Object Detection

Project I

David Nichols, BSCpE, BSEE Student
John Smith, BSCpE Student

ECGR 4433
Machine Learning for the Internet of Things

April 5, 2023

Introduction

Project I is focused on object detection, specifically, detecting stop sign vs not stop sign. The motivation behind this project is to familiarize and put into practice training a binary classification model and deploy that model onto an embedded board. Project I will use the Arduino Nano 33 BLE connected to the OV7675 Arduino camera supplied in the Tiny Machine Learning Kit to achieve this goal. The dataset used will be a concatenation of ten self derived images and an online dataset found through Kaggle.

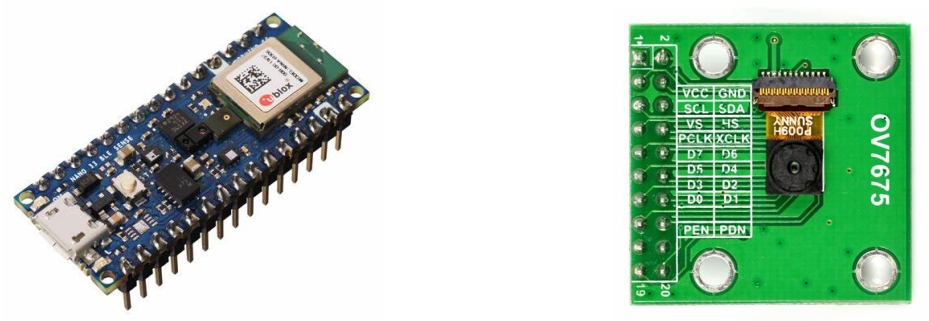


Figure 1: Arduino Nano 33 BLE and OV7675 Camera Module

In the following sections, various aspects of the project will be discussed in detail to include model architecture and data and training where the project setup and execution will be discussed. Following the model discussion, the results and conclusions drawn will be discussed.

Model Architecture

The topology for this project was derived by removing and adding layers until the most accurate model on validation data was achieved. Originally, the size of the network was constrained by the number of parameters. By increasing the number of parameters, the model was stopped at 300k parameters; an arbitrary number set by the team. The input for this projects model was greyscale images resized to 96x96 pixels. This was chosen due to the existing dataset containing mostly 96x96 images in greyscale, therefore standardizing the input.

Experimenting with other models, mobilenetv1 was explored. The issue with this model occurred when adding dropout layers, the accuracy decreased dramatically to fifty-three percent on validation accuracy while having a much higher training accuracy; a classic sign over drastic over-fitting.

Additionally, the team experimented with multiple models including: softmax vs sigmoid activation layers, same model with differing learning rates, and 70k to 108k parameters to name a few.

The model architecture for the model used in this project can be seen below in Figure 2 below.

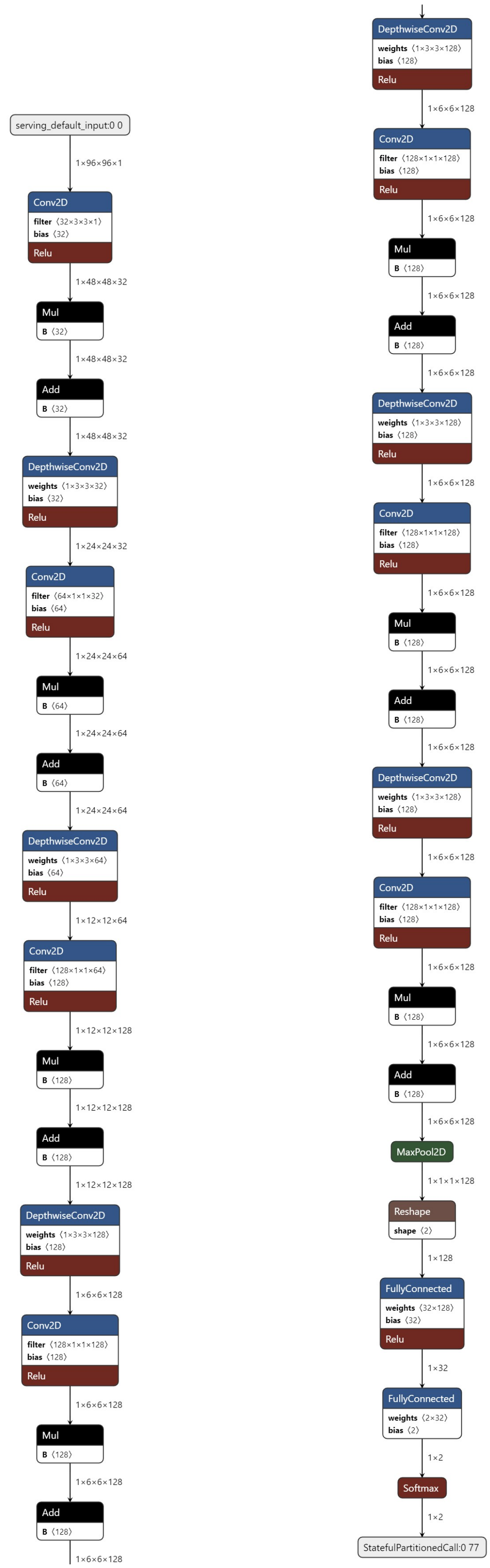


Figure 2: Model Architecture

Data and Training

For the teams custom data the team gathered pictures of a stop sign at multiple angles, having multiple buildings in the background. In addition to the stop sign, a similar sign found at the end of EPICs parking lot was photographed and used. These photos are shown in Figure 3 below. For a total of 5 stop sign images and 5 not stop sign images.

((a)) Stop Sign I

((b)) Stop Sign II

((c)) Stop Sign III

Figure 3: Stop Signs

In addition to the stop sign images, different sign images were taken that were similar to the shade and shape of a stop sign. This was done to push the boundaries of our model and supply the training with various self derived images of not-stop signs. Some examples of these images can be seen in Figure 4 below.

((a)) Not Stop Sign I

((b)) Not Stop Sign II

((c)) Not Stop Sign III

Figure 4: Not Stop Signs

Once the above images were augmented and reshaped, the output can be seen in Figure **??** below.

The team collected the images at the same time of day with an iPhone **JOHN FIX THIS**. We did not pool our resources with any classmates.

((a)) Stop Sign II

((b)) Stop Sign II

((c)) Stop Sign III

((d)) Not Stop Sign I

((e)) Not Stop Sign II

((f)) Not Stop Sign III

Figure 5: Augmented Stop Sign Images

Next the team found a dataset on Kaggle of road signs. These dataset was saved into the `Stop_sign` directory. The teams custom dataset was saved into the `customdataset` directory. Then in the `CreateDataset` notebook, all images where copied into to `Stop_sign` directory, and augmented. Each image was augmented 6 times this was chosen to get the total stop sign vs not stop sign number of images closer together. Each image had a range of contrast, and brightness it would be augmented too. This was 0.5 to 1.5 for both brightness and contrast. This was chosen due to any lower values causing the image to just be black, and any higher values causing it to be washed out. Also the images were all cropped so only the stop sign was in view and some of the background, then bounding boxes were used to make sure the stop sign or not stop sign was fully in the image.

The `Dataset` was split up into not stop, and stop folders. Then using train test split, split into training, testing and validation. The models was trained on the training and testing, then evaluated on validation. The team tested multiple different version of the model, from 72k parameters to 126k parameters, the 90k parameters model was chosen due to the ??? accuracy and anything higher started to overfit. These models were not chosen due to their overall size and heavy inference time compared to the small gains in accuracy.
vspace2mm

The team did not use any specific techniques like transfer learning, distillation or synthetic data generation. The team did experiment with a version of mobilenetv1 between 221k parameters and 300k parameters. The issue was once the team added dropout layers to the model, the models accuracy was only around 50 percent. The team did not experiment with the amount of data needed, after augmentation the team had just under 2000 images.

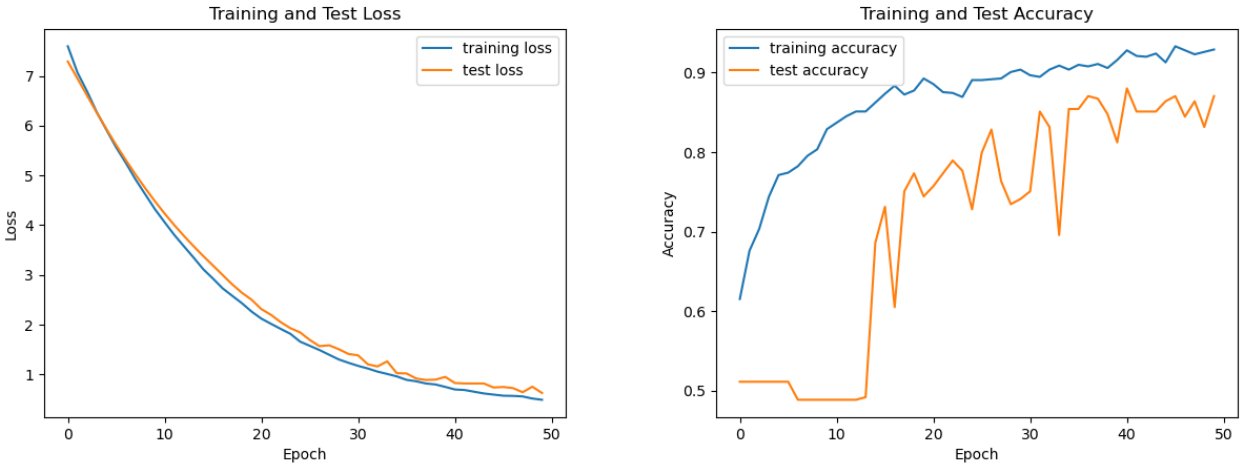
Figure 6: Augmented Stop Signs

Results

Table 1: Project I Results Summary

Metric	Measurement
Training Accuracy	87.06%
Validation Accuracy	90.32%
Test Accuracy	87.05
False Rejection Rate	0.15%
False Positive Rate	0.04%
Number of Parameters	90,018
Number of MACs	180,036 MACs
Input Tensor Shape	96x96x1

The model does respond to the target. The model also has false positives, specifically any red color sign and a dark area or sign. This is due to the augmented stop signs containing varying brightness levels versus the not stop sign images being all bright. The model also has to be close to the stop signs, due to it being trained on the cropped images. The inference takes 2.397737 seconds to run.



((a)) Loss Graph

((b)) Accuracy Graph

Figure 7: Number of Images

Discussion

Overall, this project came with many challenges. Some were easy fixes, some obstacles were never fully overcome.

The model has good performance with an inference time of 2.397737 seconds. It seems like a usable system even with the distance limitation. The team would like to improve it by training it on farther away images, and implementing bounding boxes instead of a binary classifier. Also by making the not stop and stop images more similar by having them both be centered around their signs and varying contrast and brightness.

Initially, the team attempted to use the all ops resolver, but this caused the model to become unresponsive and hang. The issue was identified when a person detection model was tested and encountered the same problem. By removing

the dropout and batch normalization layer and testing whether the model could be invoked, it was discovered that the batch normalization layer was causing the problem. The team then reintroduced the dropout layer and attempted to resolve the issue by adding mul and add resolvers for the batch normalization layer. After multiple days of troubleshooting, it was determined that the use of the all ops resolver, which had 128 operations, was the root cause of the issue.

The team would have spent less time trying to get the all ops resolver to work and done the above approach to find the missing ops. The team also would have used the given code for tflite instead of trying to follow the documentation.

Conclusion

The model starts at a very high loss value and takes to around epoch 25 to start learning, once that happens the model starts to overfit at epoch 50 so that is where the training is stopped at. The model slowly learned to a accuracy of 87%. This model was chosen due to having the most consist accuracy and loss values after epoch 30, and not overfitting the training data. The model and the quantized tflite model was tested with a not stop and stop image in python and correctly labeled the images. It was also tested on the nano where it correctly labeled the images if it is very close to the sign.

The team learned that a bigger model does not always mean its a better model, as the team increased the number of parameters to be over 90k, the accuracy slowly declines or the model began to over fit. Overall the team learned a lot about building binary classifiers and bounding box detection models. The team opted for a binary classifier due to the imbalance in the road signs dataset. This made augmenting and creating new images easier since they do not need bounding boxes for the binary classifier.

References

The below references are clickable hyperlinks to each reference.

GitHub Repo
Arduino Library
Kaggle