Project Final Report

David Liu

Ira Yago

Kaity Colbert

Work History

Date	Description	Author
05/02/2020	• Created a template for the project proposal • Wrote down several different sections that made up the whole proposal: Table of Contents, Introduction, Project Milestones, and Work Distribution	• David Liu • Ira Yago • Joy Lodra • Kaity Colbert
05/04/2020	• Updated the whole project descriptions as we changed to another project	• David Liu • Ira Yago • Joy Lodra • Kaity Colbert
5/16/2020	• We annotated 500 different pictures for pedestrians, vehicles, and sign posts using VoTT.	• David Liu • Ira Yago • Kaity Colbert
5/23/2020	• We create a pytorch file that holds the weights of our classifier using google collab and tensorflow. • Attempts to access and convert the PT file to Darknet marked it as “corrupted” using the ultralytics library. We hit a dead end.	• David Liu • Ira Yago • Kaity Colbert
5/30/2020	• Since we were low on time, we combined our annotations w/ a library of pre-annotated images of each object we were training (~3,000 images). We trained it over the span of 3 days with batch size 4 and 300 epochs. • Attempts to access the file and convert the checkpoint into a weights file using tensorflow model was also met with file corruption. (Stack Overflow confirmed this)	• David Liu • Ira Yago • Kaity Colbert
6/03/2020	• I trained it again using a pre-trained model but with 50 epochs and batch size 2	• David Liu • Ira Yago

	• It was a success and the pre-trained weights file grew in size.	• Kaity Colbert

Table of Contents

Introduction

Given a lack of knowledge between CARLA and the idea of machine learning itself, we're going to try our hand in implementing object detection with machine learning for 3 various objects found within CARLA.

Project Milestones

- May 11th: Set up the machine learning object detection for each individual object
    - David Liu: Pedestrians
    - Ira Yago: Vehicle recognition
    - Kaity Colbert: Road signs
- May 18th: Obtain a viable labeled dataset
- May 20th: Project Status Report
- May 27th: Create a preliminary machine learning model
- June 6th: Refine the final model
- June 8th: Project Final Report

Work Distribution

The team consists of three people. We decided to assign each member to work on different objects to detect, and at the end of the completion, we are going to compare and contrast each of the different results we have worked on throughout the quarter.

Project Status Report - 5/16/2020

We've looked through several guides and examples for training YoloV3 and annotated 500 images with the following tags using VoTT: Pedestrians, Vehicles, and Sign posts. Going forward, we want to learn more about how ML works in combining our annotations into a working preliminary model by the upcoming milestone on May 27th. This will be accomplished by using a mix of online web searching, watching tutorials, and doing the coding using Python and the Keras library.

Additional Project Milestones:

- May 21st - Research how YoloV3 training works and find out how it's implemented.
    - Find out if we can partition the data we train it with
- May 27th - Create the preliminary model
    - Created weight file using Ultralytics library in Google Collab
- June 3rd - Refine model
    - Combine all annotations and images into one model

Final Status Report - 6/06/2020

Before investigating how we could create our machine learning model, we hand annotated images using Fatkun Google Extension and VoTT. Fatkun allowed us to install Google Images in bulk while we handpicked and edited each image using VoTT.

Figure 1

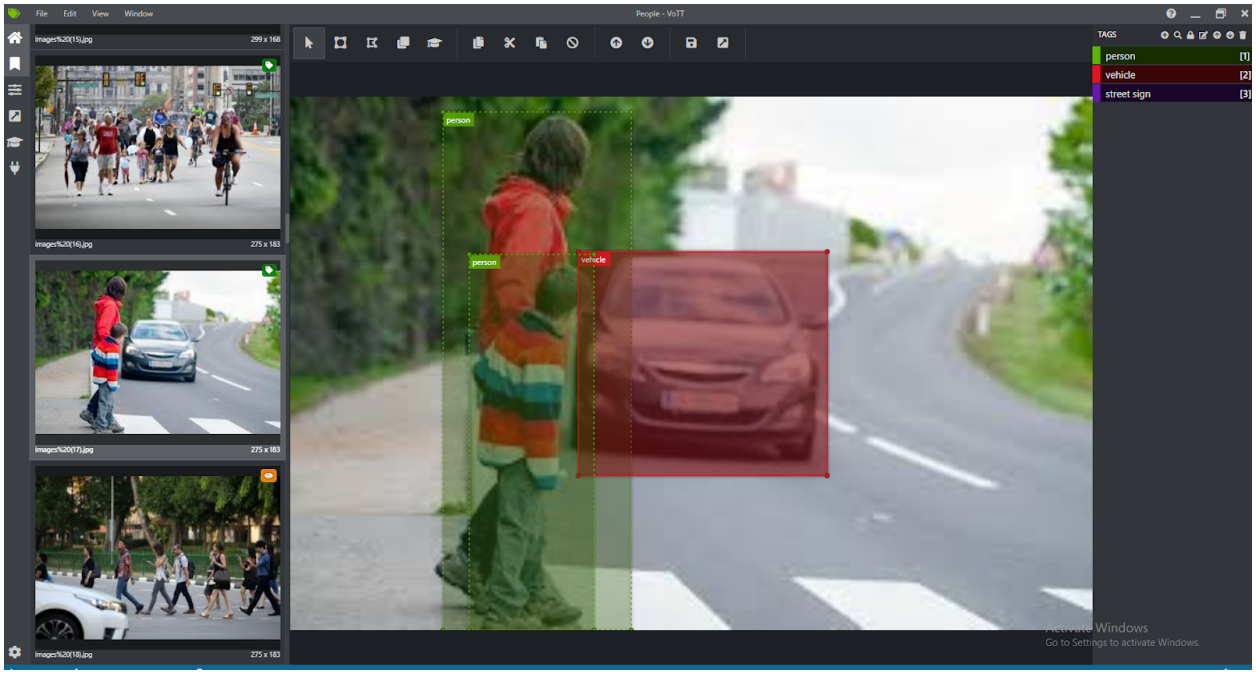

Figure 1 shows the classification process using VoTT.

VoTT allowed us to export our annotations as Pascal VOC and used the OID Tool kit library to translate the annotations to XML then to Darknet format (the one desired by YoloV3).

Our first method of creating a classifier for YoloV3 was using Google Collab and Tensorflow (5).

Figure 2

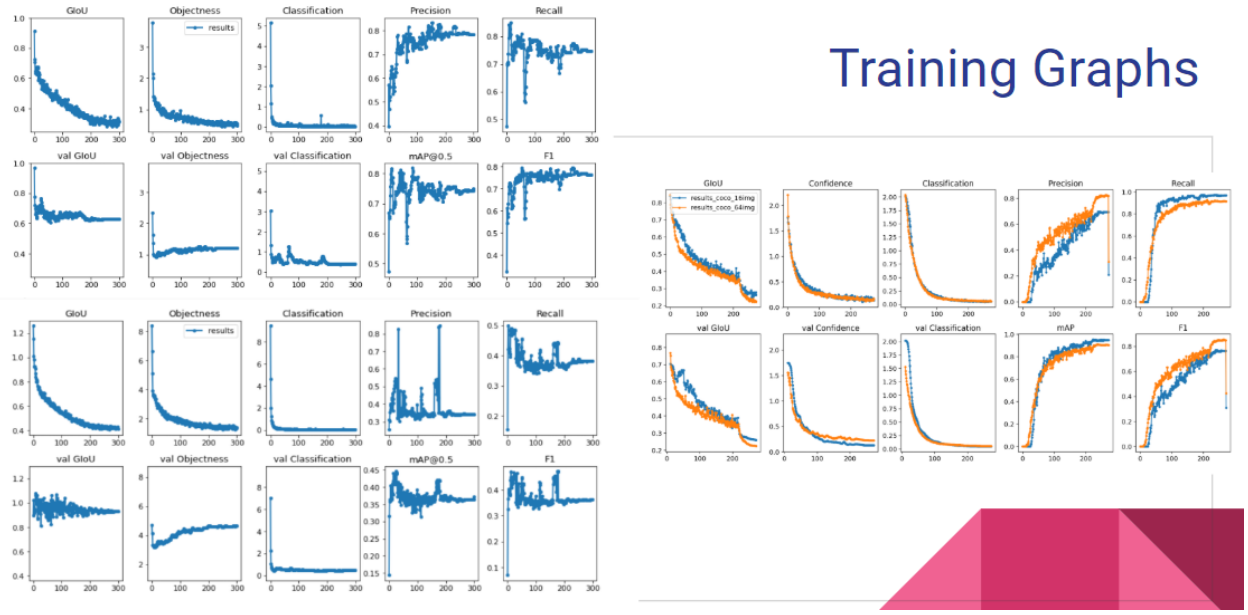

Figure 2 shows the graphs derived from the models created from each classification of our objects: vehicle and traffic sign.

This method of training our data was quick and easy as we were able to compile all of our data within Roboflows API to upload the training data to Google Collab. The graphs on the left hand side in Figure 2 shows the analysis of our models and the right hand side in Figure 2 shows the expected models. Unfortunately, the saved model within each of our Pytorch files were seemingly corrupted and I wasn't able to access them using the Ultralytics library which the code was derived from. The error codes were all discussing byte loss. By looking at our precision graphs, we can see that some of our results are inaccurate. We believe that is because of the consistency of our data and how long we trained it for. In terms of data consistency, we all had images that had varying factors, such as resolution, in them that could contribute to the noise added to our model. Also, each of us trained our models for a couple hours for around 300 epochs. However, we suspect that if we were able to combine our data successfully, it can increase our accuracy of detecting objects.

We then turned to use the OID Toolkit (1) and AlexeyAB (2) to create our YoloV3 model. The OID toolkit provided functionality to change our annotations and AlexeyAB was the Darknet for Windows which we were able to train on (3). Following the setup and instructions, we created a virtual environment for all the requirements and began training with over 3 thousand images, batch size of 4, and 300 epochs over the span of 3 days (4).

Figure 3

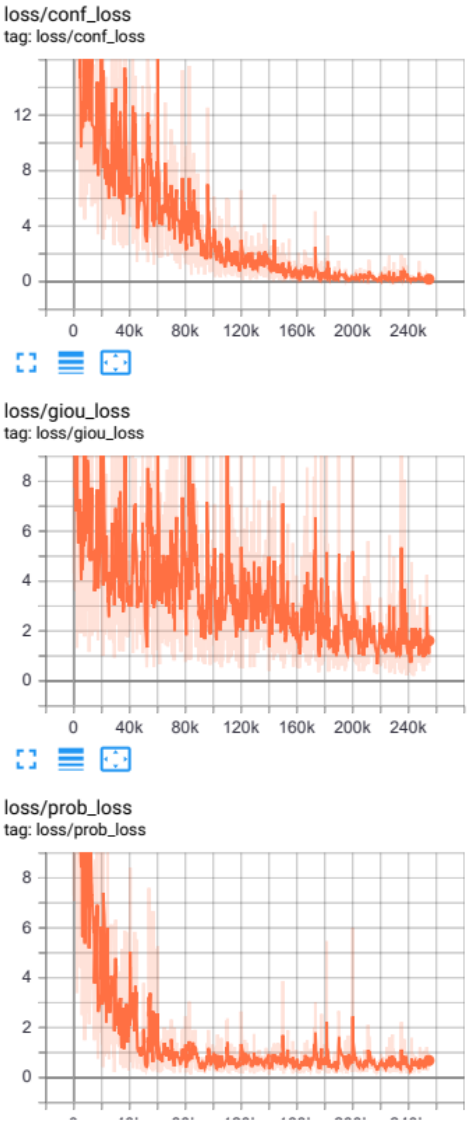

Figure 3 shows the type of loss (confidence, giou, prob, and total) in proportion to time. The X-Axis for prob_loss is in units of 40 thousand. Figure 1 displays the model with batch size 4 and over 300 epochs.

The X-axis of Figure 3 maps the steps (time) with respect to its value (Y-axis). We can see a clear downward trend close to 0 as the model nears 0 loss between validation and the trained model.

Figure 4

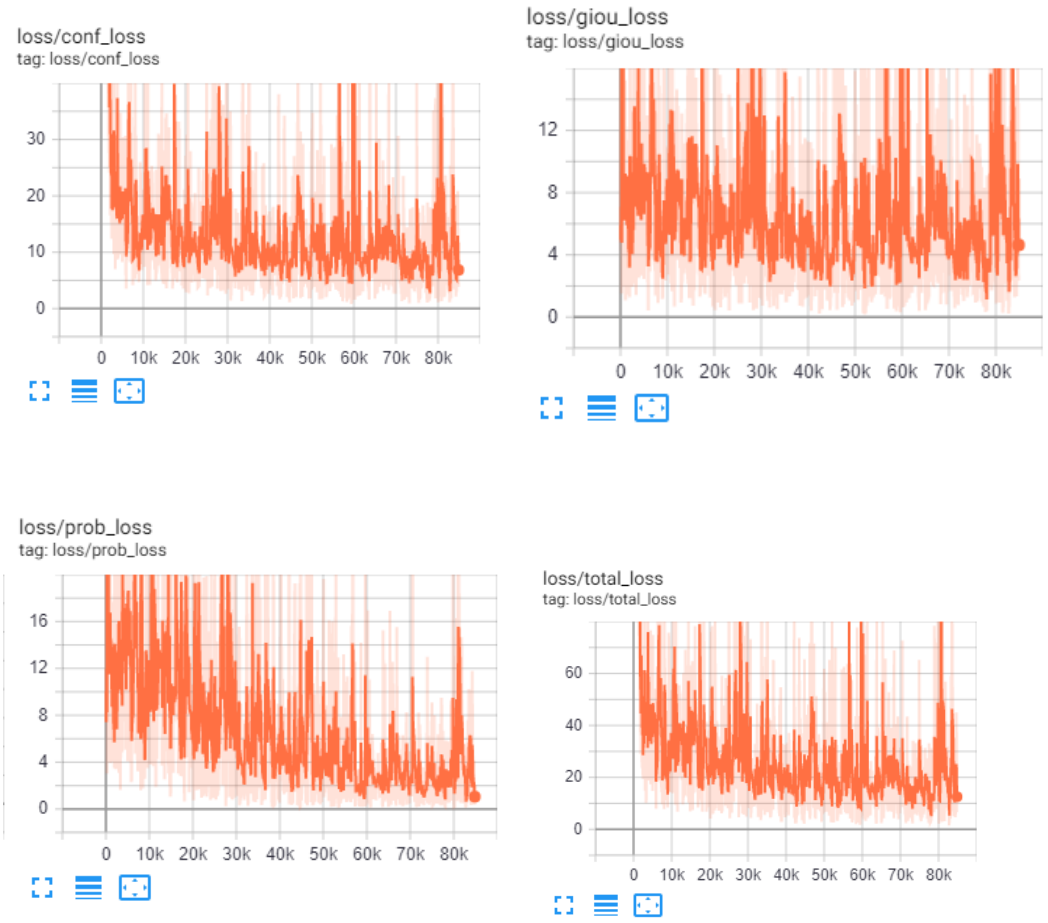

Figure 4 shows the type of loss (confidence, giou, prob, and total) in proportion to time. Figure 4 displays the model with batch size 2 and 50 epochs.

The X-axis of Figure 4 maps the steps (time) with respect to its value (Y-axis). We can see a volatile trend between the value of loss and the time spent to train the data. Despite all this, we managed to create a working model. We are able to compare and contrast the outcomes of training the model more intensively over a longer span of time vs. its shorter counterpart. Using tensorboard, we were able to see a relatively live analysis of the build of our model. We were also able to use code that would keep the best model as a checkpoint as it iterated through the epochs. This was evaluated as a summation between

confidence loss, giou loss, and probable loss. The model with the lowest sum would be our best model and would be saved so we wouldn't have to monitor the training 24/7.

Figure 5

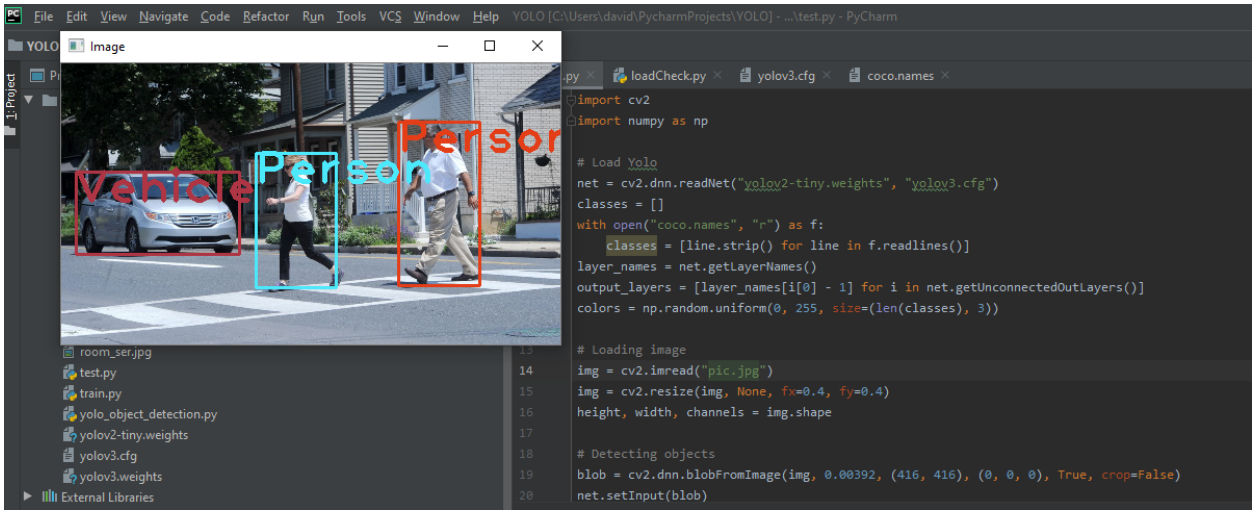

Figure 5 shows the type of loss (confidence, giou, prob, and total) in proportion to time. Figure 5 displays the model with batch size 2 and 50 epochs.

We were able to use the pre-trained modified weights of yolov3-tiny weights to classify images accurately. If we had more time, we would investigate strategies to solve the conflict of file corruption, and create a deeper analysis on the accuracy of our final model. This project taught us the intricacies as well as the work needed to create a successful machine learning model using YoloV3.

References

1. "Escvm/Oidv4_Toolkit". *Github*, 2020, https://github.com/EscVM/OIDv4_ToolKit. Accessed 6 June 2020.
2. "Alexeyab - Overview". *Github*, 2020, https://github.com/AlexeyAB. Accessed 6 June 2020.
3. Redmon, Joseph. "YOLO: Real-Time Object Detection". *Pjreddie.Com*, 2020, https://pjreddie.com/darknet/yolo/. Accessed 6 June 2020.
4. "Pythonlessons/Tensorflow-2.X-Yolov3". *Github*, 2020, https://github.com/pythonlessons/TensorFlow-2.x-YOLOv3. Accessed 6 June 2020.
5. "Releasing A New Yolov3 Implementation In Pytorch". *Roboflow Blog*, 2020, https://blog.roboflow.ai/releasing-a-new-yolov3-implementation/. Accessed 6 June 2020.