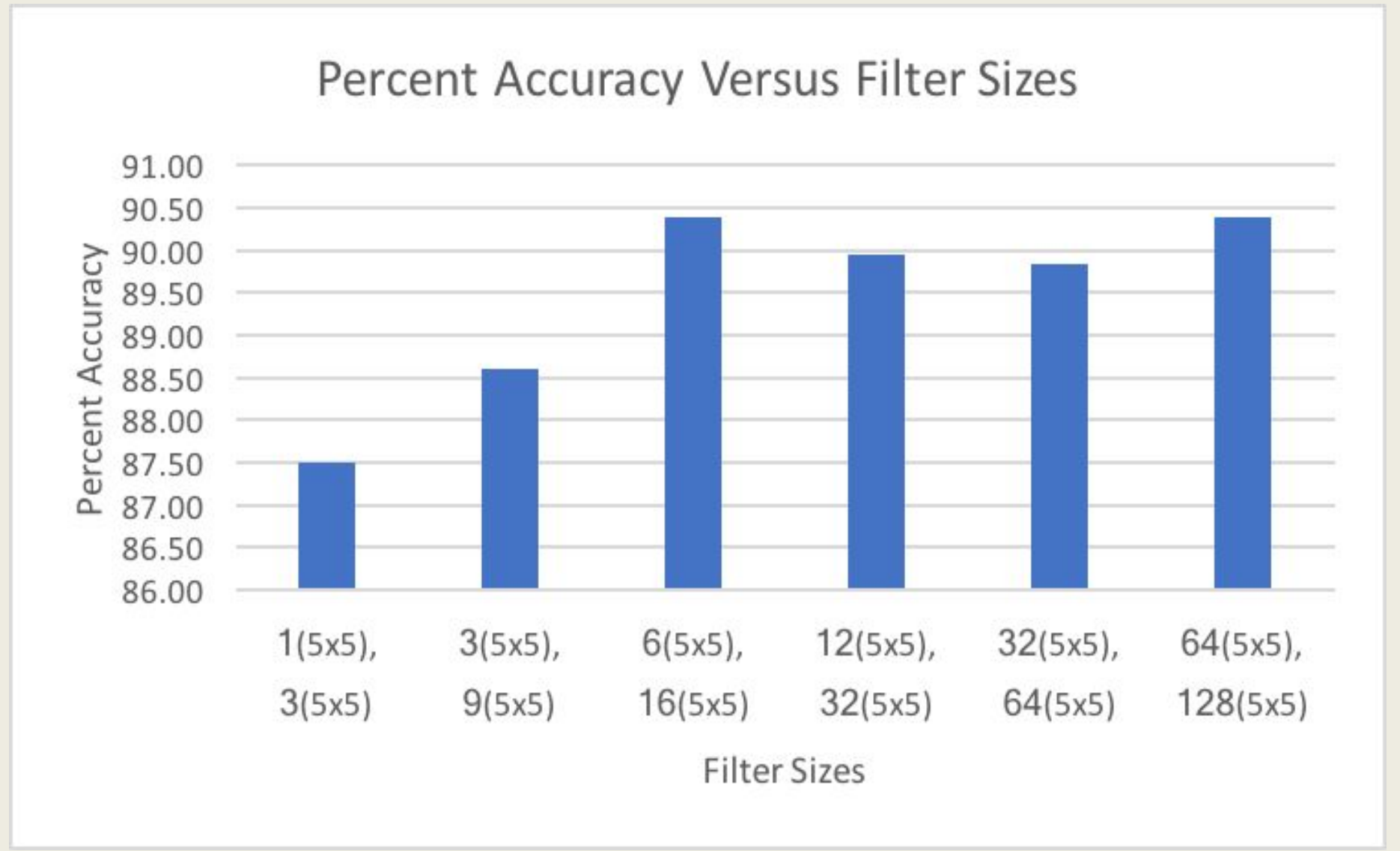
## Example Tuning Filter Size

The graphs below show the result of modifying the LeNet-5 implemented in magmaDNN varying filter sizes. Viewing the accuracy of the model as well as the total time to train the model. Batch size, learning rate, number of epochs, number, type and order of layers remained unchanged. The model was trained on the CIFAR-10 dataset which consists of 60,000 images of 10 classes 6,000 Images per class split 5:1 training: testing [1].



As the number of filters increases, training time increases.This is due to the added parameters the network is learning. On larger data with more complex models this time could be weeks.

The result on accuracy shows that added filters only increase accuracy to an extent. The accuracy plateaus at 90% with 6 and 15 filters and training time of 35 sec. A larger number of filters only increases training time.



## Future Work

- Compare magmaDNN LeNet-5 with the Keras implementations measuring training time.
- Continue Modify CNN omdel and tune hyperparameters.
- Create a Python wrapper for magmaDNN.
- Complete the GPU implementation of Winograd Algorithm for CNN with MAGMA, compare with other implementations.

## Acknowledgements

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at University of Tennessee Knoxville. This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation.In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team.

## References

[1] "CIFAR-10 and CIFAR-100 datasets," Department of Computer Science, University of Toronto [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html
[2] Kanan, C. and Cottrell, G. (2012). Color-to-Grayscale: Does the Method Matter in Image Recognition?. *PLoS ONE*, 7(1), p.e29740.
[3] Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), pp.2278-2324
[4] R. Rere, M. I. Fanany, and A. M. Arymurthy, "Metaheuristic Algorithms for Convolutional Neural Network," May 2016.
[5] Lavin, A; Gray, S. Fast Algorithms for Convolutional Neural Networks. arXiv:1509.09308, 2015