



PIRM Presentation



EE / CprE / SE 492 – sddec20-proj01

Machine learning for pilot biometrics

Team Members:

Jianhang Liu, Feng Lin, Xuewen Jiang, Xiuyuan Guo,
Sicheng Zeng, Junjie Chen

Email: zengsc@iastate.edu

Introduction



- **Problem:**
 - **Hypoxia, Fatigue, Strain Doubles would cause pilots to crash their airplane, huge losses economically and not to mention they put their life in danger**
- **Project Statement:**
 - **Machine Learning algorithm has been developed to detect the blink rate on pilots, based on research, several blink patterns are related to Hypoxia, Fatigue and strain Doubles. Machine learning is loaded onto a fast processing edge device(FPGA) to detect such fatigue patterns. Our goal is to improve the existing machine learning algorithm, in terms of accuracy and efficiency by accelerating the algorithm from both software and hardware perspective.**

Camera Interface



Project goals:

1. Build our own Daughter card (MIPI adapter for Ultra 96) for Ultra 96 (Processor Board)
2. MIPI camera works well, reducing the latency and increasing the accuracy for Ultra 96

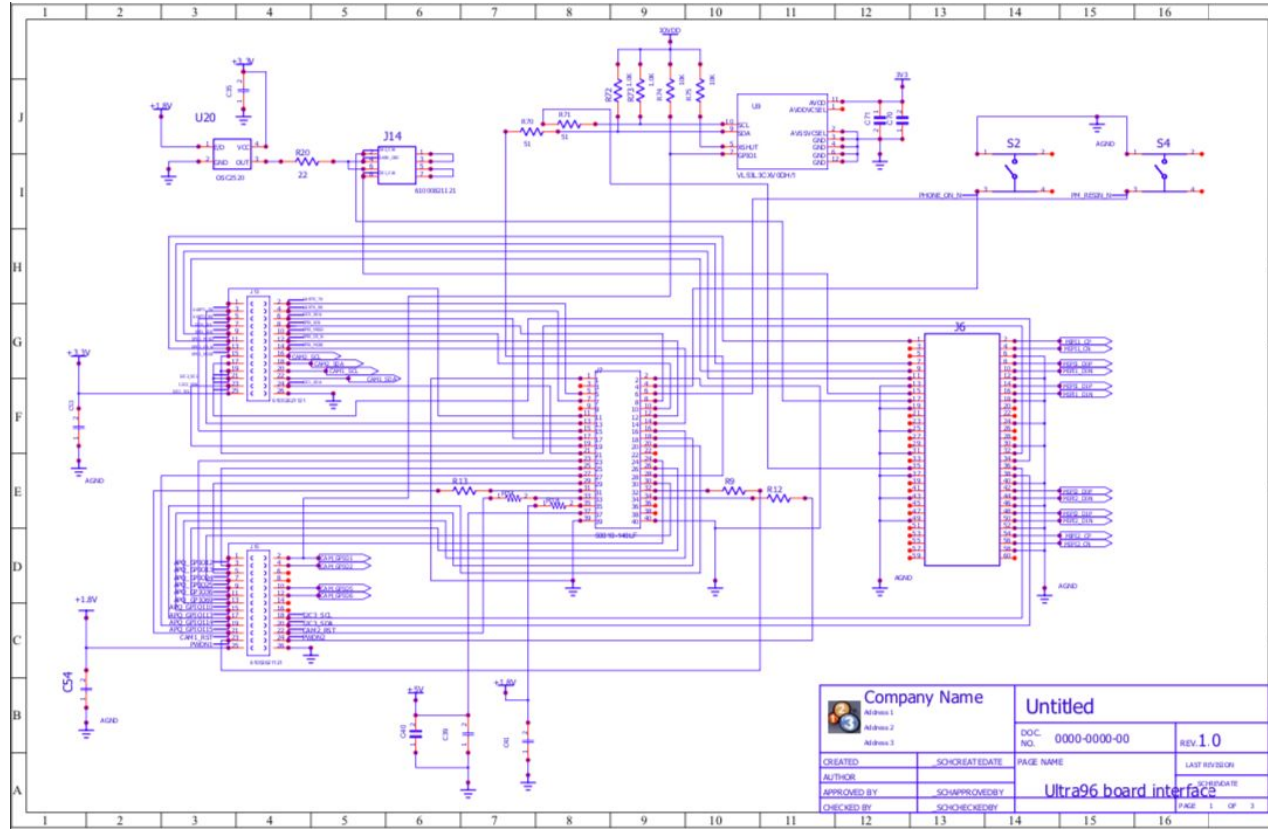
PCB Schematic for Ultra 96

Tools: PCB 123

Designer: Xuewen Jiang

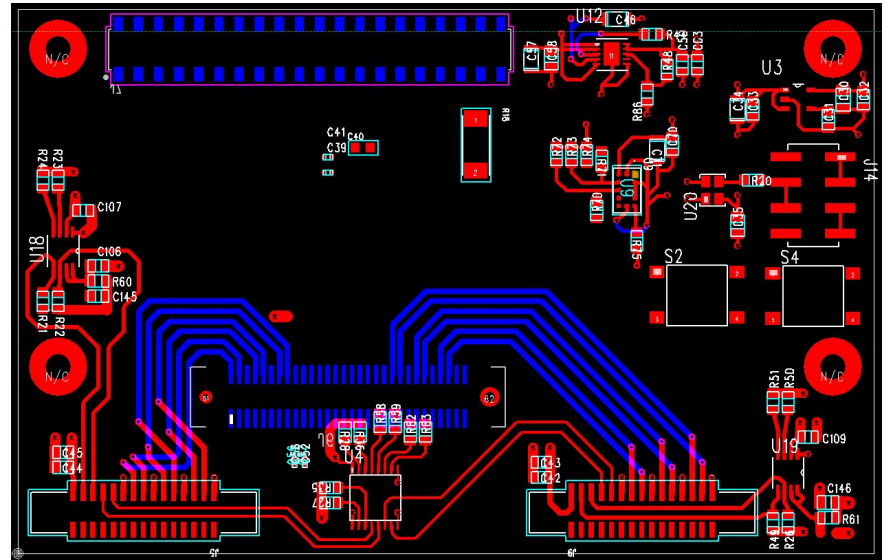
Challenge: How to make the schematic professional

Goals: Optimize it and will finish all the design in several week



PCB layout of Ultra 96 daughter card...

- Tools: PCB 123
- Designer: Jianhang and Issac
- Most parts are completed, expect another several weeks to finish.
- Acted as the interface hardware to connect to Ultra 96 board and Camera
- Challenges: Manage traces between ports and components on PCB to have better utilization of spaces





Use image processing to improve performance

Two main possible ways to increase whole performance:

1. Improve the input image of the system:

Apply the image filter to reduce the interference and calculation
--resize, noise reduction, overexposure...

1. Improve the system (algorithm) itself:

Use the pre-processed image to replace specific layer(s) in algorithm
--Gabor filter

Different image filter may or maynot benefit the system, some may even have side effects, every filter used needs to be tested and examined first.



Edge detection filter for a flower
Image from MathWorks

challenges on image processing & future plans

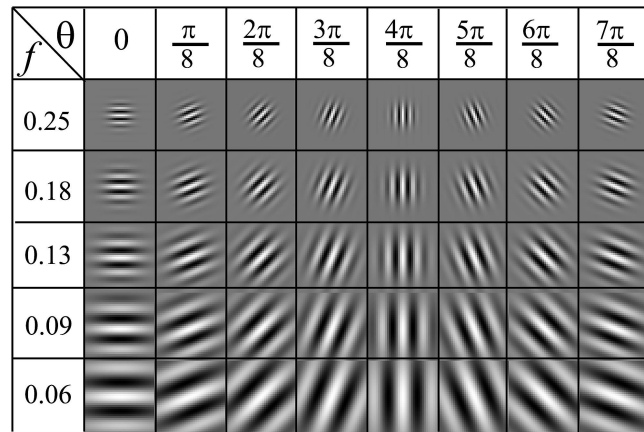
1. Challenges from improve input image of the system:

Final results from different image processing methods may vary: Some benefit the system, some did not changes, and some did bad effect. Choose filter parameter will directly affect the result.

1. Challenges from improve the algorithm itself:

Cannot observe the logic of algorithm directly, not easy to find the proper image filter to replace algorithm layers, may cause side effects if not handled properly.

Future plans: Do the image processing on both directions, take multiple trials and guess to achieve goals.



Gabor filter and its parameters, which to choose?
Image from Figshare

FPGA in Machine Learning

- In our project we are going to use FPGA(field-programmable gate array) to accelerate inference of the machine learning trained model.
- FPGA can do parallel computation(processor only do sequential computation)
- Our project is related to images processing which involved tons images computation(such as matrix multiplication), FPGA is faster than normal CPU.
- In our project I am using Xilinx DPU(Deep learning process unit) technology to approach FPGA acceleration goal.

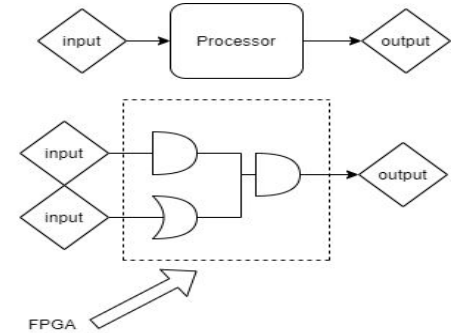
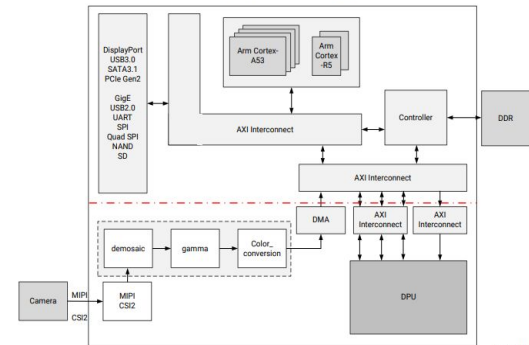


Figure 3: Example System with Integrated DPU



Reference:https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_2/pg338-dpu.pdf



Make algorithm have less memory

In the beginning, we set a primary goal for the whole semester to make the algorithm have less size but only reduce a few accuracy rate. After that, transfer the code to the correct format that can successfully run in Ultra 96 board. I focused on the task this semester and got a pretty good result.

When I start to do the task, I search how to make the TensorFlow model got less size. After I read several essays, I choose to use the pruning method to do the task. Weight pruning means eliminating unnecessary values in weight tensor. Users can get a pruned model in the end.



Prune challenge

I use TensorFlow and Keras in the pruning process. During pruning, two elements are most important, which are sparsity and schedule that directly affect final model result .

I set epochs, initial_sparsity, and final sparsity to achieve the best outcome that is less memory and more accuracy. During the process, I read lots of essay to learn new technology and try to do my best. I choose four epochs and 50% to 90% sparsity in the openclose_eye model by testing. In the end, I got a pruned model in my computer.

Sicheng Zeng (2)



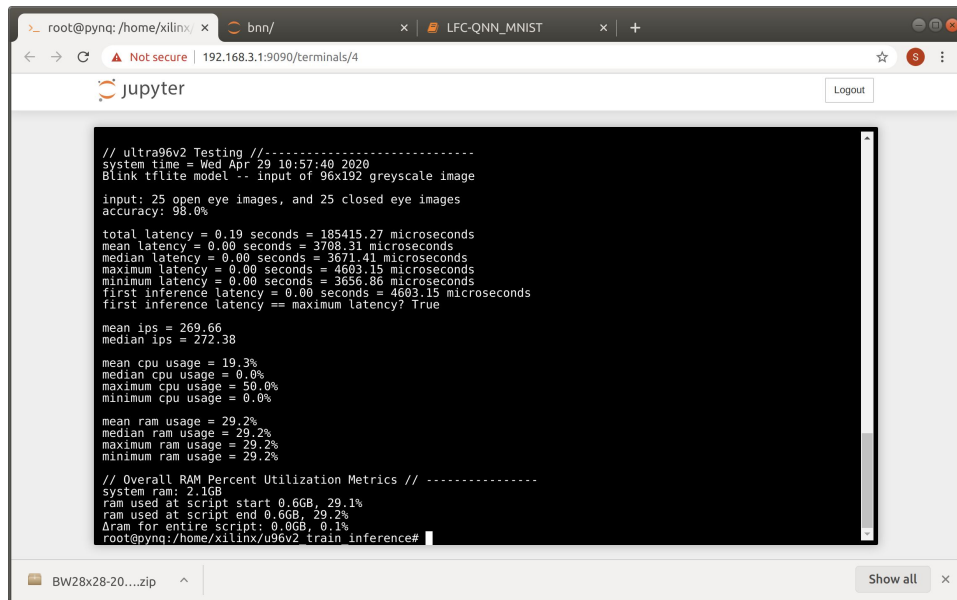
Transfer to tensorflow lite and Run in Ultra 96

After pruning process, I got a pruned model. Now, I transfer the model to the TensorFlow Lite format, which is required for Ultra 96. After transfer, I got an 80% less size model with 84% accuracy in Ultra 96 board. I also learned how to run TensorFlow Lite on the board, but I can't have one because of the coronavirus.

In beginning, we are all newbie for machine learning. However, we grow into the area and seek better technology during the semester. Thank everyone help us in the semester.

In summer, we still continue meeting once a week and I change a new method by tensorflow official to got a better result. And I start to research a new area called Lucid. It is a visualization tool that can help myself to determine optimize which layer area.

Sicheng Zeng (3)



```
// ultra96v2 Testing //-----  
system time = Wed Apr 29 10:57:40 2020  
Blink tflite model -- input of 96x192 greyscale image  
  
input: 25 open eye images, and 25 closed eye images  
accuracy: 98.0%  
  
total latency = 0.19 seconds = 185415.27 microseconds  
mean latency = 0.00 seconds = 3788.31 microseconds  
median latency = 0.00 seconds = 3671.41 microseconds  
maximum latency = 0.00 seconds = 4603.15 microseconds  
minimum latency = 0.00 seconds = 3656.86 microseconds  
first inference latency = 0.00 seconds = 4603.15 microseconds  
first inference latency == maximum latency? True  
  
mean ips = 269.66  
median ips = 272.38  
  
mean cpu usage = 19.3%  
median cpu usage = 0.0%  
maximum cpu usage = 50.0%  
minimum cpu usage = 0.0%  
  
mean ram usage = 29.2%  
median ram usage = 29.2%  
maximum ram usage = 29.2%  
minimum ram usage = 29.2%  
  
// Overall RAM Percent Utilization Metrics // -----  
system ram: 2.1GB  
ram used at script start 0.6GB, 29.1%  
ram used at script end 0.6GB, 29.2%  
Δram for entire script: 0.0GB, 0.1%  
root@pynq:/home/xilinx/u96v2 train inference#
```

In the end, I successfully
prune the model from
3.44mb to 42kb



Setting up ultra 96-v2 inference metrics

- Our project is improving the existing algorithm in terms of inferencing latency and memory usage, therefore setting up the proper metrics is a necessary and essential aspect that should be done early
- Measured existing algorithm metrics, with quantization to 8 bit using tensorflow lite

Junjie Chen



Inferencing baseline with tf-lite

```
root@pynq:/home/xilinx/u96v2_train_inference# python3 lite_inference.py
// ultra96v2 Testing //-----
system time = Mon Mar 16 08:12:30 2020
Blink tflite model -- input of 96x192 greyscale image

input: 25 open eye images, and 25 closed eye images
accuracy: 92.0%

total latency = 2.01 seconds = 2005895.14 microseconds
mean latency = 0.04 seconds = 40117.90 microseconds
median latency = 0.04 seconds = 40104.03 microseconds
maximum latency = 0.04 seconds = 41370.15 microseconds
minimum latency = 0.04 seconds = 39903.88 microseconds
first inference latency = 0.04 seconds = 41370.15 microseconds
first inference latency == maximum latency? True

mean ips = 24.93
median ips = 24.94

mean cpu usage = 25.1%
median cpu usage = 25.0%
maximum cpu usage = 29.4%
minimum cpu usage = 23.5%

mean ram usage = 19.3%
median ram usage = 19.3%
maximum ram usage = 19.3%
minimum ram usage = 19.3%

// Overall RAM Percent Utilization Metrics // -----
system ram: 2.1GB
ram used at script start 0.4GB, 19.0%
ram used at script end 0.4GB, 19.3%
Δram for entire script: 0.0GB, 0.3%
root@pynq:/home/xilinx/u96v2_train_inference#
```



Improving algorithm with hardware acceleration

1. FPGA has great advantage in terms of flexibility
2. Massive parallelism comparing to CPU
3. Using existing framework 'PYNQ-BNN' from Xilinx
4. Binary weights and hardware acceleration
5. Pre-manipulated data to required forms and promising inferencing speed, with some drops in accuracy

Junjie Chen



Improved model by Hyperparameter tuning

- Decreased memory usage and latency by reduce the hidden layer.
- Improve the later on accuracy by finding the best epoch for our model using early stopping
- Improve the later on accuracy by find the best learning rate so far by use the grid search on the model
- Combine the data preprocess data to reduce the filter layer and pool layer that make the latency decrease
- Used grid search to find the optimizer, batch size, kernel size, strip size, activation function and node number of our model.

The Result come out with original latency of 23ms per data to 19ms per data, And current accuracy is 96.73%