Autonomous Vehicle Initiative Project: Obstacle Detection System

Progress Report

Frank Dow 101140402
Joshua Gatto 101150890
Gilles Myny 101145477
Kareem El Assad 101107739

Coordinator: Richard Dansereau

Table of Contents

Glossary

Epochs: The number of times a training dataset is passed through a convolutional neural network.

Batch Size: The number of sample training images processed at a time through a convolutional neural network.

Continuous integration (CI): Automatically builds, runs test suites, and integrates the code changes within the Github Repository [11].
Continuous Development (CD): Automatically delivers the code changes to environments similar to production for approval [11].

Researched Topics & Lessons Learned

In the project proposal document submitted earlier this Fall 2022 term, Gilles researched a variety of neural network types, labelling methods, and masking objects on real-time object detection solutions. After a few weeks of project development and many hours of research and trial-and-error, new findings have been discovered and decisions made at the proposal phase have been rethought and/or scrapped in general for better, more functional design decisions.

Two types of neural network architectures were brought to the table, artificial neural networks (ANNs) and convolutional neural networks (CNNs). Although the group was leaning towards a CNN architecture, being the right choice, we never discussed the reason for this due to lack of knowledge and the project being so new to the group.

ANNs use multiple perceptrons per layer and go through a process called logistic regression where the input of the model is given a probability of an “event” taking place given that input. This ensures ANNs to be feed-forward neural networks because its inputs are only processed in this forward direction, passing the logistic regression step once per layer. This might seem like an advantage however, real-time object detection at video frame rate speeds requires quick processing time with smaller numbers of trainable parameters. ANN must convert 2-dimensional images into 1-dimensional vector pre-processing, this causes larger images to produce a number of trainable parameters on the order of 100,000 or more. Not only does this require more time to compute, time we don’t have, but it also causes the ANN to process each of these trainable parameters to lose their spacial features, meaning that trainable parameters, think of them as input pixels, are analyzed solely on their own, rather than assessing the rest of the image around them - a crucial feature in object detection. [1]

CNNs use a process called feature mapping, where an image fed into the neural network has multiple filters applied to gain a sense of object definition compared to surrounding objects, creating an outline of each object within the image. This is done via kernels, which extract this information and feed it into the neural network. One

drawback in terms of knowledge on the process is that CNNs use filters automatically without declaration, making it hard to follow and understand what is going on behind the scenes. Understanding how CNNs function is still pretty unclear within the industry however the positive results CNNs produce make up for the lack of understanding, generally. As discussed above, ANNs analyze each parameter uniquely, rather than considering the arrangement of pixels and the relationship they have to the image as a whole. Understanding the location of the parameter, and its relation to the image is done by parameter sharing. Where a single filter is applied across overlapping sections of the image, where this filter is then fed into the neural network, adding a sense of context to each trainable parameter. Not only does this method allow for a better understanding of the image - from the neural networks point of view - but it also drastically reduces the size of the trainable parameters, allowing for lower computing power and therefore shorter wait time between input and output. [1]

Taking this one step further, CNN is typically used for image classification - given an image containing one object, determine what that object is - however, with real-time object detection in the project's use case we require image detection - given an image containing any number of objects, determine what each object is. CNNs are followed by a fixed length connected layer that expects none or one bounded box within an image to decide on an output. However with image detection, the output layer must be variable, accounting for any number of occurrences with various spatial locations within the image. CNN could be used to analyze various regions of the image however this would be computationally infeasible. Therefore algorithms like YOLO (You Only Look Once) have been developed to find all occurrences while minimizing computational power. [2]

YOLO uses a single CNN to predict the bounding box locations and class probabilities for the image as a whole, based on the training dataset that has been provided. Creating a “heat map” of class probabilities within any given image. This method of image splitting based on a grid allows YOLO to operate at roughly 45 frames per second faster than other region based convolutional neural networks. The only drawback being the struggle to detect smaller objects due to its spatial constraints.

However, this is unlikely to cause any issues for the project scope as the object we are detecting are rather large and apparent, as is required with road signage, barricades, and cones, and typically the only potholes to worry about are those larger nasty sized which could inflict damage to the vehicle. [3]

The next section of Gilles' research was focused around which version of YOLO to proceed with for this project. As of writing this document there are seven main versions, each improving on the previous, with a few exceptions, but each getting more computationally intensive. The two contenders were YOLOv5 and YOLOv7 as they are the most impressive for real-time object detection at higher framerates, average precision, and image inference, as can be seen in the figure below.

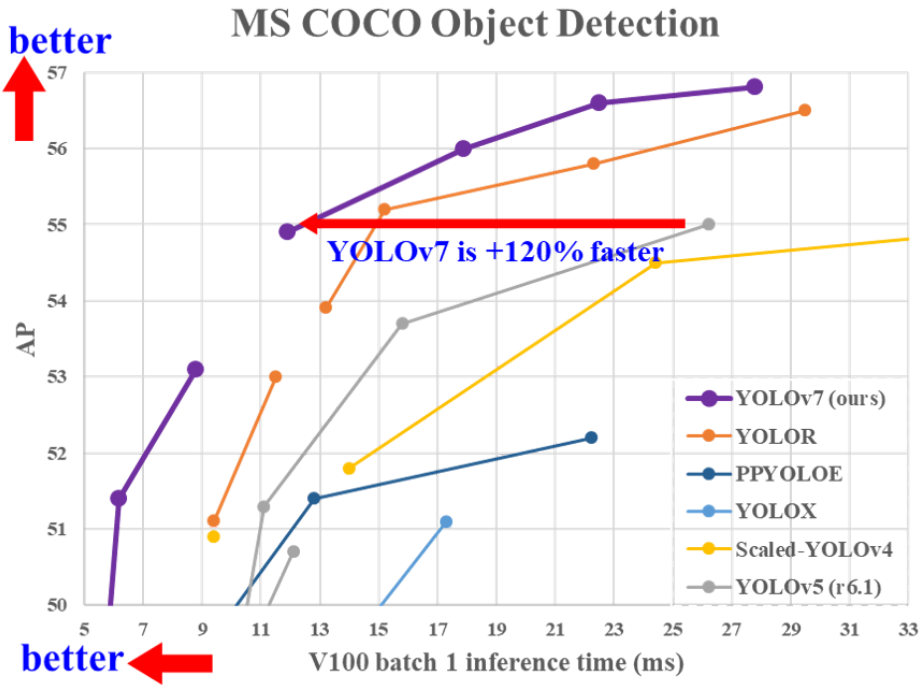

Figure 1: YOLO version comparison [4]

Choosing which YOLO version to proceed with all came down to selecting the appropriate key performance indicator(s) (KPIs). YOLOv5 favours a KPI related to training time between each YOLO CNN, due to its lower computational power requirements. On the other hand, YOLOv7 favours KPIs related to accuracy metrics such as precision, recall, and mean average precision at 0.5 obstruction (mAP @ 0.5),

as well as frames per second metrics. [5] For this project the group decided that during the testing phase a KPI related to time to train would be more favourable as we would be producing many YOLO CNN models while learning, fine-tuning, and expanding our dataset, however, as an end product a KPI more suitable would be related to accuracy metrics and real-world real-time performance. Given our limited time frame we decided to stick with YOLOv7 for the entire project development phase rather than relearning a new YOLO version later in the project phase. As seen in the figures below, YOLOv7 outperforms similar YOLO versions for accuracy metrics, however will nearly double the time to train on a 50 epoch run training run.

After 50 iterations of training	YOLOv5	YOLOv6	YOLOv7
Precision	0.982	0.951	1
Recall	0.98	0.805	0.982
mAP@0.5	0.974	0.9514	0.985
mAP@0.5 :0.95	0.854	0.768	0.916

	YOLOv5	YOLOv6	YOLOv7
Time to Train	7min 48s	14 min 2s	16min 40s
Model Parameters (Million)	20.87 M	17.2 M	37.2 M

Figure 2 & 3: YOLO versions accuracy and time to train metrics [5]

Since YOLO covers all aspects of a CNN and not just the convolutional layers, the group decided as a whole that a YOLO model would best suit the requirements of the project [8]. A YOLO model would only need to be trained and not require any design of any convolutional or fully connected layers, allowing the group to focus on developing quality data sets for input.

During the testing phase it was discovered that the number of epochs iterated increased the accuracy. On a small data set of approximately 100 images, increasing the number of epochs from 50 to 300 affected the accuracy by +50% on average. Research on the topic indicates that the optimal number of epochs is dependent on the size of the data set trained on [9].

In the experimentation phase, Joshua discovered that increasing the size of the dataset is the most effective method of increasing accuracy. As previously mentioned, increasing the number of epochs increases the accuracy to a certain extent; however, when increasing the number of images in the data set, unparalleled increases in accuracy were observed. This finding is supported by the fact that companies and organizations involved in the development of AI prioritize data set size and often have upwards of 10,000 images for a single class of object [10].

Frank narrowed down the available online resources for datasets and introduced Roboflow to the group. Roboflow has a variety of uses for vision detection, but the most important use for the group was the creation of datasets. Roboflow allows for the manual annotation of uploaded pictures or videos. Roboflow also gives access to Roboflow Universe, which is an open-source library of over 100,000 already labelled datasets. Once a collection of images is gathered, Roboflow can be used to organize data into 3 categories, training data, validation data and testing data. Once a dataset is created and organized on Roboflow there is also a health check feature which can check the average image size and ensure every image has an annotation. This feature also provides a balance of classes if you are trying to detect multiple objects. An example of this can be shown in figure 4. Where the person and head classes are being underrepresented. This feature will be useful when combining all our different objects into one dataset.

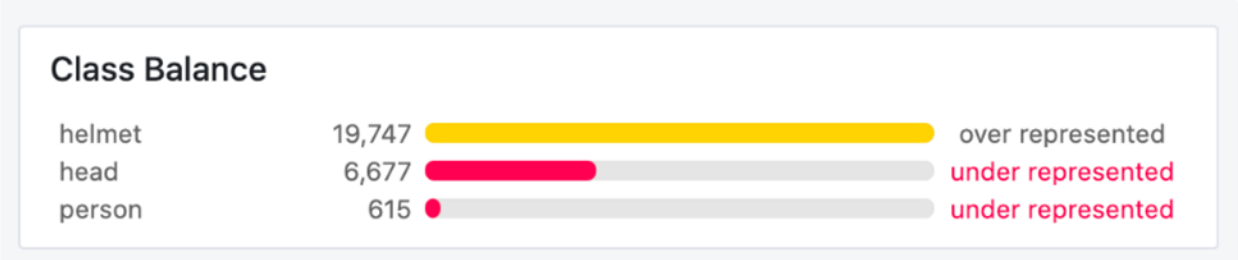

Figure 4: Results from Roboflow health check [6].

Before finalizing the dataset Roboflow provides two last steps called preprocessing and augmentation. In preprocessing it gives the option to resize all the images to one size or not. The augmentation step can allow you to create new images based on your current dataset. It can create image level augmentations which include rotating the images or changing the RGB values. There was also bounding box level augmentations where it can apply similar changes to only the object in the bounding boxes. These steps have been left out for the groups initial separated datasets but can be added in the future if needed. Once the Roboflow dataset has been generated, Roboflow gives you the ability to create a model using less than 1000 images or export it to a specific format. Since the group has decided to use YOLOv7, the dataset can be exported to a specific YOLOv7 PyTorch code fragment with an API key for that specific dataset. Since there were limited resources for the group to train models at the time, Google Colab was used to train the initial models.

Google Colab was a great online resource for experimenting with training models until the group members each reached the maximum of Google Colab's allowable GPU usage. Thankfully, the group now has access to the Maker Space PC which will now be used for training models. The main variables that can be changed during training on Google Colab were batch sizes and epochs. The test results are then displayed directly on Google Colab showing the objects detected in the test images along with their accuracies. Further discussions on each member's models will be discussed in the Individual Progress section.

While the team was researching CNNs, Kareem was creating Jupyter notebooks to help and support the research conducted. Kareem was able to use tensorflow and a basic sequential model to identify airplanes, automobiles, birds, cats, deer, and trucks at a 68% accuracy. While this is not particularly useful for our use case, it gave the rest of the group a clear starting point and helped kick start the project.

While researching YOLO, Kareem examined a large number of loosely related machine learning projects. Kareem was able to set up the repository and the architecture of the system for future use. The architecture model chosen uses a modified strategy based pattern. This allowed the team to create scripts that could easily be shared and organized for later use.

The repository currently has this structure:

+---references
+---snippets
+---src
  +---data
  a +---tensorflow
  a +---yolo
  +---models
  +---visualization

Each team member commits their changes to the repository in their own branch. The branches are then compared and reviewed before being merged into the main branch of the repository.

Update on Funding Process & Equipment

Shortly after the project proposal document was submitted, the group had to communicate and team up with the other groups within the Autonomous Vehicle Initiative project to brainstorm, create, and publish a funding request to CUESEF for all the equipment the larger group would be needing for this project. Our group in particular will be using various pieces of equipment requested by other groups towards the end of the project timeline, however we requested for $400 worth of Google Cloud VM credit which would allow the team to run trial-and-error neural network trainings on our prototype datasets to determine what version of YOLO would be best, what configurations we can feed YOLOv7 for desired results, and what changes need to be made to our prototype image datasets to ensure success in the future. The funding request was a success and the larger group got funding approval for all the equipment we requested. Unfortunately this funding request was placed before we were given access to the Maker Space PC, therefore this $400 funding will most likely not see use by our group unless we find the need to, and can also be used by the other groups in the Autonomous Vehicle Initiative if need be.

Earlier on in the term, the group was informed of the possibility that a powerful PC could be made available in the Maker Space in Mackenzie Building. After discussion with Professor Dansereau and back-and-forth email conversations with the lab manager Patrick Fairs, the group managed to setup a powerful PC with a Nvidia RTX 3080 8GB VRAM and 64GB RAM to run Ubuntu 22.04.1 LTS and be both physically accessible via the Maker Space and via Remote Desktop. This will most likely eliminate the need to spend the funding received by CUESEF towards Google Cloud VM credit as this allows the group to perform trial-and-error YOLO model training with ease, without worrying about VM usage and costs.

The group also sent in an order request under the 500$ funding allowance per group to purchase necessary test equipment which will see extensive use in the weeks to come. A Raspberry Pi 4 Model B, Raspberry Pi HQ Camera, 16mm Telephoto Lens,

Tripod, and Mounting Plate were purchased for roughly $450 and were picked up successfully.

Changed Aspects of the Proposal

In the original proposal, it was assumed that the image data sets would be labeled manually where bounds of an object would be specified in a config file then overlaid onto an output image using a custom developed program. The project has moved away from this idea and incorporated the use of a tool called Roboflow. Roboflow has greatly increased the productivity of the group as images can be tagged in a fraction of the time it would otherwise take.

The project proposal discussed developing a CNN from the ground up. After further research into the specifications of CNNs, it was discovered that a CNN would not be satisfactory for the project requirements. The time to process a single image for a CNN would be too slow for the real time environment that the system is being designed for. The development of a CNN was also deemed to be too complex for a project of this size and scope. Developing an RCNN was also explored; however, similar issues came to light. As a result, the project now employs YOLOv7 for object detection.

The project will no longer make use of a google cloud VM for the training of the neural network. Instead, a workstation in the maker space has been made available to be used locally through a remote session. Unfettered access to this PC allows the group to continuously train improved versions of the model rather than being limited by lack of funds for VM credits.

Milestone Updates & Changes

Milestone 1 has been slightly altered by changing the testing of several different neural networks to everyone creating different YOLOv7 models, as this was decided to be the algorithm best suited for the project. All members made YOLOv7 models using datasets of different objects that are found while driving. Objects were assigned as follows, stop signs and traffic lights to Gilles, Potholes to Frank, cones to Joshua, and barricades to Kareem. This was divided up so that all members could get a good understanding of YOLO and labeling or sourcing images for a dataset. The group is now in the final stages of Milestone 1 where they have discussed all the results from experimentation and are now looking for ways to improve accuracy.

Milestone 2 will have a significant change. For this milestone, each member will attempt to annotate 1000 images for their specific objects. Also, the group will need to research whether to proceed with their separate trained YOLOv7 models or whether to create a combined dataset of all classes and create a new model. If it is discovered that the group can use transfer learning for our models, this will be helpful so each member can proceed to work on their separate models to later be combined. If a combined dataset must be used, there must be class balance between all objects. It is noted that the goal of 1,000 annotated images may not be possible for all classes. The deadline for each member to have a dataset of 1,000 images annotated will be January 9th, 2023. After the database/databases is/are complete there will still be a second part to Milestone 2 which will be the training and testing of a model based either based on a combined dataset or a model created from transfer learning. This goal will remain in line with Milestone 2's previous date January 14th, 2023.

For now, Milestone's 3 and 4 will remain as is. However, the group may experiment with their individually trained models on the Raspberry Pi.

Individual Progress - Sprint Board

On top of the research work done by Gilles, we individually tracked larger work items with tickets on a Sprint Board. These tickets for Gilles represent topics of work done with minimal deviation done from the listed topic. The figure below shows all tickets completed, under review, in progress, and planned up to date of this document. The titles of each ticket are high level descriptive to allow a project manager to understand the general scope of work being done during this period.

Aa Ticket	Priority ...	Status	Epic	Sprint	Assigned	Time Created	Ticket Finish Date
002 - Discord Server Creation	Low	Complete	Project Setup	Sprint 1	Gilles Myny	September 10, 2022 1:01 PM	September 12, 2022
003 - Research Neural Network	Low	Complete	Research	Sprint 1	Gilles Myny	September 10, 2022 1:08 PM	September 14, 2022
004 - Research Hardware for Project	Low	Complete	Research	Sprint 1	Gilles Myny	September 10, 2022 1:08 PM	September 14, 2022
007 - Create Google Drive for Training Pictures	Low	Complete	Project Setup	Sprint 1	Gilles Myny	September 15, 2022 5:28 PM	September 16, 2022
010 - Proposal - Risk & Mitigation Strategies	Medium	Complete	Project Proposal Document	Sprint 1	Gilles Myny	September 22, 2022 4:50 PM	October 21, 2022
011 - Proposal - Components & Facilities Required	Medium	Complete	Project Proposal Document	Sprint 1	Gilles Myny	September 22, 2022 4:52 PM	October 21, 2022
012 - Proposal - Applicable Knowledge Section	Medium	Complete	Project Proposal Document	Sprint 1	Gilles Myny	September 22, 2022 4:53 PM	October 21, 2022
013 - Decide on Type of CNN Architecture	High	Complete	Research	Sprint 2	Gilles Myny	November 13, 2022 9:05 PM	November 17, 2022
014 - Decide on Apprioriate YOLO version	Medium	Complete	Research	Sprint 3	Gilles Myny	November 13, 2022 9:04 PM	November 17, 2022
015 - Source Image Sets for Signs	High	Complete	Trial Dataset	Sprint 3 Sprint 4	Gilles Myny	November 13, 2022 9:08 PM	November 26, 2022
016 - Label Signs Images for Trial Dataset	High	Complete	Trial Dataset	Sprint 3	Gilles Myny	November 13, 2022 9:09 PM	November 23, 2022
019 - Email Lab Manager for Maker Space PC Inquiry	Medium	Complete	Miscellaneous	Sprint 4	Gilles Myny	November 21, 2022 4:43 PM	November 28, 2022
021 - Submit Funding Request for Pi Testing Equipment - PURCH4907-21	Medium	Complete	Project Setup	Sprint 4	Gilles Myny	November 21, 2022 4:44 PM	November 21, 2022
027 - Setup Maker Space PC for Training	Medium	Complete	Maker Space PC	Sprint 5	Gilles Myny	December 3, 2022 1:44 PM	December 3, 2022
028 - Transform Stop Sign Classes into Polygonal Bounded Boxes	Medium	Blocked / Review	Trial Dataset	Sprint 5	Gilles Myny	December 3, 2022 1:45 PM	
029 - Progress - Researched Topics	Medium	In Progress	Progress Report Document	Sprint 5	Gilles Myny	December 3, 2022 1:46 PM	
030 - Progress - Individual (Sprint)	Medium	In Progress	Progress Report Document	Sprint 5	Gilles Myny	December 3, 2022 1:47 PM	
031 - Request Maker PC Password Change & Ethernet Upgrade	High	Not Started	Maker Space PC	Sprint 5	Gilles Myny	December 3, 2022 1:48 PM	

Figure 5: Sprint Board for Gilles

The key takeaways from tickets/work completed to date for Gilles are some administrative tasks - such as discord server creation, funding requests, and communications with the lab manager - project proposal document work, research on neural network architectures, YOLO version comparison, and his personal road signs image dataset creation and labelling, and running YOLOv7 training scripts on said image datasets for comparison.

The current image dataset for road signs within a trial phase consists of four classes: stop signs, do not enter signage, red lights, and orange lights, as these are most important regarding the vehicles trajectory halting during travel. This will be expanded later on in the project timeline to include more warning signs and green lights.

The image dataset consists of 695 images with 805 bounded boxes (annotations). As of writing this document and tests performed to date the over-representation of stop signs and under representation of all other classes is apparent in the accuracy metrics when training the YOLOv7 model. This will be supplemented over time to ensure a fair distribution of images from all four classes as this will affect the bias in the learning pattern of the neural network.

As discussed in the research done by Gilles, the class probability uses a “heat map” of where objects have been annotated in the training dataset. The figure below shows the current heat map of the road signage dataset, as expected most road signs will be seen towards the center of the vehicles point of view, as would be expected with road signage to increase the chances of it being seen by a driver.

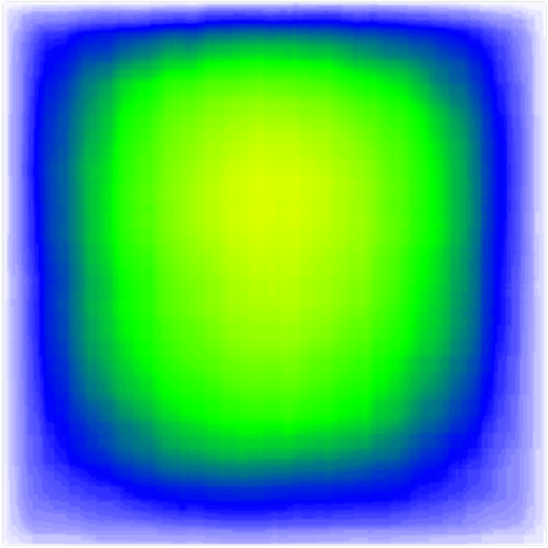

Figure 6: Heat Map for Road Signs Dataset

To date, the YOLOv7 training has been rather successful in terms of accuracy metrics for true positive (identifying an object, and doing it correctly) detection. The figure below shows a great example of the current difficulties being ironed out by Gilles on his image dataset. This sample was taken from a training run at 300 epochs and a

batch size of 8.

Figure 7: Testing Image from Road Signs YOLOv7

This image is quite large in terms of resolution so zooming in may be required. However, you will see that the true positive detection of the Do Not Enter sign has an accuracy of 92% which is impressive and up to par with an accurate and confident neural network, given that the sign is on an angle too, dispersing its spacial properties within the image. However, there are three false positive detections in this image that are being addressed by fine tuning the training method and contents of the image dataset. Firstly, up to the left of the Do Not Enter sign the neural network has detected a stop sign with 10% accuracy. Furthermore, right of the sign there is another false positive for a stop sign with 36% accuracy. Lastly, a red traffic light has been detected below the previous stop sign detection with an accuracy of 18%. Although YOLO CNN architectures are hard to understand, this is most likely due to the red leaves with lighter shades which could seem like a stop sign to a neural network which has been trained on an incomplete image dataset, however in the real world there are many false

positives which are ignored due to their low accuracy levels such as the previously mentioned numbers.

Many alterations and additions will need to be made to this image dataset to minimize false positives and increase accuracy of true positives further. This can be done by eliminating the over and under representation of specific classes, increasing the size of the image dataset to allow for the neural network to gain more exposure to images during training, and by hand selecting real-world dash cam videos. These changes will be covered in the next section of this document.

Since the project proposal document submission, Joshua has worked towards developing a YOLO V7 model to identify three (3) types of cones in any given image. The figure below shows what tasks have been completed by Joshua.

Ticket	Priority ...	Status	Epic	Sprint	Assigned	Time Created	Ticket Finish Date
003 - Research Neural Network	Low	Complete 🙌	Research	Sprint 1	Gilles Myn	September 10, 2022 1:08 PM	September 14, 2022
005 - Research Code Libraries OPEN	Low	Complete 🙌	Research	Sprint 1	Kareem As	September 10, 2022 1:09 PM	
012 - Proposal - Applicable Knowledge Section	Medium	Complete 🙌	Project Proposal Document	Sprint 1	Gilles Myn	September 22, 2022 4:53 PM	October 21, 2022
013 - Build Cone Data Set	High	Complete 🙌	Trial Dataset	Sprint 2	Joshua	December 5, 2022 3:10 PM	November 17, 2022
013 - Build Cone Data Set	High	Complete 🙌	Trial Dataset	Sprint 3	Joshua	December 5, 2022 3:08 PM	November 17, 2022
022 - Source Image Sets for Cones	High	Complete 🙌	Trial Dataset	Sprint 3 Sprint 4	Joshua	November 21, 2022 4:46 PM	
029 - Progress - Researched Topics	Medium	Complete 🙌	Progress Report Document	Sprint 5	Gilles Myn	December 3, 2022 1:46 PM	
030 - Progress - Individual (Sprint)	Medium	In Progress	Progress Report Document	Sprint 5	Gilles Myn	December 3, 2022 1:47 PM	
031 - Supplement Cone Data Set	High	Complete 🙌	Trial Dataset	Sprint 5	Joshua	December 5, 2022 3:08 PM	December 2, 2022
031 - Supplement Cone Data Set	High	Complete 🙌	Trial Dataset	Sprint 4	Joshua	December 5, 2022 3:04 PM	December 2, 2022
032 - Proposal - Methods Used	Medium	Complete 🙌	Project Proposal Document	Sprint 1 Sprint 2	Joshua	December 5, 2022 3:36 PM	September 23, 2022

Figure 8: Sprint Board for Joshua

The types or classes are cone, barrel and tube. These types were selected and deemed common enough to be found in everyday travel. Thus ample images for training material would be available online. For the first step, Joshua collected images from the internet containing traffic cones in a natural environment. Data sets were also sourced from publicly available libraries on the internet and Roboflow. Joshua then labeled the cone images using the Roboflow tool by selecting a bounding box region and assigning a class. The data sets were then combined and finalized into a single package.

Using Google Colab, several preliminary YOLO V7 models were trained on the data set. The models each varied in some parameters upon training such as epoch number and batch size. Each model had a varying degree of success detecting images accurately but the clear best models had the greatest number of epochs iterated.

After obtaining these results, Joshua went back and augmented the data set with more images. Training new models on this augmented data set demonstrated much greater increases when compared to increasing the epoch number. This observation

The work done by Frank since the proposal was mainly experimenting with a dataset of potholes in the Google Colab environment. The pothole dataset was obtained from the Roboflow Universe and had 1000 pre labelled pothole images. Frank had examined the dataset to make sure all potholes were properly labelled. The figure below shows the tasks delegated to Frank.

Aa Ticket	Priority ...	Status	Epic	Sprint	Assigned	Time Created	Ticket Finish Date
001 - Create Document & Embed in Notion	Low	Complete 🙌	Project Proposal Document	Sprint 1	F Frank	September 10, 2022 12:33 PM	October 7, 2022
008 - Proposal - Project Description	Medium	Complete 🙌	Project Proposal Document	Sprint 1	F Frank	September 22, 2022 3:58 PM	October 11, 2022
009 - Proposal - Background	Medium	Complete 🙌	Project Proposal Document	Sprint 1	F Frank	September 22, 2022 4:00 PM	October 14, 2022
012 - Proposal - Applicable Knowledge Section	Medium	Complete 🙌	Project Proposal Document	Sprint 1	F Frank G	September 22, 2022 4:53 PM	October 21, 2022
017 - Source Image Sets for Potholes	High	Complete 🙌	Trial Dataset	Sprint 3 Sprint 4	F Frank	November 14, 2022 2:01 PM	November 10, 2022
023 - Train Models on Pothole Dataset	Medium	Complete 🙌	Trial Dataset	Sprint 5	F Frank	December 1, 2022 11:13 AM	November 17, 2022
026 - Obtain Data Tables from Trained Models	High	In Progress	Trial Dataset	Sprint 5	F Frank	December 1, 2022 11:16 AM	
029 - Progress - Researched Topics	Medium	Complete 🙌	Progress Report Document	Sprint 5	F Frank G	December 3, 2022 1:46 PM	
030 - Progress - Individual (Sprint)	Medium	Complete 🙌	Progress Report Document	Sprint 5	F Frank G	December 3, 2022 1:47 PM	
035 - Progress - Milestone Changes	High	Complete 🙌	Progress Report Document	Sprint 5	F Frank	December 8, 2022 3:43 PM	December 7, 2022
036 - Research - Online Datasets	Medium	Complete 🙌	Research	Sprint 2	F Frank	December 8, 2022 3:44 PM	November 7, 2022

Figure 9: Sprint Board for Frank

Although there are many different types of potholes with varying sizes and depths, they were still all categorized under one pothole class. It is one task to detect potholes and another task to detect the size of the potholes. When trying to detect the size of a pothole in a moving vehicle the size of the pothole increases as the vehicle gets closer, so Frank had ditched the idea of classifying potholes by their sizes. Frank had trained four different models on Google Colab with varying batch sizes, and epochs. It was found that the more epochs the more accurate the detections. For

example, Figure 8 shows two identical photos where the one on the left was trained at 155 epochs and the one on the right was trained at 55 epochs with equal batch sizes.

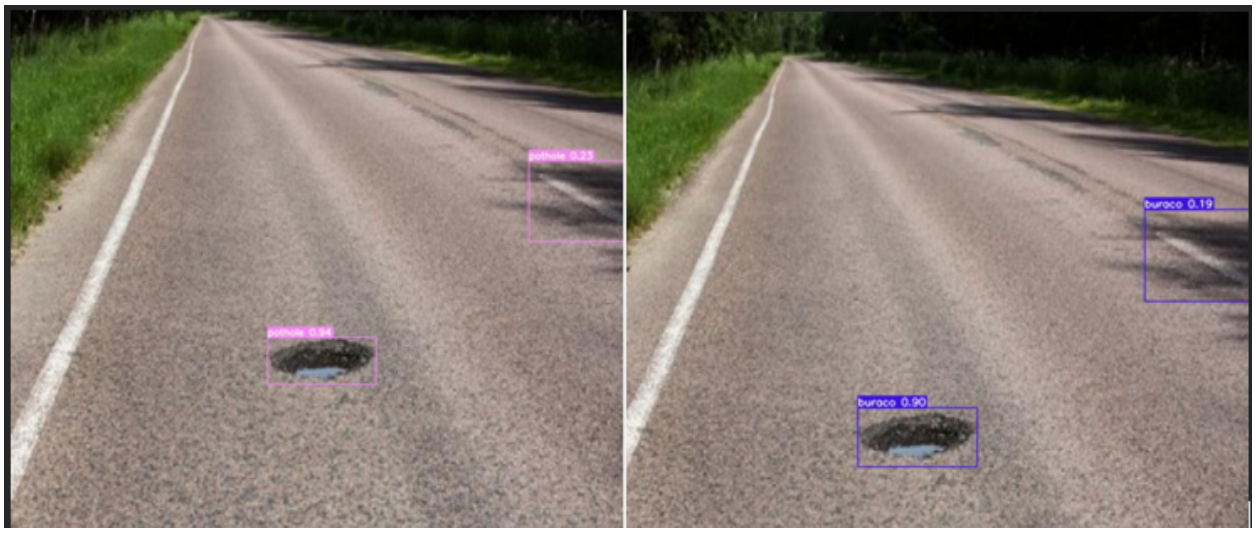

Figure 10: Testing images from Potholes YOLOv7

The results from the higher epoch have a 4% accuracy increase, however there is also an accuracy increase on the shadow in the photos. This was likely detected due to how potholes are slightly darker than the rest of the pavement and since shadows have a similar effect it will be a tricky problem to solve. There is also an assumption that puddles will most likely be perceived as potholes, as they too are generally darker than the rest of the road. The training of the YOLO model on the left took about an hour longer than training for the YOLO model on the right. Unfortunately, like other members of the group Frank had run out of GPU space on Google Colab and was unable to run a 300 epoch model, which is the maximum allowed on Google Colab. It would be interesting to see just how much more accurate the model could get, but time must also be considered during training. The 300 epoch model would have probably taken over 3 hours to train and if the accuracy only went up a percent or two, the increase in epochs might not be worth it at a certain point. This will be the next step for Frank to test on the new Maker Space PC, as well as start to test the pothole model on video footage from a dash cam recording.

Throughout the semester, Kareem researched a wide variety of approaches used in identifying and labeling barricades. Initially, the target was the standard traffic barricades typically used to block major intersections. Later, the team decided that barriers were too broad of a category and must be specified further.

Aa Ticket	Priority ...	Status	Epic	Sprint	Assigned	Time Created	Ticket Finish Date
001 - Create Document & Embed in Notion	Low	Complete	Project Proposal Document	Sprint 1	Kareem Assad	September 10, 2022 12:33 PM	October 7, 2022
003 - Research Neural Network	Low	Complete	Research	Sprint 1	Kareem Assad	September 10, 2022 1:08 PM	September 14, 2022
005 - Research Code Libraries	Low	Complete	Research	Sprint 1	Kareem Assad	September 10, 2022 1:09 PM	
006 - Setup Notion Workspace	Medium	Complete	Project Setup	Sprint 1	Kareem Assad	September 10, 2022 1:15 PM	
012 - Proposal - Applicable Knowledge Section	Medium	Complete	Project Proposal Document	Sprint 1	Kareem Assad	September 22, 2022 4:53 PM	October 21, 2022
020 - Source Image Sets for Barricades	High	Complete	Trial Dataset	Sprint 3 Sprint 4	Kareem Assad	November 21, 2022 4:46 PM	
024 - Setup Remote-SSH on Makerspace PC	Medium	Blocked / Review	Maker Space PC	Sprint 5	Kareem Assad	December 1, 2022 1:57 PM	
025 - Train Models on Barricades Datas OPEN	High	In Progress	Trial Dataset	Sprint 5	Kareem Assad	December 1, 2022 1:59 PM	
029 - Progress - Researched Topics	Medium	Complete	Progress Report Document	Sprint 5	Kareem Assad	December 3, 2022 1:46 PM	
030 - Progress - Individual (Sprint)	Medium	Complete	Progress Report Document	Sprint 5	Kareem Assad	December 3, 2022 1:47 PM	

Figure 11: Sprint board for Kareem

To more accurately represent them, four major categories of barricades were created. Those categories are Cable Barriers (a), Box Beams (b), Concrete Barriers (c), and Hybrids (d).

The distinction between the barriers was made after collecting over 1000 images of barricades. It became apparent that it was much more complicated to detect all the types of barriers as they were vastly different. To fix this issue, Kareem created four separate datasets and then trained them individually. The CNN was then capable of predicting and identifying the barricades at a much higher accuracy than before.

After splitting the datasets, Kareem ended up with four imbalanced datasets. Some categories such as Concrete Barriers and Box Beams were overrepresented in the dataset while Cable Barriers were notably underrepresented. The majority of machine learning techniques favor categories with large sample sizes [7]. For the classes to be comparable, they must be the same size [7]. To address this issue, the largest datasets were scaled down to the size of the smallest one.

The resolution of the images was scaled down significantly before training. This is because CNNs tend to pick up important information using layers of filters. Many of

the barriers detected have simple geometric characteristics that can easily be picked up using a simple 3x3 filter. To identify key horizontal features, a matrix of [1,1,1],[0,0,0],[-1,-1,-1] was applied [7]. Additionally, this had the added benefit of taking the focus away from the unrelated background. Features such as the sky, clouds, grass, and light rays were no longer accidentally identified.

Using Google Colab, several training models were created for the datasets using YOLOV7. The models had a varying degree of success detecting images based on the number of epochs and the batch size used. After thorough testing, it was identified that a batch size smaller than 16 was ideal for our use case. A batch size of size 16 or greater typically yielded minuscule better results at the cost of significantly longer training times.

Planned Progress for the Winter Term

In regards to the newly purchased Raspberry Pi equipment, the plans for this equipment are as follows. This equipment will be assembled, initialized, and tested in the coming weeks after final examinations. This will allow the group to record dash camera videos in the streets of Ottawa to allow for self-curated scenarios where the obstacles we are detecting can be seen in conditions controlled by the group. A handful of these videos will then be dissected frame by frame to annotate them in Roboflow for training purposes in our various datasets. Another use for the Raspberry Pi Camera equipment is also the recording of dash camera videos for testing purposes, where we can log and analyze what the neural network is doing while active to see where the group can make alterations in the datasets, training methodology, and code to allow for higher accuracy, lower false positives, and the potential to find new objects/classes we find to be useful in our real-time object detection project.

The team intends on reorganizing and refreshing the current Sprint Board used on Notion. We intend to add much stricter sprints and properly implement the Agile workflow. For the most part, we already followed the Agile principles within the group, but now that the responsibilities of each member are more clear, we feel as though a fresh restart is due. We intend to have weekly stand ups where each member is expected to have completed their weekly assigned tasks.

The team also intends on implementing CI/CD pipelines on Github to help avoid any loss of work. The Pipelines will push local changes on the Makerspace PC to the repository daily. This should allow us to accurately track the project progress and alleviate any stress regarding loss of progress.

Finally, after a discussion with Professor Danserau, the team intends on exploring various solutions for distance estimation. Some of the topics we intend to research are:

- Binocular Vision
- SIFT Algorithm

- Disparity Estimation for Distance & Disparity Maps
- Elevated LIDAR Mount
- Distance Finding AI

References

[1] A. Pai, “Ann vs CNN vs RNN: Types of neural networks,” *Analytics Vidhya*, 19-Oct-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/.

[2] R. Gandhi, “R-CNN, fast R-CNN, Faster R-CNN, YOLO - object detection algorithms,” *Medium*, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

[3] J. Nelson, “The yolo algorithm: A guide to yolo models,” *Roboflow Blog*, 18-Nov-2022. [Online]. Available: https://blog.roboflow.com/guide-to-yolo-models/.

[4] WongKinYiu, “Wongkinyiu/Yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” *GitHub*. [Online]. Available: https://github.com/WongKinYiu/yolov7.

[5] A. Banerjee, “Yolov5 vs Yolov6 vs yolov7,” *YOLOv5 vs YOLOv6 vs YOLOv7 - by Amitabha Banerjee*, 25-Jul-2022. [Online]. Available: https://www.learnwitharobot.com/p/yolov5-vs-yolov6-vs-yolov7.

[6] “Dataset Health Check,” *Roboflow,* 30-Nov-2021. [Online]. Available: https://docs.roboflow.com/dataset-health-check

[7] Rezapour, M.; Ksaibati, K. Convolutional Neural Network for Roadside Barriers Detection: Transfer Learning versus Non-Transfer Learning. Signals 2021, 2, 72–86. https://doi.org/10.3390/signals 2010007

[8] G. Yue, C. Liu, Y. Li, Y. Du, and S. Guo, “GPR Data Augmentation Methods by Incorporating Domain Knowledge,” Applied Sciences, vol. 12, no. 21, p. 10896, Oct. 2022, doi: 10.3390/app122110896.

[9] M. H. Junos, M. Dahari, S. Thannirmalai, and A. S. M. Khairuddin, “An optimized yolo‐based Object detection model ... - wiley online library,” doi.org, 18-Mar-2021. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ipr2.12181.

[10] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” arXiv.org, 06-Jul-2022. [Online]. Available: https://arxiv.org/abs/2207.02696.

[11] GitHub, “Continuous integration and continuous delivery (CI/CD) fundamentals,” *GitHub Resources*. [Online]. Available: https://resources.github.com/ci-cd/.