Team Number: sddec20-01
Client: Rockwell Collins
Advisers: Akhilesh Tyagi
Team Email: sddec20-01@iastate.edu
Team Website: https://sddec20-01.sd.ece.iastate.edu

MACHINE LEARNING FOR PILOT BIOMETRICS

Team Members: Jianhang Liu -- Data Manipulation SME / Feng Lin -- Hardware SME / Xuewen Jiang -- Camera Interface SME / Xiuyuan Guo -- Algorithm SME / Sicheng Zeng -- Python SME / Junjie Chen --C code SME

Introduction:

Problem: Hypoxia, Fatigue, Strain Doubles would cause pilots to crush their airplane, huge losses economically and not to mention they put their life in danger.

Solution: The processor board with camera will embed into the pilot's helmet to do the open-closed eye detection (This part is already designed).

Our Goal: Machine Learning algorithm has been developed to detect the blink rate on pilots, our job is to improve the existing machine learning algorithm, in terms of accuracy, latency memory usage. The solution will work on both software and hardware perspective.

Requirements:

Functional: Our large functional requirements are to reduce the Latency, accuracy and memory consumption for the current design.

Reduced latency by 75%
Accuracy higher than 97%
Memory usage less than 30%

Non-functional:

Usability: The system shall be used by pilot to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use
Scalability: The system shall be robust enough to work with different pilot's eyes
Maintainability: The system shall be easily maintainable
For non-functional requirements, we don't have environmental requirements.

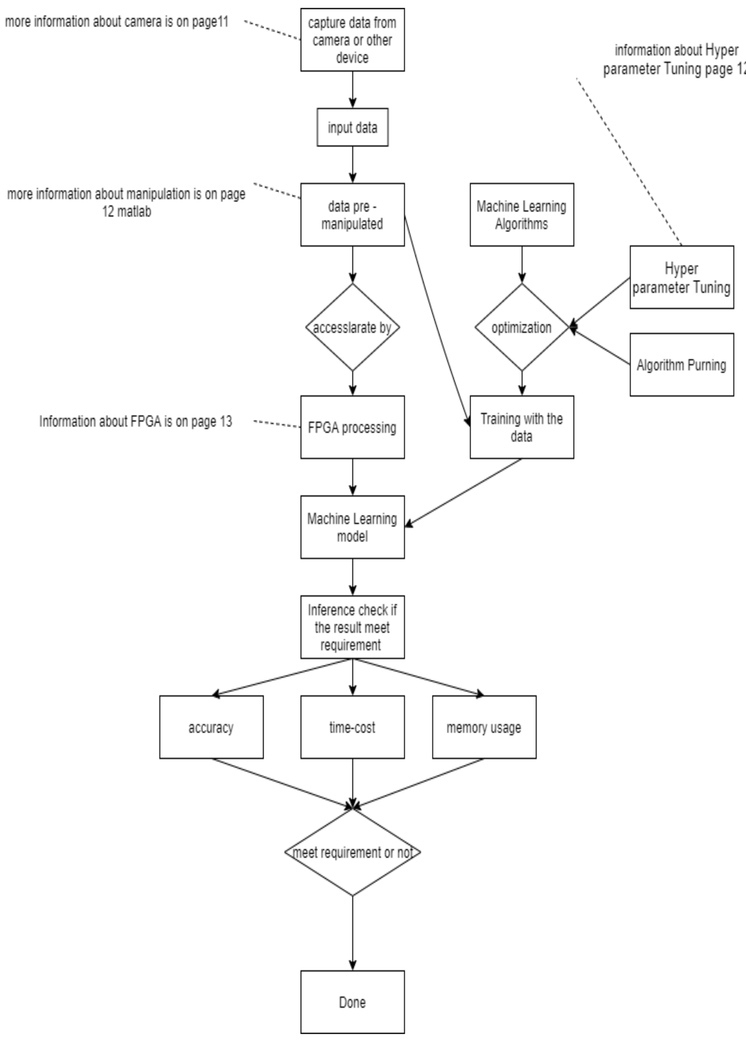
Operating environment: The system must work under the Xilinx FPGA board.

Intended users and uses

Pilot who needs fatigue checking when flying aircrafts. However, we are the middle part of the whole large project. In our perspective, we are working for our client's request, and the client will use our design to improve the end product. So, we don't need to think about how to serve the military pilot into our project. We also don't need to think about the airspace usage thing like that.

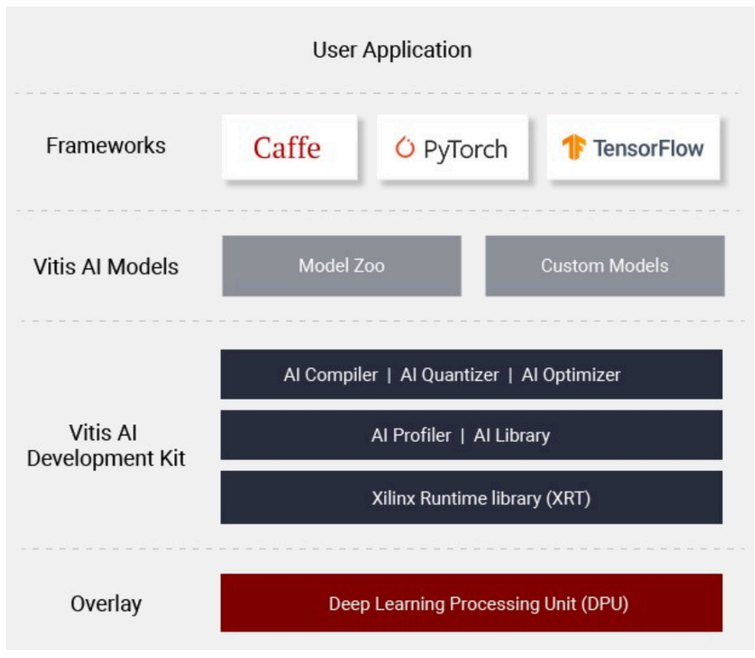
Design Approach:

Concept sketch:



This figure shows how our project works. First， we get data from a camera interface, we would use two cameras, one is a MIPI camera the other one is a USB camera. For the camera selection, one of our teammates did a grid search and we did a group trade study for several selections of camera. Then we would test which format of image is best for the ML algorithm. For example 28x28 black and white image, 36x36 RGB image, 28x28 overexposure image, edge detection image. for this step we are still in a search stage, we are still looking for other image manipulation methods that could work best for our ML algorithm, for know we got the data like memory usage latency and accuracy for 28x28 black and white image, 36x36 RGB image, 28x28 overexposure image, edge detection image. The FPGA part, we use DPU(deep learning process unit) to do complex matrix math for the algorithm in order to improve the performance of the algorithm. For hyper-parameter, we did a lot of tests via different parameters for the algorithm such as layers in neural network, epoch, batch size, optimize algorithm. After finding the best parameters, we want a pruning for the algorithm. then we will test if the algorithm meets our metrics based on accuracy, latency and memory usage.

Block Diagram:



Testing:

Testing environment: The system will be tested on Ultra96-v2 hardware with petalinux operating system.
Testing strategy : Metrix on latency, memory usage and accuracy will be captured and compared with the original algorithm.

Technical Details:

Software language: python
framework: Xilinx vitis-ai, Tensorflow Keras, numpy
operating system: petalinux, ubuntu
Hardware Design tool: PCB123
Platform: Ultra96-v2 FPGA

Hardware Design:

Daughter Card for Ultra96 Schematic & layout

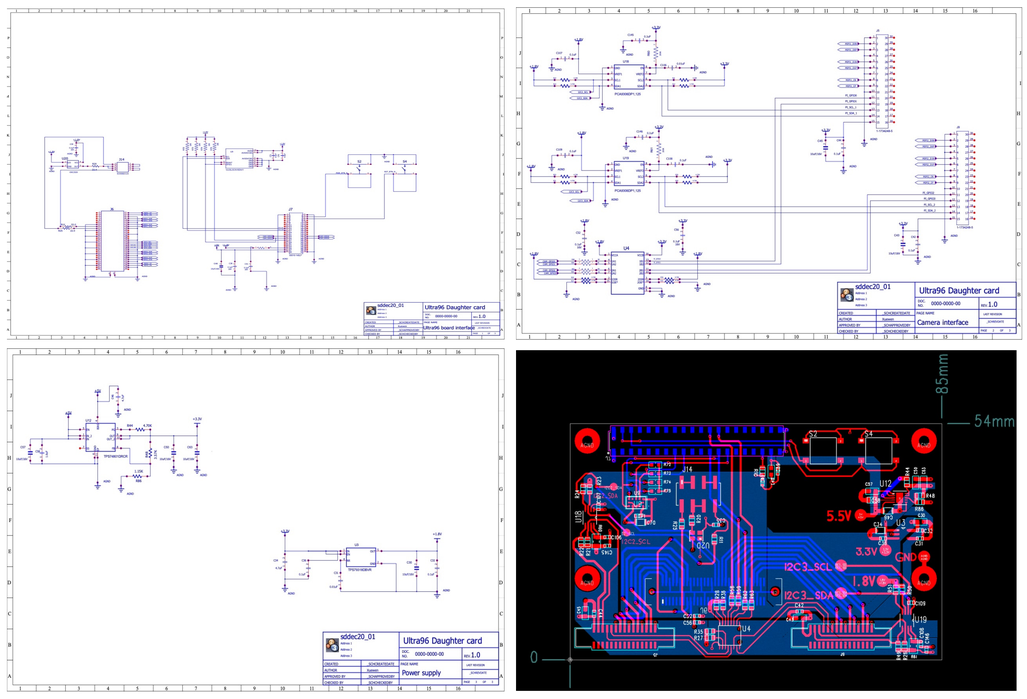


Image-premaputating filter result

externel Filter	Accurancy	Latency(ms)	Original Lantency(ms)	latency Improvement	Accurancy Decrease
sobel edge detection	76.56%	25.14	28.61	12.12%	21.44%
noise filtering	53.12%	24.44	28.61	14.58%	44.88%
over exposed	95.31%	24.70	28.61	13.65%	2.69%
gabor 20-0	84.38%	26.27	28.61	8.19%	13.62%
gabor 20-45	75.00%	24.95	28.61	12.78%	23.00%
gabor 20-90	79.69%	25.30	28.61	11.58%	18.31%
gabor 20-135	79.68%	25.84	28.61	9.67%	18.32%
gabor 50-0	75.00%	27.84	28.61	2.68%	23.00%
gabor 50-45	89.06%	29.09	28.61	-1.69%	8.94%
gabor 50-90	73.43%	27.16	28.61	5.08%	24.57%
gabor 50-135	89.06%	31.69	28.61	-10.76%	8.94%
gabor 100-0	60.93%	29.78	28.61	-4.10%	37.07%
gabor 100-45	70.32%	29.31	28.61	-2.46%	27.68%
gabor 100-90	70.31%	29.89	28.61	-4.48%	27.69%
gabor 100-135	71.88%	25.69	28.61	10.21%	26.12%
clahe	90.62%	25.48	28.61	10.92%	7.38%