# Proposal: Accelerating VGG16 Network on FPGA

Wenqi Jiang(wj2285), Manqi Yang(my2577)

February 2019

## 1 Introduction

Convolutional Neural Networks (CNN) are widely used in Computer Vision tasks such as Image Classification, Object Detection and Semantic Segmentation. The amount of processing required by CNNs leads to an increasing need on dedicated hardwares. FPGAs, due to its programmable property and hardware acceleration capacity, can serve as the intermediate product accelerating CNNs between General Purpose Graphics Processing Unit (GPGPU) and Application-specific Integrated Circuit (ASIC)[1][4].

In this project, we will implement CNN computation on FPGA: accelerate VGG16 network on DE1-SOC board. In section 2, we will discuss several optimization methods for CNN acceleration on FPGAs. Then, assessment of this project and milestones will be set in section 3 and 4 separately.

## 2 Methods

In this section, we will introduce optimization methods for CNN from several aspects. Firstly, the convolution computation itself can be optimized by algorithms such as Winograd and Fast Fourier Transform: these methods reduce the multiplication times of convolution. Secondly, maximize data reuse during the stream processing process can decrease energy consumption significantly. Finally, using fixed point computation is much more computationally efficient. Without losing much accuracy, we can both save energy cost and increase throughput.

### 2.1 Algorithmic Optimization for CNN-Acceleration on FPGAs

Two popular algorithms, Winograd and Fast Fourier Transform are widely used today accelerating CNNs[3], both of them use the philosophy of reducing the number of multiplications.

One of the algorithm is Winograd, which separates the convolution into several parts and one of the parts only consists of filter weights. This part can be computed by CPU and feed into the network inference as constants. Thus, Winograd can reduce the time of multiplications using this preprocessing strategy.

Another algorithm is Fast Fourier Transform, which transforms the convolution in time domain to frequency domain. While paying cost on FFT and Inverse FFT, this method can reduce multiplications dramatically.

In this project, we will use Winograd because this method is specifically useful on small filters, e.g. 3 x 3 filters on VGG16 network structure.

### 2.2 Energy-Efficient Dataflow for CNN

The most significant limits computing convolutions on FPGA are the limited Processing Element and the bandwidth constraint communicating with DRAM. By wisely reuse data and Processing Elements, we can save the energy cost of FPGAs.

In this project, we will employ row stationary strategy[2], which has been proved as a successful dataflow structure. It aims to maximize the reuse and accumulation at RF level for all types of data for overall energy efficiency.

### 2.3 Approximate Computing for CNN

Aside from Algorithmic and Data-path Optimizations, CNN execution can also be accelerated by approximate computing. There are two directions to implement this: by reducing digital accuracy, or by reducing computations.

There are four methods for reducing digital accuracy: Static Fixed Point (SFP), Dynamic Fixed Point (DFP), pseudo-Binary Nets, Stochastic Computing (SC). We will use SFP in this project.

For SFP, instead of floating point, Feature Maps (FM) and weights are quantized by a fixed point scheme, thus numbers are encoded with the same bit-width (bw). To prevent overflow, there is also a formula to compute the upper bound of bit-width using the bits that FMs and weights are allocated.

## 3 Assessment

### 3.1 Correctness

We will download or generate some datasets, using both FPGA and our own computers to implement some matrix and convolution operations, and compare their results. Since we use SFP method, some extent of inaccuracy are tolerable.

### 3.2 Throughput

Same as above, we will use both FPGA and our own computers, and compare how many operations they can complete in a same period of time.

### 3.3 Energy Efficiency (optional)

We will try to measure the energy the FPGA consumed if possible.

## 4 Milestones

Milestone 1 (Apr 5) Construct the basic building blocks of the whole neural network, including convolutional layers, fully-connected layers, pooling layers and activation function. These blocks are implemented by System Verilog.

Milestone 2 (Apr 19) Implementing the dataflow strategy. Maximize the data reuse to achieve better energy efficiency.

Milestone 3 (May 3) Test the overall throughput and energy efficiency and write up necessary documents.

## References


[1] Kamel Abdelouahab, Maxime Pelcat, Jocelyn Serot, and François Berry. Accelerating cnn inference on fpgas: A survey. *arXiv preprint arXiv:1806.01683*, 2018.

[2] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 367–379. IEEE Press, 2016.

[3] Liqiang Lu, Yun Liang, Qingcheng Xiao, and Shengen Yan. Evaluating fast algorithms for convolutional neural networks on fpgas. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 101–108. IEEE, 2017.

[4] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.