Project Proposal: YOLO (You Only Look Once) in Hardware

Marc Gepigon, Swetha Pillai, Min Jian Yang

1 Hypothesis

With our implementation of YOLO in hardware, we will run inferences on images with respect to the class label of objects in the image along with their respective positions as outlined by bounding boxes. As a part of our analytical study, we will compare the performance of our YOLO model on hardware to previous software implementations of the YOLO model as well as other detection methods, using mean average precision (mAP) as the evaluation metric. Our classifier will be faster than its software counterparts solely because it is built on hardware, given the quantization of data for image compression, but at the cost of final accuracy. However, this is a reasonable trade-off. Although accuracy may not be high, our model will be able to predict class probabilities and bounding boxes with speed and precision across a full image.

2 Abstract

This project will cover the implementation, analysis, and demonstration of the You Only Look Once (YOLO) model. Primarily used for object detection, we will be building this model ground-up on the Snickerdoodle FPGA development board.

To design the ETL pipeline, we will use Xilinx's PYNQ library to interface with the FPGA using the Python environment. We will use the PASCAL Visual Object Classes (VOC) dataset for validation. Images will be fed in a compatible format from the PYNQ environment into the Snickerdoodle FPGA. The YOLO model on hardware will then run object detection on this image. There exists a trained model built previously – we will use the weights of this model to construct our model on the FPGA.

The model boils down to a convolutional neural network, with 24 convolution layers leading into two fully connected layers. The final layer consists of a leaky rectified linear activation function. Preventative measures will be taken to curb overfitting. The model will output the class prediction and its probability for that prediction. It will also provide bounding boxes for the detected objects, specfically the coordinate of the top left corner with its associated length and width.

With this data now sent back to the PYNQ environment, we can use it to overlay the image with such visual markers and display it. If time permits, we would also record a live video and pass it through the network to demonstrate the processing power. This entire process will be accurate because YOLO applies global reasoning on the full image. It will also be quick for two reasons: (1) YOLO combines all elements of an otherwise disjoint pipeline into a single, compact network, and (2) the network will be hardware-accelerated since it exists on our Snickerdoodle FPGA.

3 Timeline

Week #	Description
8	look into models' architectures and how to extract weights from them (software side metrics, (CPU / GPU speed))
9	start building architecture on hardware (conv layers, fully connected); debugging and putting everything together by end of the week; develop / enable different precision (number representation) and parameter schemes
10	verify w/ different dataset, organize results and find performance metrics on hardware end

4 Task Distribution

Tasks will be evenly distributed.

5 Expected Challenges & Solutions

A few challenges we anticipate on the hardware side include which Vivado IP to use and finding how much memory/resources/components we need to use. On the PYNQ side, we might find it difficult to determine how to feed in the images properly for hardware utilization and after the model runs, how exactly we can quantify its performance, whether it be with mean average precision, class accuracy, IOU, etc.

Potential solutions to these issues can be found in the lecture material, research papers, and other online resources. If all else fails, we will implement a simple CNN and carry out the rest as mentioned.