# ECE 2195 Project Report

**Hardware AI acceleration with data pre-processing**

**Team member:**

Yue Tang [YUT51@pitt.edu](mailto:YUT51@pitt.edu)

Jinming Zhuang [JIZ230@pitt.edu](mailto:JIZ230@pitt.edu)

1. **Publicity**

We agree to include our project in the ECE2195 22Spring Class Project Website

1. **Problem Descriptions**

In recent years, various researchers have incorporated machine learning (ML) and convolutional neural network (CNN) to a wide range of applications in different domains including disease diagnosis [1][2][3], natural language processing (NLP) [4][5], electrocardiogram (ECG) signal classification [6][7], image dehazing [8] and so on. However, deep learning algorithms incur false-positive and false-negative cases when targeting the aforementioned applications. Pre-processing is deemed to be an effective method for improving the performance of DL models.

In this project, we plan to re-implement several state-of-art ML algorithms with and without pre-processing to verify the accuracy gain provided by the pre-processing steps. We have two tasks: GridDehazeNet for Image Dehazing and CNN based image classification with ZCA. For Task 1, we reproduce the GridDehazeNet algorithm training on the Indoor Training Set (ITS) of the RESIDE dataset [9] and testing on the Synthetic Objective Testing Set (SOTS) of the RESIDE dataset [9]. For Task2,….Besides, in order to deploy the pre-processing and ML algorithms on a 16nm data-center AMD/Xilinx U200 board, we will build analytical models to guide the pre-processing and CNN kernel design.

1. **Background and Discussion on Prior work**

For different applications, several pre-processing methods are added to improve the inference performance. [1] and [2] used the Sobel filter and median filter before classifiers to improve the accuracy of disease detection. Before doing text classification, [4] took tokenization to break the raw text into words. [5] covers most of the common pre-processing techniques for NLP including lemmatization, stemming, handling negations, and so on. For ECG signal classification applications, [6][7] implement their filter to remove the noises under/above a certain frequency. For image dehazing, which can recover the clear version of a hazy image, [8] proposed an end-to-end trainable CNN network, named GridDehazeNet, which consisted of three modules: pre-processing, backbone, and post-processing. [10]…

To implement CNN, [11] exploited various optimization techniques including loop unrolling, loop tiling, and loop transformation on the FPGA accelerator, and proposed a roofline model to quantitatively analyze its computing throughput and required memory bandwidth.

1. **Proposed Methodology**

In this project, we plan to implement several state-of-art ML algorithms and compare the inference performance with and without data pre-processing. The algorithms that are expected to be implemented are as follows.

Task 1. GridDehazeNet for Image Dehazing.

[8] proposed an end-to-end trainable CNN, named GridDehazeNet, for single image dehazing. The GridDehazeNet consists of three modules: pre-processing, backbone, and post-processing, which is illustrated in Figure 1. The pre-processing module contains a convolutional layer and a residual dense block (RDB) to effectively converts the single image dehazing problem into a multi-image dehazing problem by generating several variants of the given hazy image. The backbone is a combination of RDB blocks and upsampling/downsampling blocks. The post-processing module is also an RDB followed by a Conv layer, which aims to improve the quality of the dehazed image. The GridDehazeNet is trained on the Indoor Training Set (ITS) of the RESIDE dataset [9] and tested on the Synthetic Objective Testing Set (SOTS) of the RESIDE dataset [9]. The evaluation metric is peak signal to noise ratio (PSNR) and structure similarity (SSIM).



Figure 1 The architecture of GridDehazeNet[8]

Firstly, we reproduce the GridDehazeNet algorithm with and without the pre-processing module on GPU. The whole GridDehazeNet has 111 Convolution (Conv) layers. Followed by [8], to evaluate the performance of the network without pre-processing, we remove the pre-processing module and replace the first three learned inputs by the RGB channels of the given hazy image and the rest by all-zero feature maps. Such network has 105 Conv layers. The input image size of the testing dataset is 620460 ().

Then, we build analytical model to find the optimal design parameters to map the testing phase on U200 FPGA board. Followed by [11], the parameters we need to optimize include Tm, Tn for the whole network and Tr and Tc for each convolution layer. We use eq. () to calculate the *computation clock cycles*. Then, we use the following equation to calculate the *communication clock cycles,* where *f* is the working frequency of the FPGA board, *DDR\_BW* is the theoretical DDR bandwidth, and the *total amount of external data access* is calculated by eq. (). In FPGA-based accelerator design, double buffer is adopted so that the computation and communication can be processed in parallel. Therefore, to estimate the total cycles of the whole Conv layer, we select the maximum of the *computation clock cycles* and the *communication clock cycles*.

()

It should be noted that, we have more than 100 layers to schedule the optimal parameters, the input image size is 620460, the maximum number of input channel is 128, and the maximum of output channel is 64. Therefore, complexity of searching the best Tm, Tn, Tr, and Tc is around 10012864620460≈2.341011, which costs so many times. Therefore, we need to make an optimization. We find that the value of the *computation clock cycles* is mainly dependent on the Tm and Tn, and the selection of Tr and Tc have little impact on it. As for the communication latency, it decreases when the TrTc increases. Under the same TrTc, the smaller number of Tc can decrease the continuity of data. Previous works have shown that the discontinuity of memory access can degrade the DMA transferring speed from about 8GB/s to around 1GB/s [12]. Therefore, we fix Tc=C to improve memory access continuity and reduce the searching complexity.

Task 2. CNN based image classification with ZCA

In [10], Zero Component Analysis (ZCA) has been proposed to preprocess images before input data entering the CNN model while the mean normalization and standardization filter serves as the baseline for comparison with different pre-processing techniques. In this project, we first reproduce the CNN models which were used in [10] and apply different pre-processing kernels before entering CNN. Then we estimate the inference latency using the same way in Task 1. Finally, we will figure out how to implement the ZCA on the FPGA and estimate the inference latency by the analytical model.

1. **Artifacts Description**

* The project code Github repo: <https://github.com/JinmingZhuang/ECE2195>.
* Hardware details: We first reproduce the GridDehazeNet algorithm of Task 1 with and without the pre-processing module and the CNN model of Task 2 with and without ZCA on the V-100 GPUs from the Pittsburgh Supercomputing Center (PSC) server to validate the PSNR and SSIM of Task 1 and classification accuracy of Task 2 with and without data preprocessing. Then, we build the analytical model to estimate the CNN inference latency under the hardware constraints of U200. The number of DSPs is 6833, the on-chip buffer size is 41.92MB, the datatype is floating point 32 bit, the clock frequency is 250MHZ, and the DDR bandwidth is 77GB/s.
* Software dependencies: For Task 1, we install Python 3.7 on the GPU with the CUDA version as 10.2. The GridDehazeNet is built on the Pytorch framework with the Pytorch version as 1.7.0 and Torchvision version as 0.8.0. Other necessary software packages are shown in our Github repo. The analytical model can be successfully implemented with the same software dependencies.
* Installation: For Task 1, we first download the ITS training dataset and the SOTS testing dataset. The required software packages can be installed my ‘conda’ or ‘pip’ command.
* Experiment workflow: In Task 1, we first train the GridDehazeNet with and without the pre-processing module. Same with [8], we train the model for 100 epochs, the learning rate is 0.001, the crop size is 240240. Due to the CUDA memory limitation, the batch size decreases from the original 18 to 16. Then, we test the well-trained model on the testing dataset. Same with [8], we use the original size of the image, which is 620460. After that, we build an analytical model to estimate the testing phase targeted on the U200 board. The GridDehazeNet with the pre-processing module has 111 Conv layers, and the network without the pre-processing has 105 Conv layers. The commands to run the code can be seen in the repo.
* Evaluation and expected results: For Task 1, after training for 100 epochs, the testing results of GridDehazeNet with the pre-processing module are PSNR as 32.12 and SSIM as 0.9833, while the results reported in the original paper are PSNR as 32.16 and SSIM as 0.9836. The results of GridDehazeNet without the pre-processing module are PSNR as 30.03 and SSIM as 0.9789, while the results reported in the original paper are PSNR as 31.48 and SSIM as 0.9820. From our analytical model, the optimal Tm and Tn for GridDehazeNet with and without pre-processing are both Tm=16 and Tn=80, which means the two situations can use the same FPGA design framework. As for the estimated latency, the GridDehazeNet with pre-processing takes up around 142901583 clock cycles, while the network without pre-processing takes up around 129711843 clock cycles, so the pre-processing module takes up around 9.23% of the total latency of GridDehazeNet. The results of every step are also uploaded in our repo.

**Reference:**

[1]Beeravolu, Abhijith Reddy, et al. "Preprocessing of breast cancer images to create datasets for deep-cnn." IEEE Access 9 (2021): 33438-33463.

[2]Wang, Yu, and Shan Shan. "Accurate disease detection quantification of iris based retinal images using random implication image classifier technique." Microprocessors and Microsystems 80 (2021): 103350.

[3]Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." Computers and electronics in agriculture 145 (2018): 311-318.

[4]Vijayakumar, T., and R. Vinothkanna. "Capsule network on font style classification." Journal of Artificial Intelligence 2.02 (2020): 64-76.

[5]Symeonidis, Symeon, Dimitrios Effrosynidis, and Avi Arampatzis. "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis." Expert Systems with Applications 110 (2018): 298-310.

[6] Diker, Aykut, et al. "A novel ECG signal classification method using DEA-ELM." Medical hypotheses 136 (2020): 109515.

[7] Park, Yeonghyeon, Il Dong Yun, and Si-Hyuck Kang. "Preprocessing Method for Performance Enhancement in CNN-based STEMI Detection from 12-lead ECG." IEEE Access 7 (2019): 99964-99977.

[8] Liu, Xiaohong, et al. "Griddehazenet: Attention-based multi-scale network for image dehazing." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.

[9] Li, Boyi, et al. "Benchmarking single-image dehazing and beyond." IEEE Transactions on Image Processing 28.1 (2018): 492-505.

[10] Pal, Kuntal Kumar, and K. S. Sudeep. "Preprocessing for image classification by convolutional neural networks." 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2016..

[11] Zhang, Chen, et al. "Optimizing fpga-based accelerator design for deep convolutional neural networks." Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays. 2015.

[12] Guan, Yijin, et al. "FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates." 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2017.