Independent Research Project Logbook

A Semi-trailer Truck Right-Hook Turn Blind Spot Alert System for Detecting Vulnerable Road Users with Transfer Learning

Charles Tang

Massachusetts Academy of Math and Science at WPI

Research Advisors: Mr. Nicholas Medeiros, Kevin Crowthers, Ph.D.

Table of Contents

Table of Contents **1**

GLP Record Keeping Contract **4**

Logbook Etiquette Guidelines **4**

Section I: Brainstorming Diagrams **5**
- Pie Diagram #1 5
- Pie Diagram #2 6
- Pie Diagram #3 7
- Mindmap #1 8
- Mindmap #2 9
- Mindmap #3 10
- Fishbone Diagram and Five Whys #1 11
- Fishbone Diagram and Five Whys #2 12
- Fishbone Diagram and Five Whys #3 13
- Colored Hat Activity 14
- Hobbies and Basic Ideas 15

Section II: Project Overview **16**
- Project Abstract 16
- Project Introduction 16

Section III: Professional Communication **17**
- Email #1 17
- Email #2 18
- Email #3 19
- Email #4 20
- Email #5 22
- Email #6 23

Section III: STEM BiWeekly Meeting Notes **24**

Section IV: Materials and Methods **28**
- Research Goals 28
- Criteria 28
- Constraints 28
- Materials List 28
- Procedure 28
- Testing 29
- Potential Roadblocks 29

Section V: Background **30**

Bicyclist Safety and Collisions 30
Blind Spots of Semi-Trailer Trucks 30
Bicyclist Infrastructure 31
Object Detection 31
Blind Spot Bicyclist Detection Systems 32
References 32

Section VI: Daily Entries **34**
Color Legend 34
Daily Entry #1: System Diagram 34
Daily Entry #2: Design Matrix 36
Daily Entry #3: TF Object Detection API 37
Daily Entry #4: Materials + Pipeline 44
Daily Entry #5: Train Model (N) 50
Daily Entry #6: Train Model (SOFTMAX) 52
Daily Entry #7: Continue Training 55
Daily Entry #8: Continue Training 58
Daily Entry #9: Graphic Design 61
Daily Entry #10: Finish Training 63
Daily Entry #11: Begin Training ResNet 71
Daily Entry #12: Continue Training 73
Daily Entry #13: Continue Training MobileNet 76
Daily Entry #14: Finish Training MobileNet 79
Daily Entry #15: V-Model 83
Daily Entry #16: ASIL 84
Daily Entry #17: Setup TF1 86
Daily Entry #18: Colab Training 87
Daily Entry #19: Further Colab Training 89
Daily Entry #20: Single-Class Dataset 91
Daily Entry #21: Train Single-Class Model 96
Daily Entry #22: Train Another Single-Class Detection 98
Daily Entry #23: Setup Coral Board 100
Daily Entry #24: Train Third Single-Class Detection 101
Daily Entry #25: Figure Creation 103
Daily Entry #26: Deploy Model 104
Daily Entry #27: Run Model on Video 105
Daily Entry #28: Run On Startup 108
Daily Entry #29: December Fair Design 110
Daily Entry #30: Prototyping 111
Daily Entry #31: Blind Zone Estimation 112
Daily Entry #32: December Fair Poster 114

Daily Entry #33: Cyclist Images 115
Daily Entry #34: New TFRecord 116
Daily Entry #35: Training Procedure 121
Daily Entry #36: Annotate Images. 122
Daily Entry #37: Continue Annotating 125
Daily Entry #38: New TFRecord 127
Daily Entry #39: CAD Sketch 130
Daily Entry #40: Train on New Dataset 131
Daily Entry #41: Deploy New Model 133
Daily Entry #42: Docker Environment 134
Daily Entry #43: CAD Design 136
Daily Entry #44: Train MobileNet TF2 137
Daily Entry #45: Test Models 138
Daily Entry #46: Test in Real-Time 142
Daily Entry #47: Object Tracking Design 145
Daily Entry #48: Implement SORT Algorithm 146
Daily Entry #49: Test Sort Algorithm (TBD) 152

GLP Record Keeping Contract

I, Charles Tang, commit to record keeping in accordance with Good Laboratory Practices.

- My experiments and records will be reproducible, traceable, and reliable.
- I will not write my notes on disposable items and will go into my laboratory notebook.
- My data will be recorded in real-time. If I cannot record data in real-time, I will record raw data as soon as physically possible.
- I will record both qualitative and quantitative observations in my laboratory notebook and laboratory reports.
- My laboratory notebook will include information on the materials and instruments utilized during experimentation.
- I will initial and date over the edge of any material that is taped into my laboratory notebook.
- I will provide a real-time record of any analysis I perform.
- I will define ALL abbreviations.
- If I make a mistake in my laboratory notebook, laboratory worksheets, or other written material, I will not obliterate or obscure the mistake. Instead, I will cross out the mistake using a single line. Any empty spaces in tables or partially used notebook pages will be crossed out using a single diagonal line.
- I will initial and date each page in my notebook and the front of each laboratory report.

Charles Tang

August 20, 2022

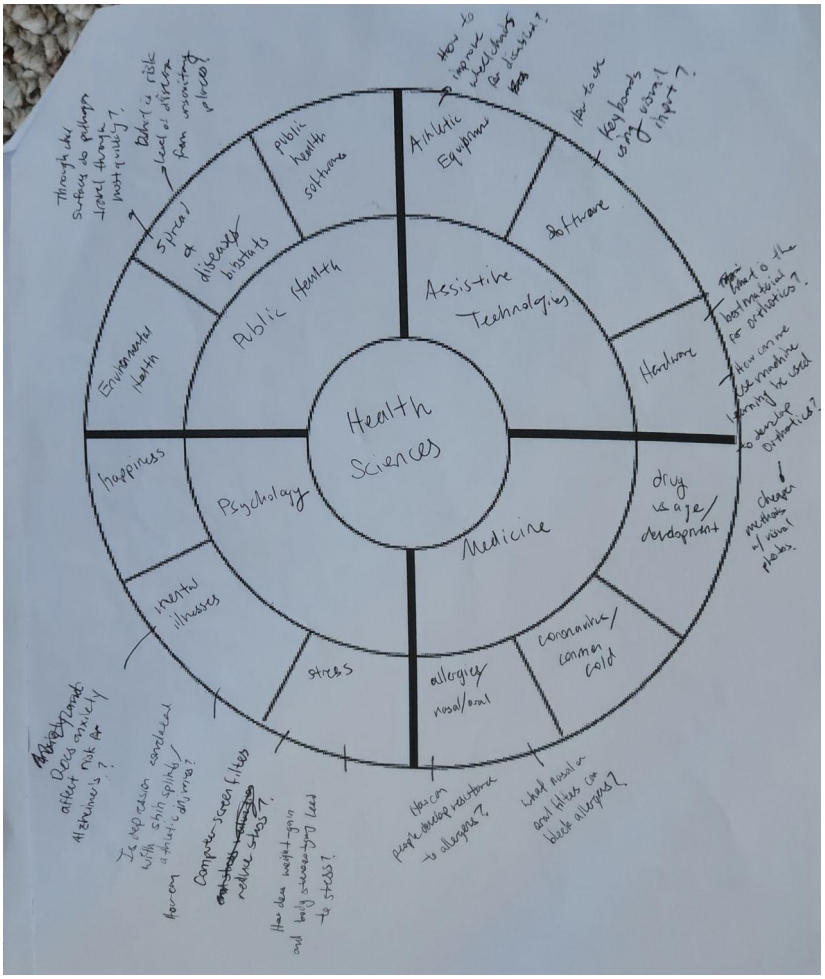
Logbook Etiquette Guidelines

1. Make an entry every time you work on your project
2. Make sure your entries are verified by a mentor/ teacher signature and your signature
3. Organize your notebook in the format.
    - A: Table of Contents
    - B: Brainstorming and Topic Ideas
    - C: Project Introduction
    - D: Communications.
    - E: Draft of Materials and Methods.
    - F: Background- ie. competitor/market analysis, criteria/constraints
    - G: Daily Entries (every time you complete work on the project)
        - 1: Title and Date
        - 2: Short Introduction (putting the experiment/observations into context/objectives)
        - 3: Methods/Materials (if not included in the beginning of the notebook)
        - Materials become important when someone needs to repeat your experiments
        - 4: Observations/Experimental Data (both RAW and ANALYZED)-
        - 5: Calculations and Data Analysis (STATISTICS)
        - 6: Final Concluding Remarks

Section I: Brainstorming Diagrams

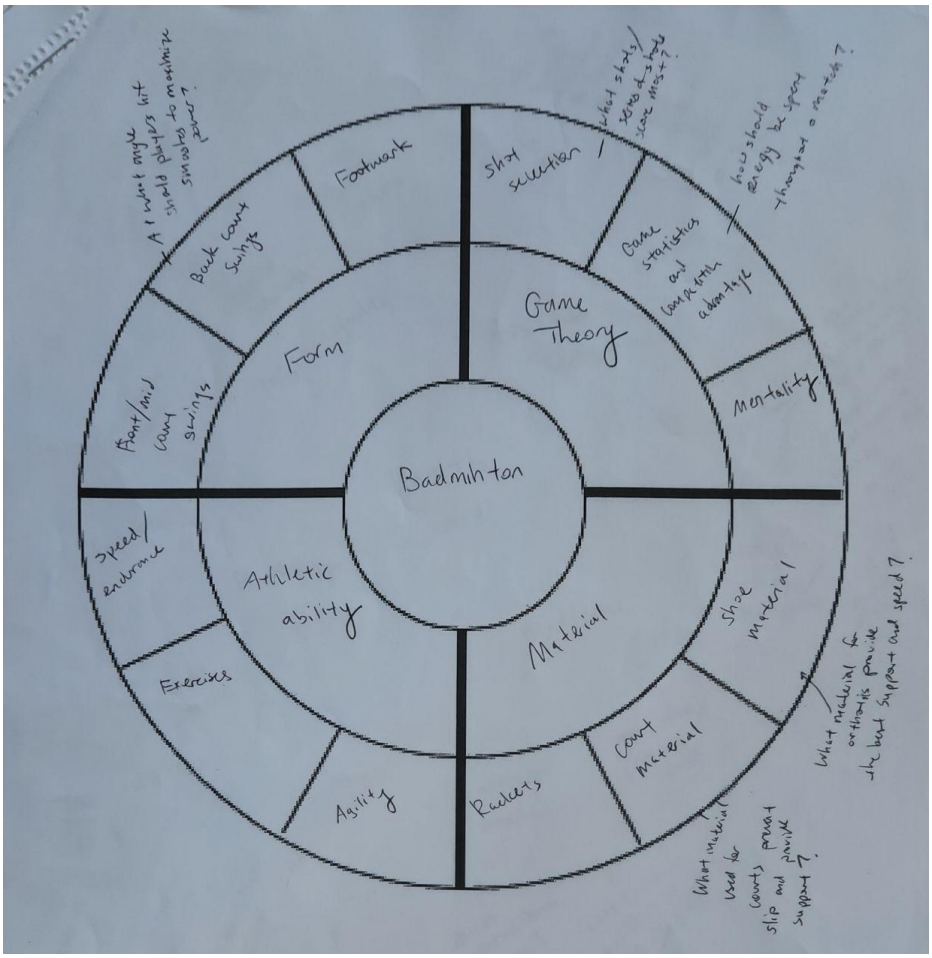
Pie Diagram #1

Date	August 18, 2022
Signature	*Charles Tang*,
Brief Description	The health sciences field is a field I have been interested in, especially during the COVID-19 pandemic. Since participating in past research on minimizing COVID-19 exposure in hallways, I have wanted to learn more about public health and ways to measure the spread of diseases throughout the human population. Assistive technologies is a field that seems like a big concern, especially in the sports industries with athletes. As someone with oral allergies, I have grown interested in medicines, the development of allergies, and the medicines that could treat these allergies. Lastly, the current mental health epidemic is of interest to me, and it would be beneficial to learn more about how mental illnesses develop and some of the risk factors that associate with them.



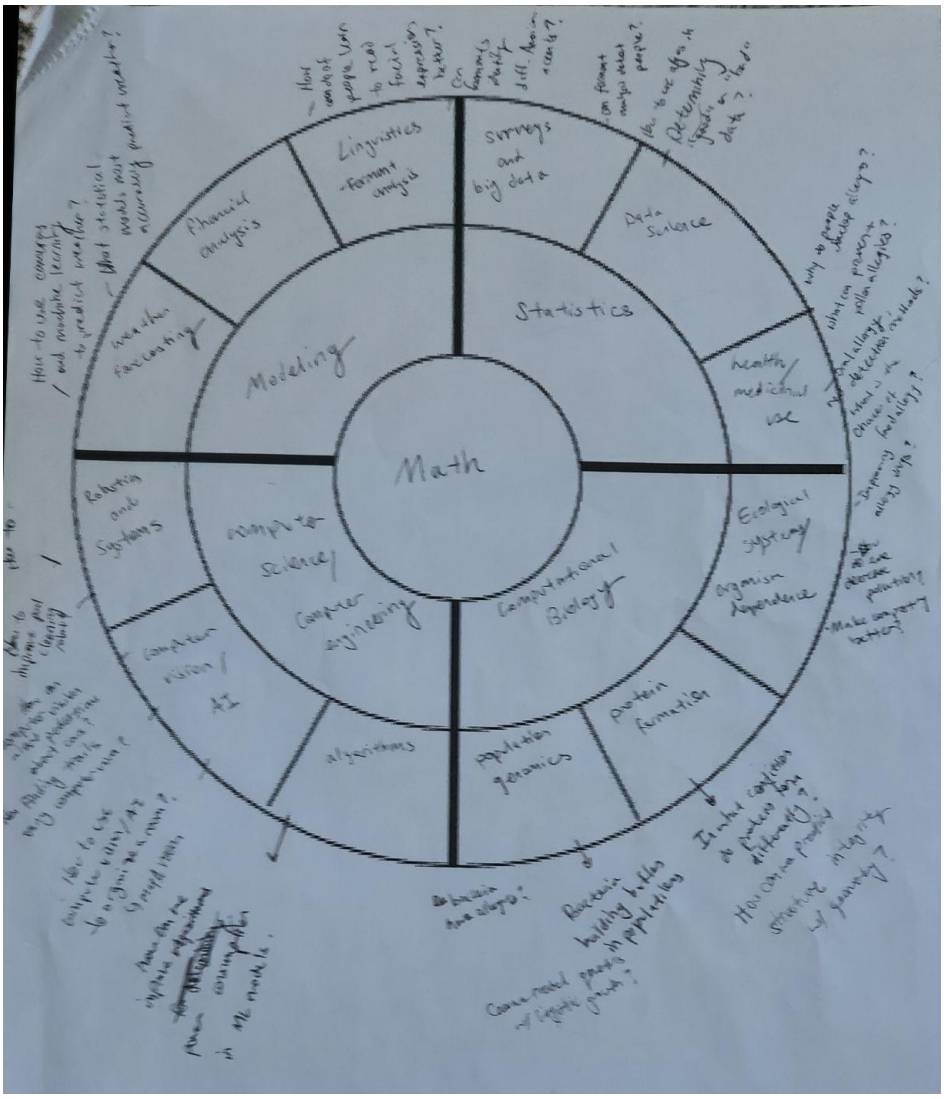
Pie Diagram #2

Date	August 19, 2022
Signature	*Charles Tang,*
Brief Description	As a badminton athlete, I have always been interested in methods of improving performance and gaining a competitive advantage on the court. One of the specific things badminton athletes train is the form, and it interests me what impact it has on shot quality and speed. Another important aspect of the game is the competitive part of it, and what the game theory is behind it. Additionally, I want to take a deeper look at the muscle groups worked by badminton and what muscle groups to train to improve athletic performance on the court. Lastly, I never fully understood the materials used in rackets, shoes, and other equipment. I would like to learn more about the materials used and what alternatives could be used in the future to improve various aspects of gameplay.



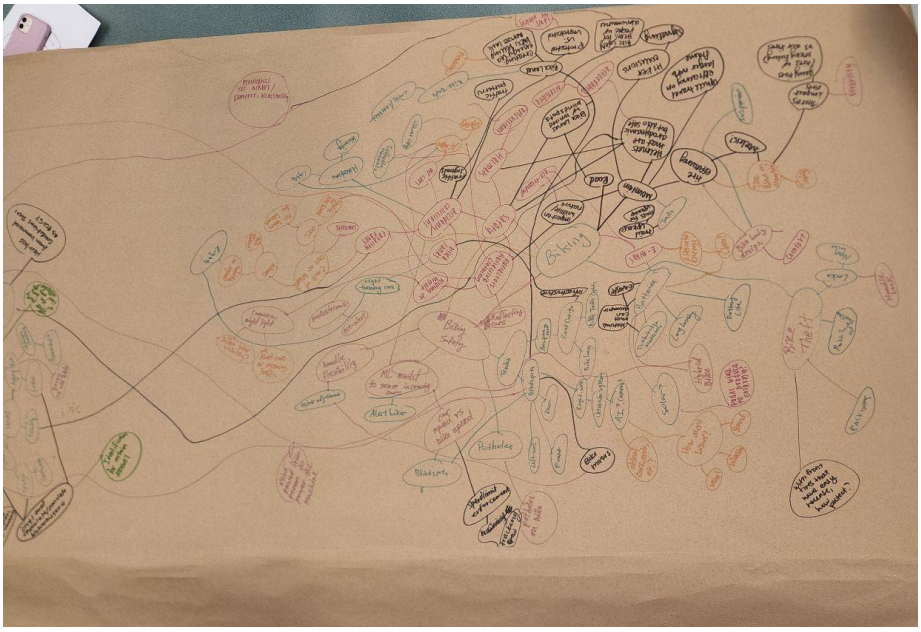
Pie Diagram #3

Date	August 19, 2022
Signature	*Charles Tang,*
Brief Description	Since a young age, I have always been intrigued by mathematics and what we can do with it. One part of math that always interested me was computer science. Another part of math that is of interest is modeling with math and statistics. For example, things that could be improved with math could be weather forecasting, linguistics, and more. Lastly, computational biology is something I have heard of but have not looked into too deeply. However, I have tried modeling proteins and the formation of them on computer software and it interests me to research more about it.



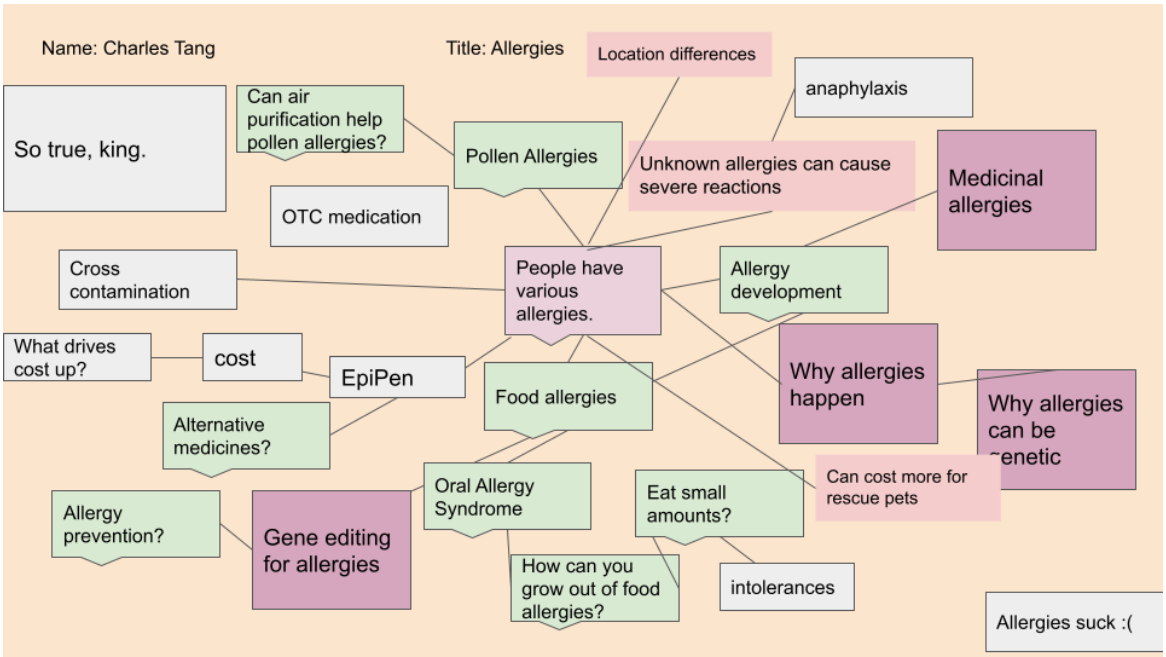
Mindmap #1

Date	August 12, 2022
Signature	*Charles Tang*,
Brief Description	This first mind map was created on the topic of biking. I have always enjoyed biking to new places around me, but there are many concerns regarding safety and bike infrastructure that should be addressed. Some of the key observations I made while biking were the presence of larger trucks, the lack of bike infrastructure, poor biking performance, and the high prices of ebikes.



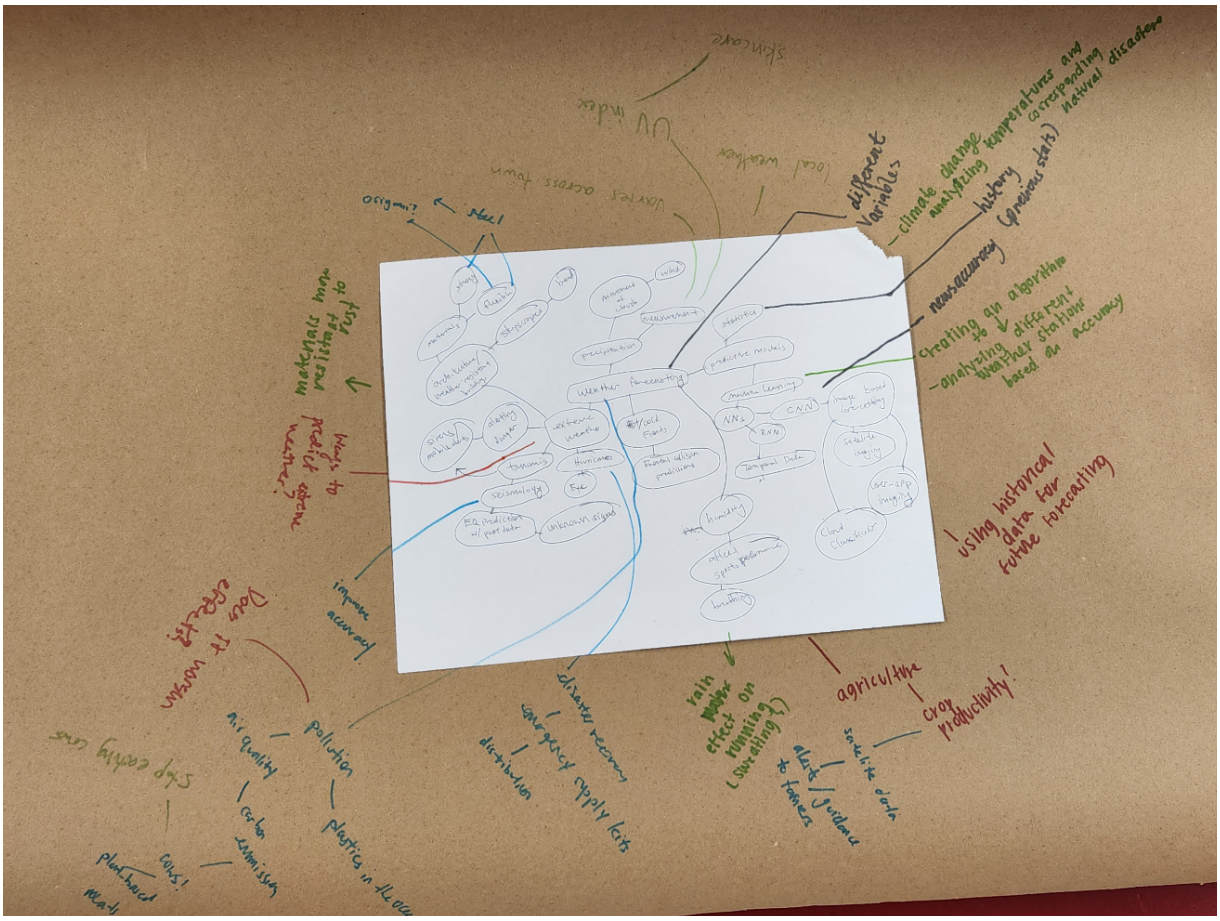
Mindmap #2

Date	July 17, 2022
Signature	*Charles Tang,*
Brief Description	As someone who developed oral allergy syndrome and pollen allergies, I have always been interested in ways to prevent allergies and the process in which allergies develop. One aspect of allergies that should be addressed are EpiPens and other cost-effective anaphylaxis treatments. I have also noticed that more of my friends have been developing oral allergy syndrome, and would like to know more about what is causing it and what can prevent the development of them.



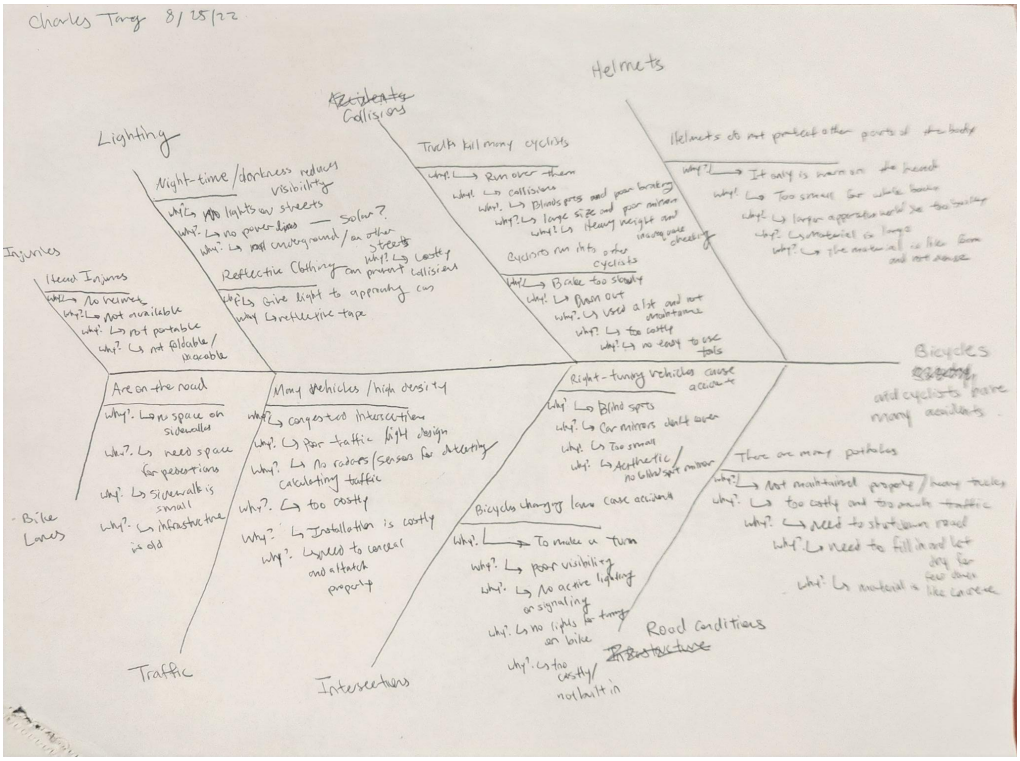
Mindmap #3

Date	September 10, 2022
Signature	*Charles Tang,*
Brief Description	This mind map's central topic is weather forecasting. It is a common experience to read the weather forecast and it is completely wrong. I find this common issue intriguing, and I hope to learn more and research about methods to improve predictions. In addition, natural extreme disasters pose another issue to us. Having experienced hurricanes on an almost yearly basis, it has caused much damage to property and more. I am interested in investigating ways to prevent and mitigate the damages caused by natural disasters and ways of protecting more people.



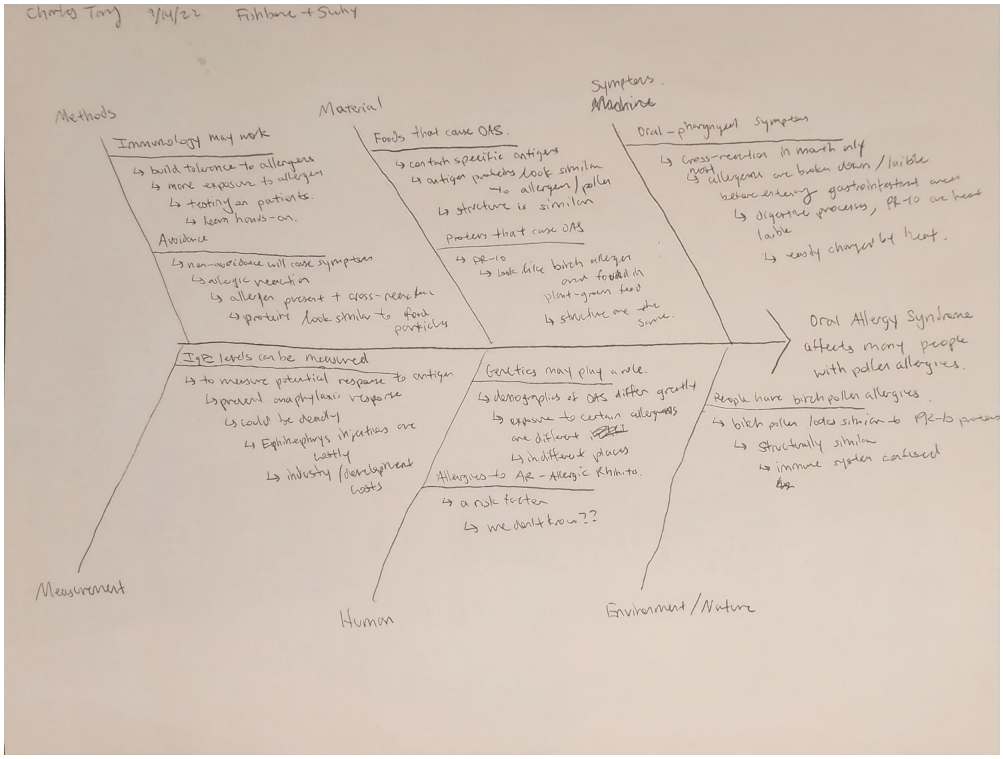
Fishbone Diagram and Five Whys #1

Date	September 5, 2022
Signature	*Charles Tang*,
Brief Description	This first fishbone diagram investigates issues of bicycle safety. One important factor of bicyclist safety is helmets. Since I often see many people not wearing their helmets when biking, I would like to learn more about the risk factors and what could be done to improve helmets. When I'm biking and a large truck passes by, I always need to move out of the way so the truck has space to pass. This brings me to the idea of learning more about collisions, possible injuries, and traffic density and how they affect the safety of a bicyclist. Lastly, blind spots are something that should be addressed, especially when vehicles make turns or lane changes. It is always hard to know if a driver sees you when driving by, so as a bicyclist it is important to be very cautious. In each rib of the fishbone diagram is a 5 Whys brainstorming method to investigate the root of each problem.



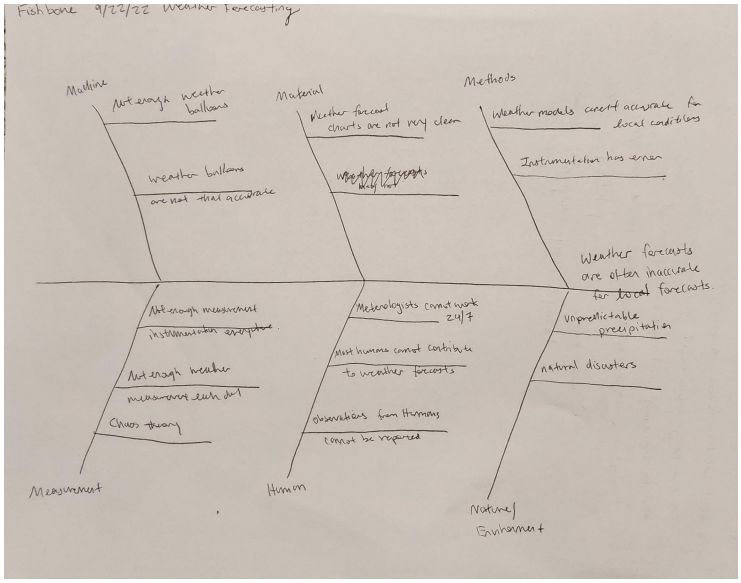
Fishbone Diagram and Five Whys #2

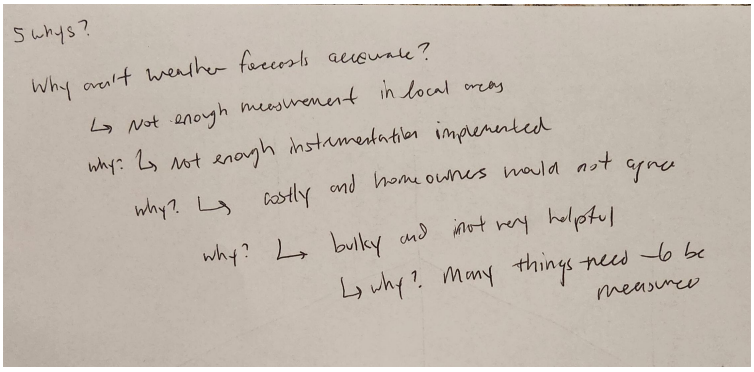
Date	September 12, 2022
Signature	*Charles Tang*,
Brief Description	This second fishbone diagram breaks down the problem of Oral Allergy Syndrome (OAS). This fishbone diagram breaks down the problem into six sections: Methods, Material, Symptoms, Measurement, Human, and Environment/Nature. Each one of these ribs identifies a cause or topic related to OAS in that section. This first topic related to OAS were Oral-Pharyngeal symptoms which were related when allergic reactions occur. I thought of this topic because I often experience these symptoms and am interested in why they occur. The second topic I investigated were foods that trigger OAS. I am interested in this topic because current research is limited about what specific foods trigger OAS and it could be dangerous to many if these are not identified. I also identified genetics as a possible cause for OAS, although the research on this hypothesis is not clear.



Fishbone Diagram and Five Whys #3

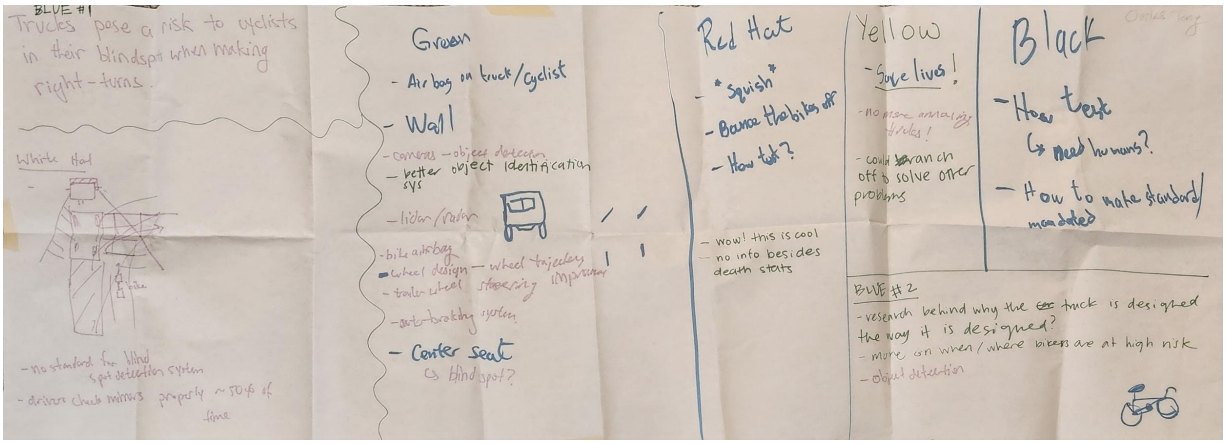
Date	September 20, 2022
Signature	*Charles Tang,*
Brief Description	The third fishbone diagram I have is about weather forecasting models and the accuracy of them. The central problem for this diagram is that weather forecasts are often inaccurate for local predictions. Each rib described a problem about the accuracy of weather forecasting models. I was interested in this topic because I often experience inaccurate weather forecasts and it often makes planning difficult. Some problems for weather forecasts can be broken down to a lack of widely distributed weather instruments, inaccurate forecasting models, or chaos theory.





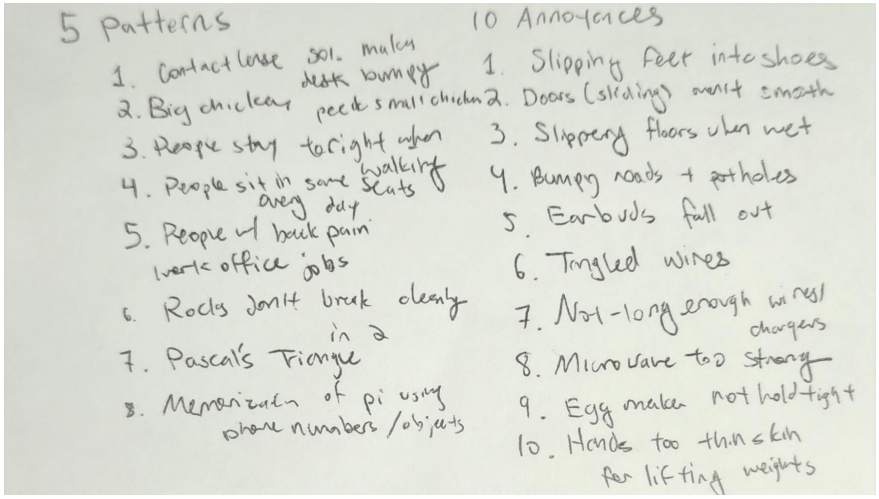
Colored Hat Activity

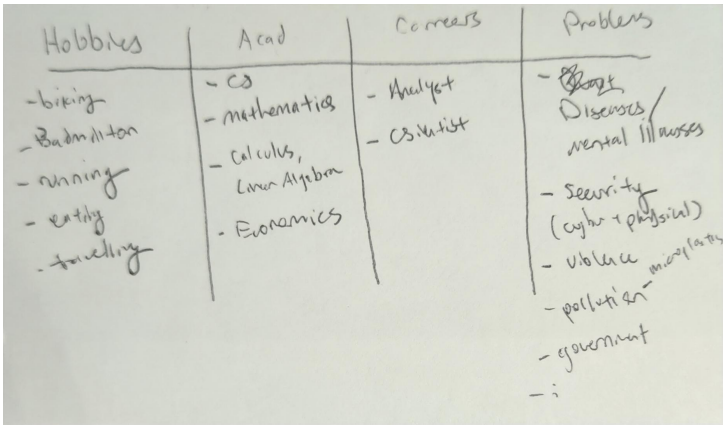
Date	September 7, 2022
Signature	*Charles Tang,*
Brief Description	This colored hat activity was performed in a group setting to further explore a project idea of the risk that trucks pose to cyclists in their blindspots. This activity analyzed the problem in a variety of different colored hats to take on different perspectives to make judgements and improve the idea. Ideas that were helpful in developing my project included analyzing different places to protect the cyclist, how important this issue is, and some places for future research to learn more in the field.



Hobbies and Basic Ideas

Date	July 10, 2022
Signature	*Charles Tang,*
Brief Description	This initial brainstorming step was conducted to think of ideas that are commonly linked to our daily lives.





Section II: Project Overview

Project Abstract

Cycling is a growing in popularity method of transportation for sustainability and health benefits. However, cyclists face growing risks, especially when encountering semi-trailer trucks. This study aims to reduce the number of truck-cyclist collisions, which are often caused by semi-trailer trucks making right-hook turns and poor driver attention to blind spots. To achieve this, we designed a visual-based blind spot warning system that can detect cyclists for semi-trailer truck drivers. First, several greater than 90% mAP cyclist detection models, such as the EfficientDet Lite 1 and SSD MobileNetV2, were created using state-of-the-art lightweight deep learning architectures fine-tuned on a newly proposed cyclist image dataset composed of a diverse set of over 20,000 images. Next, the object detection model was deployed onto a Google Coral Dev Board mini-computer with a camera module and analyzed for speed, reaching inference times as low as 15 milliseconds. Lastly, the end-to-end blind spot cyclist detection device was tested in real-time to model traffic scenarios and analyzed further for performance and feasibility. We concluded that this portable blind spot alert device can accurately and quickly detect cyclists and have the potential to significantly improve cyclist safety. Future studies could determine the feasibility of the proposed device in the trucking industry and improvements to cyclist safety over time.

Project Introduction

Bicyclists face numerous risks when traveling along urban roads and intersections. Semi-trailer truck drivers often have trouble identifying cyclists in their blind spots through mirrors when making right-hand turns which causes bicyclist-truck collisions. The overall aim of this project is to engineer an apparatus that can provide alerts for semi-trailer truck drivers when cyclists are present in their right-rear blind spots. The expected outcome of this project is to be developed using the Tensorflow Object Detection API and trained on collected bicyclist data. The developed apparatus will be analyzed using various object detection metrics and real-time testing.

Section III: Professional Communication

Email #1

Date	September 14, 2022
Email	s.j.summerskill2@lboro.ac.uk
Subject Line	Questions on a Truck Concept to Improve the Visibility of VRUs
Email	Dear Dr. Summerskill, I'm Charles, a student at the Massachusetts Academy of Math and Science at Worcester Polytechnic Institute in Massachusetts, USA. As a part of our curriculum, each student is responsible for designing a 5-month long individual science fair project. My project idea concerns the risks of blind spots in semitrailer trucks on bicycle safety. I recently read your paper "The Development of a Truck Concept to Allow Improved Direct Vision of Vulnerable Road Users by Drivers." I found this approach to the VRU safety issue unique, as the window additions and lowered design significantly improved the visibility of cyclists in the blind spots of an HGV. I have a few questions regarding this paper and some design implications. ● Is the injury severity of cyclists being run over decreased with this lower design? ● Can the lower window apertures pose a risk for glass shard exposure when front-end collisions occur? ● How does the lower design affect right-rear and left-rear blindspots when detecting VRUs? Thank you for your time and expertise in this matter. Best regards, Charles Tang

Email #2

Date	September 22, 2022
Email	2006088@hebut.edu.cn
Subject Line	Questions on the Blind Spots of a Semitrailer Truck
Email	Dear Dr. Li, I'm Charles, a student at the Massachusetts Academy of Math and Science at Worcester Polytechnic Institute in Massachusetts, USA. As a part of our curriculum, each student is responsible for designing a 5-month long individual science fair project. My project idea concerns the risks of blind spots in semitrailer trucks on bicycle safety. I recently read your paper "Exploring the Influencing Factors and Formation of the Blind Zone of a Semitrailer Truck in a Right-Turn Collision." I liked this study because the simulations run on PC-Crash provided important insight into potential collisions and the detailed breakdown of blind spots formed by a right-turning truck. I have a few questions regarding this paper and I would appreciate if you could answer them. ● Do double-trailer trucks have a greater inside wheel difference blind zone and greater risk of collision? ● How can a red light or stop before an intersection change the blind zones of right-turning trucks? Thank you for your time and expertise in this matter. Thanks, Charles Tang

Email #3

Date	September 25, 2022
Email	diego.mercado@cimat.mx
Subject Line	Questions on Cyclist Orientation Detection with Deep Learning
Email	Dear Dr. Mercado, I'm Charles, a student at the Massachusetts Academy of Math and Science at Worcester Polytechnic Institute in Massachusetts, USA. As a part of our curriculum, each student is responsible for designing a 5-month long individual science fair project. My project idea concerns the risks of blind spots in semitrailer trucks on bicycle safety. I recently read your paper "On the safety of vulnerable road users by cyclist orientation detection using Deep Learning." I found this study interesting because of the comparisons between the performances of various object detection models. I have a few questions regarding this paper, and I would appreciate it if you could answer them. ● Is the dataset DetectBike v1 and DetectBike v2 publicly available? ● Were preloaded weights from transfer learning on the COCO dataset, OpenImages, or ImageNet, and why were these weights chosen? ● How could semantic segmentation have been implemented for detecting cyclists, and how would the accuracy or speed compare to bounding box detections? Thank you for your time and expertise in this matter. Thanks, Charles Tang

Email #4

Date	October 1, 2022
Email	enquiries@sammiecad.com
Subject Line	Licensing For High School Academic Research
Email	Hello, I am Charles, a high school student at the Massachusetts Academy of Math and Science at the Worcester Polytechnic Institute in Massachusetts, USA. As a part of our curriculum, each student is responsible for designing a 5-month long individual science fair project. My project idea concerns the risks of blind spots in semitrailer trucks when making turns and how we can use blind spot detection systems to protect bicyclists and their safety. I am emailing in question about SAMMIE licensing for high school student scientific research. I am interested in using SAMMIE to analyze the blind spots of semitrailer trucks when utilizing a camera system. Since my current research is conducted without any funding or external support, would it be possible to provide a license at no or discounted cost to support my project? Best regards, Charles Tang
Response	Hi Charles, Good to hear from you. You have found yourself one of the experts in this area. My team and I have defined an approach for measuring direct vision from trucks that will be applied in the 58 countries. You can see the report for this project here. https://repository.lboro.ac.uk/articles/report/The_definition_production_and_validation_of_the_direct_vision_standard_DVS_for_HGVS_Final_Report_for_TfL_review/9353513 Some earlier work was done using SAMMIE (report link below) but this is very time consuming and complicated. https://repository.lboro.ac.uk/articles/report/Understanding_direct_and_indirect_driver_vision_in_heavy_goods_vehicles/9354344 SAMMIE is not the tool that we use for this work, and in fact RHINO 3D is the tool that we use. See if you can get a student licence for that.

	There are also efforts by the US DoT (Volpe centre) to measure direct vision. Look up the DoT VU system. Have a look at the report I send you. Best regards, Dr. Steve Summerskill

Email #5

Date	October 16, 2022
Email	Khaled.Aboufarw@uts.edu.au
Subject Line	Questions on Cyclist Detection with LiDAR
Email	Dear Dr. Khaled, I'm Charles, a student at the Massachusetts Academy of Math and Science at Worcester Polytechnic Institute in Massachusetts, USA. As a part of our curriculum, each student is responsible for designing a 5-month long individual science fair project. My project idea concerns the risks of blind spots in semitrailer trucks on bicycle safety. I recently read your paper "Cyclist detection in LIDAR scans using faster R-CNN and synthetic depth images." I found this study interesting because of the significant improvement from previous HOG-SVM methods for cyclist detection. I have a few questions regarding this paper, and I would appreciate it if you could answer them. ● What is the cost of implementing a LiDAR system with an object detection model on a real-time application? ● How would a Faster RCNN perform against SSD models for LiDAR cyclist detection? ● What is the difference in prediction time between HOG-SVM and Faster RCNN? ● How can this RCNN be fault-redundant to satisfy the requirements for safety-critical designs? Thank you for your time and expertise in this matter. Thanks, Charles Tang

Email #6

Date	November 6, 2022
Email	pradhakrishnan@wpi.edu
Subject Line	Research Project Inquiry
Email	Hello Prof. Radhakrishnan, I'm Charles, a student at the Massachusetts Academy of Math and Science located at WPI. As a part of our curriculum, each student is responsible for designing a 5-month long individual science fair project. My current project is regarding a blind spot alert apparatus for cyclists in right-turning semi-trailer trucks. I read through your webpage and found that my project closely aligns with your research interests in mechatronic systems and automation in vehicles. As an expert in this field, I was hoping to receive some feedback on my current project and potential improvements to my current design. Would it be possible to schedule a meeting sometime in the next two weeks on campus to discuss this further? Thank you for your time and consideration, Charles Tang _____ Charles Tang Mass Academy of Math and Science 85 Prescott Street Worcester, MA. 01605 ctang5@wpi.edu

Section III: STEM BiWeekly Meeting Notes

Meeting #1 - 9/15/22	TODO: ● Determine competitors in the industry ● Read more patents ● Talk to Svabhu about their project
Talk with Svabhu - 9/18/22	Notes: ● Use an SSD model to be able to detect quick enough in real time ● NVIDIA Jetson Nano, Raspberry Pi are good choices for portable computer ● Tensorflow Object Detection API ● Deploying model takes the most time
Meeting #2 - 9/30/22	TODO: ● Create a systems diagram ● Create a decision matrix with alternative competitors ● Reword project proposal ● Email SAMMIE CAD for a license ● Begin daily entries ● Begin ordering materials
Meeting #3 - 10/10/22	Notes: ● Continue with daily entries ● Order materials ● Project Proposal due Oct 1
Meeting #4 - 10/26/22 (Mr. Medeiros)	Notes: ● Continue working ● Learn more about safety-critical systems and how to make sure everything is well tested ● For presentations, use less words and explain everything on the slide from scratch ● Memorize competitors because it will probably be a common question for judge panels
Meeting #4.5 (Informal)	● Testing methods ○ Model car + fake cyclist ○ Virtual simulation videos ○ Cyclist pulled by string with dummy on it fixed camera for results ● Code coverage ○ Unit tests to cover ISO26262 ● Presentation feedback

	○ Anything put on slides should be a conscious design choice, should be explained ○ Easily readable slides ○ Hook / danger / problem ○ Balance technical terms and more general terms ○ Teach the stuff to someone else

Meeting #5	Feedback: “Please keep in mind that your audience may have varying scientific backgrounds, and with a wide range of proficiencies in machine learning — some: none at all. Balance jargon and high-level concepts carefully to create understanding that is technically accurate, while accessible. Throughout, connect your contribution to the field to the broader picture and applications. Ask: How is my research affecting the community? The project is safety-critical in nature — together, we discussed the response to the question: how much exactly does a human life cost? Please also keep in mind to think carefully about this question and its delicate response as you frame your research for its impact. Style of delivery was excellent: it was clear that you were prepared, and this showed in the confidence of your presentation. Regarding the visual aspects of your presentation: we want to remember that the audience wants to listen to you speak. They cannot read textually-heavy slides, decipher complex images, and listen to a speaker on a highly advanced topic all at the same time. Let’s think about how an audience member can a) quickly and b) with minimal effort glean insights from each slide. Every single thing — a diagram, picture, chart, word, number that you put on your slide should and must be a conscious presentation design choice. Overall, I thought your presentation was well-executed and at an A-level quality. Well done.”
Meeting #6	Feedback: Fantastic and interesting presentation. A few things stood out:

	* you welcomed the audience to participate, inviting them to recall the near-ubiquitous experience of riding a bike. * The hot wheels truck you brought to allow you, and the audience to have something to "touch" to bring the point home. * The improvement in the cleanness of the visual component of your presentation. (compared to the last formal). The style was minimalist, but not bare; included what was needed, and nothing extraneous. I recommend this to virtually everyone: you should record yourself presenting, audio/video. It can be jarring to watch and hear yourself, but the benefit is that you can see you! You'll become cognizant of certain behaviors you didn't before notice, and either repeat them, correct/improve them, or eliminate them. I'd also recommend for our next formal presentation to deliver the presentation in a large room, to try another style and setting. Overall, this presentation was high-quality work. Continue.

Section IV: Materials and Methods

Research Goals

The goal of this research project is to learn about places of high risk involving cyclists and collisions, train a cyclist object detection model, engineer an apparatus to detect bicyclists in blind spots using the object detection model, and implement this apparatus on a real-time application.

Criteria

- This apparatus should be able to cover the right-rear blind spot of a semi-trailer truck.
- This apparatus should be able to detect cyclists with greater than 80% accuracy.
- This apparatus should be able to detect cyclists in less than two seconds.
- This apparatus should be able to be portable and installable on any semi-trailer truck.
- This apparatus should be able to provide active warning alerts for cyclists.
- This apparatus should be able to withstand most weather conditions.

Constraints

- This apparatus shall cost no more than $400.00.
- This apparatus shall not interfere with the mirror visibility.

Materials List

- Computer system with GPU
- Tensorflow + Object Detection API
- Google Coral Dev Board + Case
- Webcam or Google Coral Camera
- SolidWorks 3D CAD Software
- 3D Printer
- Adhesive
- Screws

Procedure

Independent Variables: Object detection feature extractor and meta-architecture, cyclists in view of camera

Dependent Variables: Accuracy and speed of the object detection model

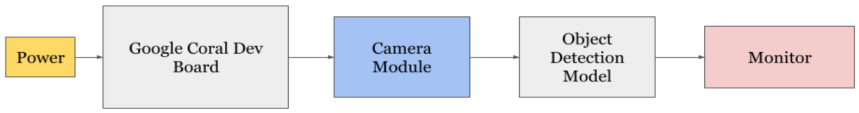
Control Variables: Training and testing dataset, computer for training the model, real-time computer

Design Process

1. Determine or create a dataset with bounding box annotations for cyclists.
2. Create a training pipeline in preparation for training a model.
    a. Convert annotations and images to tf.record files. Create label map files.
    b. Import pre-trained SSD MobileNet V2, SSD ResNet50, and Faster RCNN Inception V2 models

   c. Modify pipeline configuration files to match training specifications.
3. Train an object detection model using Tensorflow Object Detection API. Track the model performance over time using Tensorboard.
4. Export the model as a tflite graph or .pb frozen graph.
5. Deploy the model on a Google Coral Dev Board with a Coral Camera or a webcam.
6. Design a 3D printed attachment for the apparatus to attach to a semitrailer truck mirror.
7. Test the apparatus in real-time and in the field.

Overview of design of cyclist detection apparatus for blind spots of semi-trailer truck



Note. This apparatus is to be attached to a semi-trailer truck right-hand mirror.

Testing

The object detection model will be tested with COCO Object Detection metrics. The speed of the model would also be tested in frames per second (FPS) and latency (ping) on testing images. Next, the model would be tested for speed and latency on the Coral Dev Board computer. Finally, the model would be tested in real-time with a live feed.

Potential Roadblocks

One potential roadblock that may occur is difficulty in finding computational resources available to train an object detection model. A potential solution to this roadblock would be to inquire about computational resources at the WPI Academic and Computing center or borrow a colleague's computer for faster training. Another potential roadblock that may occur is finding a semi-trailer truck or a large vehicle to test real-time bicyclist detections. This roadblock can be overcome by inquiring friends or past connections for semi-trailer truck drivers.

Data Analysis

The object detection model would be tested locally using COCO Object Detection metrics with average precision (AP). Throughout the model training process, evaluation metrics such as loss and mAP can be tracked over time using Tensorboard. A chart comparing these metrics would provide a representation of the accuracy of each model. In addition, the frames per second (FPS) of each model would be compared and graphed to determine the optimal model for speed locally and on the Coral Dev Board. In addition, the latency would be computed on the final chosen model locally and on a real-time application. When testing in real-time, results would be analyzed qualitatively to determine the effectiveness of a model.

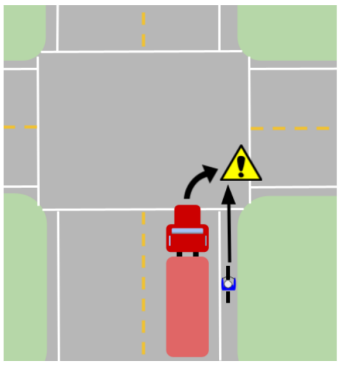
Section V: Background

Semi-trailer truck drivers often have trouble identifying cyclists in their blind spots through mirrors when making right-hand turns which causes bicyclist-truck collisions. The goal of this project is to engineer a low-cost apparatus that can accurately and quickly detect cyclists in the blind spots of a semi-trailer truck in order to provide warnings of any potential collision.

Bicyclist Safety and Collisions

A cyclist is a rider of a vehicle that travels without the help of a motor. Bicycling is a hobby and sport that many enjoy for its fitness benefits and improvement in mental health. Unfortunately, cyclists often face risks when traveling along a road or path with other motorized vehicles; in the U.S, over 1,000 cyclists die, and 130,000 more are injured each year (Centers for Disease Control and Prevention [CDC], 2022). Current safety measures for cyclists include wearing fluorescent clothing, avoiding areas of heavy traffic, and wearing helmets.

Semi-trailer truck right-hook turn scenario



Note. This scenario has a high risk of a right-hook crash, which can cause severe cyclist injuries.

Some risk factors that increase the rate of bicyclist collisions include places with fast-moving vehicles, intersections, larger vehicles, poor infrastructure, steep terrain, and poor lighting. Places that are safer for bicyclists include boulevards and broader roads. In addition, implementing lower speed limits and expanding bicycle infrastructure would reduce fatality rates (Carvajal et al., 2020). Furthermore, over 71% of cyclist incidents happen in urban areas, with 30% of collisions happening at intersections. A lack of visual attention is also a leading cause of over 55% of vehicle-cyclist collisions. A lack of visual attention includes distraction and inadequate visual understanding of the surrounding environment (Jannat et al., 2020).

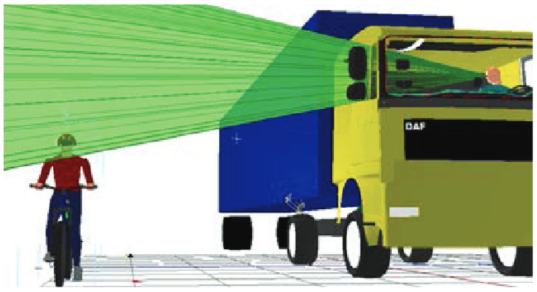
One type of collision that frequently occurs between cyclists and vehicles is in right-hook turns (Figure 1). Right-hook collisions occur at intersections when right-turning vehicles collide with a cyclist moving straight forwards along the right side of the road. These types of collisions occur when drivers do not detect bicyclists in their oncoming blind spots or when bicyclists do not observe that a driver is going to take a right turn. Some factors that increase the risk of right-hook crashes include the presence of left-turning oncoming traffic, high bicyclist speed and position in a blind spot, and the presence of pedestrians (Jannat et al., 2020).

Blind Spots of Semi-Trailer Trucks

Semi-trailer trucks have large blind spot zones on the vehicle's left, right, front, and rear. The right-hand blind spot (Figure 2) poses the most danger to vulnerable road users (VRU)–cyclists, pedestrians, moped riders, and motorcyclists–especially when making right turns. Blind spots are a leading cause of cyclist-truck incidents—they account for 45% of all collisions between bicycles and trucks. When a semi-trailer truck makes a right turn, the blind spot from the right rearview mirror increases as the view is blocked by the trailer's body. This makes turns especially dangerous because VRUs in this blind spot zone cannot be detected and face a high risk of collision and fatality (Wang et al., 2022).

As a safety measure, in the European Union, semi-trailer truck drivers are instructed to check their blind spots before and during a right turn to check for VRUs. However, a study investigating the glance behavior of European truck drivers during right-hand turns shows that drivers only properly check their blind spot mirrors about 50% of the time, which results in more significant risks of collision between trucks and VRUs (Jansen & Varotto, 2022).

Limited visibility of cyclists from a semi-trailer truck driver



Note. From "The development of a truck concept to allow improved direct vision of vulnerable road users by drivers" by S. Summerskill and R. Marshall, 2015, *Procedia Manufacturing*, 3, 3717-3724. CC BY-NC-ND.

Bicyclist Infrastructure

Several types of bicycle infrastructure exist to support safer cycling and promote cycling as a mode of transportation, such as protected intersections and bike lanes (Deliali et al., 2021). The recent growth in interest for cycling places a more significant need on infrastructures and addressing safety concerns to make cycling more inclusive for all. A survey covering the top 50 metropolitan areas found that 51% of people are interested in cycling but concerned about the risks of cycling without dedicated infrastructure or improved safety systems (Dill & McNeil, 2016).

Object Detection

Object detection is when algorithms can locate and classify objects in an input image or video. Object detection algorithms predict bounding boxes to find the location of each object. Previous methods for object detection include image processing techniques such as Histograms of Oriented Gradients, which use filters to extract features from images (Dalal & Triggs, 2005). However, these methods struggle with extracting refined details from an image that can help differentiate objects and improve accuracy. Novel object detection algorithms utilize deep learning and convolutional neural networks.

The majority of object detection architectures have a feature extractor and a meta-architecture. First, images pass through feature extractors–such as the VGG, MobileNet, Inception, and ResNet–which extract features through a convolutional neural network. Then, the media is passed through a meta-architecture such as an R-CNN or SSD model. Region-Based Convolutional Neural Networks (R-CNN) consist of a region proposal network (RPN) to locate the general regions an object may be. Then, an R-CNN predicts each object's bounding boxes and classifications with a separate neural network (Girshick et al., 2014). On the other hand, Single-Shot Detector (SSD) models consist of a single deep-learning neural network with more convolutional layers to predict bounding boxes and classifications (Liu et al., 2016). SSD models generally have faster computational speed but lower accuracy on smaller objects than R-CNN models (Huang et al., 2017).

Object detection models often are trained using transfer learning, which uses a pre-trained model on a large dataset and fine-tunes it to specialize in a specific task. One such dataset that is

commonly used is the Microsoft COCO Dataset, which contains over 200,000 labeled images and 80 object categories (Lin et al., 2015).

Blind Spot Bicyclist Detection Systems

Various bicyclist detection or blind spot assistance systems for drivers and vehicles have been proposed in recent years. One such design involves a modification to the design of a semi-trailer truck cab by reducing the overall height and adding window apertures to improve the ability to observe cyclists in the driver's blind spots (Summerskill & Marshall, 2015). However, this design lacks portability and raises questions about the costs of implementing such a design. Another design that attempts to solve the blind spot issue involves a mechanism that changes the angle of the side and rearview mirrors to the driver to view the blind spots when making turns or lane changes (Clegg, 2012). One essential attribute that this system does not provide is an active warning system that can alert drivers of cyclists in their blind spots. Without an active warning system, drivers may not pay attention to their blind spots when making maneuvers, which causes a significant risk to through-going cyclists. One deep-learning solution to the cyclist detection problem involves LiDAR scans and a convolutional neural network (Saleh et al., 2017). While LiDAR systems for cyclist detection prove to be an accurate solution, implementing LiDAR systems on modern vehicles is very costly and difficult. It is essential that any design that provides active bicycle detection and warning capabilities must be accurate, low-cost, and easily integrable into current technologies. Lastly, a common design in modern vehicles is a camera-based blind spot system. However, visual blind spot systems in commercial vehicles are unable to provide detection or alert capabilities.

References

Bicycle Safety | Motor Vehicle Safety | CDC Injury Center. (2022, May 4). Centers for Disease Control and Prevention. https://www.cdc.gov/transportationsafety/bicycle/index.html

Carvajal, G. A., Sarmiento, O. L., Medaglia, A. L., Cabrales, S., Rodríguez, D. A., Quistberg, D. A., & López, S. (2020). Bicycle safety in Bogotá: A seven-year analysis of bicyclists' collisions and fatalities. *Accident Analysis & Prevention*, *144*, 105596. https://doi.org/10.1016/j.aap.2020.105596

Clegg, T. (2012). *Apparatus and Methods for Eliminating or Reducing Blind Spots in Vehicle Mirror and Camera Systems* (Patent No. US2012022749 (A1)). https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20120126&DB=EPODOC&locale=&CC=US&NR=2012022749A1&KC=A1&ND=1

Deliali, K., Christofa, E., & Knodler Jr, M. (2021). The role of protected intersections in improving bicycle safety and driver right-turning behavior. *Accident Analysis & Prevention*, *159*, 106295. https://doi.org/10.1016/j.aap.2021.106295

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1*, 886–893 vol. 1. https://doi.org/10.1109/CVPR.2005.177

Dill, J., & McNeil, N. (2016). Revisiting the Four Types of Cyclists: Findings from a National Survey. *Transportation Research Record*, *2587*(1), 90–99. https://doi.org/10.3141/2587-11

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2017). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3296–3297. https://doi.org/10.1109/CVPR.2017.351

Jannat, M., Tapiro, H., Monsere, C., & Hurwitz, D. S. (2020). Right-Hook Crash Scenario: Effects of Environmental Factors on Driver's Visual Attention and Crash Risk. *Journal of Transportation Engineering, Part A: Systems*, *146*(5), 04020026. https://doi.org/10.1061/JTEPBS.0000342

Jansen, R. J., & Varotto, S. F. (2022). Caught in the blind spot of a truck: A choice model on driver glance behavior towards cyclists at intersections. *Accident Analysis & Prevention*, *174*, 106759. https://doi.org/10.1016/j.aap.2022.106759

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). *Microsoft COCO: Common Objects in Context* (arXiv:1405.0312). arXiv. https://doi.org/10.48550/arXiv.1405.0312

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision*, *9905*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Saleh, K., Hossny, M., Hossny, A., & Nahavandi, S. (2017). Cyclist detection in LIDAR scans using faster R-CNN and synthetic depth images. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC.2017.8317599

Summerskill, S., & Marshall, R. (2015). The Development of a Truck Concept to Allow Improved Direct Vision of Vulnerable Road Users by Drivers. *Procedia Manufacturing, 3*, 3717–3724. https://doi.org/10.1016/j.promfg.2015.07.803

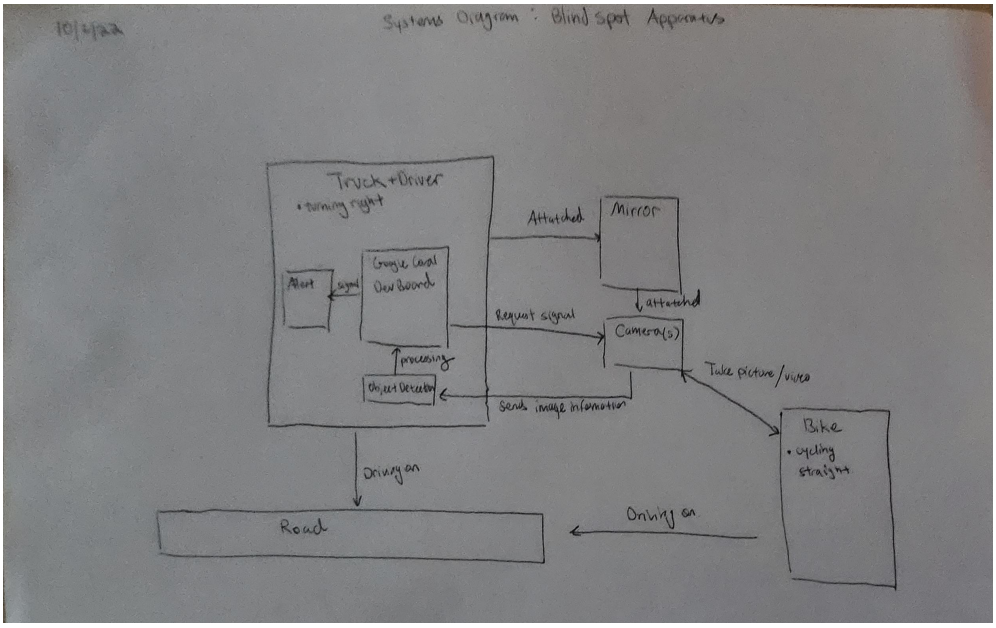
Wang, Q., Sun, J., Wang, N., Wang, Y., Song, Y., & Li, X. (2022). Exploring the Influencing Factors and Formation of the Blind Zone of a Semitrailer Truck in a Right-Turn Collision. *Sustainability*, *14*(16), Article 16. https://doi.org/10.3390/su14169805

Section VI: Daily Entries

Color Legend

Planning	Cyclist Dataset	Model Training	Deployment	Testing	Object Tracking

Daily Entry #1: System Diagram

Title	Systems Diagram
Date	October 2, 2022
Signature	*Charles Tang,*
Introduction	I developed this system's diagram to model the most important components of the system I will improve. This system diagram includes 4 main components: the truck, the object detection model, the camera, and the bicyclist. The relationships between the components are illustrated in the systems diagram shown above.
Figures	Figure 1. Systems Diagram  *Note. This diagram represents the different components of what is going on in the bicyclist detection system. Some key components include the truck, the single board computer, the camera, the bicyclist, and the object detection model.*

Concluding Remarks	This system's diagram will support my decisions on what I should prioritize and solve in this system. My first steps will be to train an object detection model and then deploy it onto a computer. Some of the parts that were most essential to developing first were the mini-computer and deep learning model.

Daily Entry #2: Design Matrix

<table>
<tr><td>Title</td><td>Engineering Design Evaluation Matrix</td></tr>
<tr><td>Date</td><td>October 3, 2022</td></tr>
<tr><td>Signature</td><td>Charles Tang,</td></tr>
<tr><td>Introduction</td><td>I developed this engineering design evaluation matrix to evaluate my design against previous competitors, patents, and research. Each design was predicted on a set of criteria each weighted based on their importance to detecting cyclists.</td></tr>
<tr><td>Figures</td><td>Table 1. Design Matrix
<table>
<tr><td></td><td></td><td colspan="4">SCORE (/1)</td></tr>
<tr><td>Criteria</td><td>Weight (%)</td><td>My Design</td><td>LiDAR System</td><td>Ultrasound System</td><td>New Cab Design</td></tr>
<tr><td>Can cover blind spots</td><td>20</td><td>1</td><td>1</td><td>1</td><td>0.8</td></tr>
<tr><td>Detect cyclists with > than 80% accuracy</td><td>5</td><td>0.9</td><td>0.9</td><td>0.7</td><td>N/A</td></tr>
<tr><td>Detect cyclists in < 1 sec</td><td>10</td><td>1</td><td>1</td><td>1</td><td>N/A</td></tr>
<tr><td>Portable and installable on any truck</td><td>20</td><td>1</td><td>0.4</td><td>0.6</td><td>0</td></tr>
<tr><td>Active alert system</td><td>15</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>Withstand most weather</td><td>2.5</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Cost no more than $400</td><td>15</td><td>1</td><td>0</td><td>0.7</td><td>0</td></tr>
<tr><td>< 10% False Detections</td><td>10</td><td>0.7</td><td>0.7</td><td>0.4</td><td>N/A</td></tr>
<tr><td>Not interfere with mirrors</td><td>2.5</td><td>0.8</td><td>1</td><td>1</td><td>N/A</td></tr>
<tr><td>TOTAL (MAX: 100)</td><td>100</td><td>96</td><td>65.9</td><td>80</td><td>18.5</td></tr>
</table>
Note. This table evaluates my current design, a LiDAR bicyclist detection system (Saleh et al., 2017), an ultrasound system, and a new cab design (Summerskill & Marshall, 2015). The weights for each criteria are based on the criteria levels for a new blind spot bicyclist detection system. The results of this matrix would be able to prove in what areas my current idea would work well and what other areas should be worked on. This matrix would also prove the efficacy and usefulness of my proposed design.</td></tr>
<tr><td>Concluding Remarks</td><td>The results show that my proposed design satisfies the criteria much better than competitors or previous research. Some of the most prominent benefits to my design is that it can be portable, has a low cost, and provides active alerts. A next step that I would have is to construct a diagram that can outline the hardware structure of the design. Furthermore, I would need some preliminary prototype object detection models for a proof-of-concept model.</td></tr>
</table>

Daily Entry #3: TF Object Detection API

Title	Setting Up Tensorflow Object Detection API
Date	October 4, 2022
Signature	*Charles Tang,*
Introduction	I followed the Tensorflow Object Detection API and installed software onto my computer. Additionally, I downloaded the CIMAT bicyclist detection dataset and converted the dataset into a TF record file. I also imported an SSD mobilenet model to train a prototype model. To export the .record files, I edited a python file in the export to tf record process to account for this dataset by looping over each file name properly.
Methods	Laptop Specification: 11th Gen Intel(R) Core(TM) i5-1135G7, 8Gb DDR4 RAM 1. CIMAT Dataset: https://gitlab.com/MaryChelo/cimat-cyclist I downloaded this dataset which was used in research on detecting orientations of cyclists (Garcia-Venegas et al., 2021). This previous research would help my project identify bicyclists whether or not they are at risk of collision to semi-trailer trucks in their blindspots. 2. Initialize File Tree ├———annotations │ └———train.recod │ └———test.record ├———exported_models ├———images │ ├———test │ └———train ├———models │ └———ssd_mobilenet_v2_fpnlite │ └———train └———pre-trained-models └———ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8 ├———checkpoint └———saved_model └———variables This file tree is the workspace for training object detection models and having all of the code and relevant information in one location. This file tree was created with the help of the Tensorflow Object Detection API – TensorFlow 2

	Object Detection API tutorial — TensorFlow 2 Object Detection API tutorial documentation (tensorflow-object-detection-api-tutorial.readthedocs.io). 3. Create a label map file. A label map file was created to generate a tf.record file. Since there are 8 cyclist orientation types in the dataset, the label map has 8 classes to specify the object and its classification. (Code #1) 4. Generate training records. The training records were then generated by the following command line command: python generate_tfrecord.py -x C:/.../Tensorflow/workspace/training_demo/images/train -l C:/.../Tensorflow/workspace/training_demo/annotations/label_map.pbtxt -o C:/.../Tensorflow/workspace/training_demo/annotations/train2.record The generate_tfrecord.py file was edited with the label map values to account for the 8 classes in the model rather than the 90 classes of the original COCO dataset (Code #2).
Code #1	`./annotations/label_map.pbtxt` `item {` `  id: 1` `  name: 'CyclistN'` `}` `item {` `  id: 2` `  name: 'CyclistNE'` `}` `item {` `  id: 3` `  name: 'CyclistE'` `}` `item {` `  id: 4` `  name: 'CyclistSE'` `}` `item {` `  id: 5` `  name: 'CyclistS'` `}` `item {` `  id: 6` `  name: 'CyclistWS'` `}`

	item { id: 7 name: 'CyclistW' } item { id: 8 name: 'CyclistNW' }
Code #2	./scripts/generate_tfrecord.py import os import glob import pandas as pd import io import xml.etree.ElementTree as ET import argparse os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # Suppress TensorFlow logging (1) import tensorflow.compat.v1 as tf from PIL import Image from object_detection.utils import dataset_util, label_map_util from collections import namedtuple # Initiate argument parser parser = argparse.ArgumentParser(description="Sample TensorFlow XML-to-TFRecord converter") parser.add_argument("-x", "--xml_dir", help="Path to the folder where the input .xml files are stored.", type=str) parser.add_argument("-l", "--labels_path", help="Path to the labels (.pbtxt) file.", type=str) parser.add_argument("-o", "--output_path", help="Path of output TFRecord (.record) file.", type=str) parser.add_argument("-i", "--image_dir", help="Path to the folder where the input image files are stored. " "Defaults to the same directory as XML_DIR.", type=str, default=None)

```
parser.add_argument("-c",
                    "--csv_path",
                    help="Path of output .csv file. If none
provided, then no file will be "
                         "written.",
                    type=str, default=None)

args = parser.parse_args()

if args.image_dir is None:
    args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
    """Iterates through all .xml files (generated by labelImg) in a
given directory and combines
    them in a single Pandas dataframe.

    Parameters:
    ----------
    path : str
        The path containing the .xml files
    Returns
    -------
    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        filename = root.find('filename').text
        width = int(root.find('size').find('width').text)
        height = int(root.find('size').find('height').text)
        for member in root.findall('object'):
            bndbox = member.find('bndbox')
            value = (filename,
                     width,
                     height,
                     member.find('name').text,
                     int(bndbox.find('xmin').text),
                     int(bndbox.find('ymin').text),
                     int(bndbox.find('xmax').text),
```

```
                    int(bndbox.find('ymax').text),
                    )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def class_text_to_int(row_label):
    # return label_map_dict[row_label]
    if row_label == "CyclistN":
        return 1
    elif row_label == "CyclistNE":
        return 2
    elif row_label == "CyclistE":
        return 3
    elif row_label == "CyclistSE":
        return 4
    elif row_label == "CyclistS":
        return 5
    elif row_label == "CyclistWS":
        return 6
    elif row_label == "CyclistW":
        return 7
    elif row_label == "CyclistNW":
        return 8

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    print(group.filename)
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename) + ".jpg"), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
```

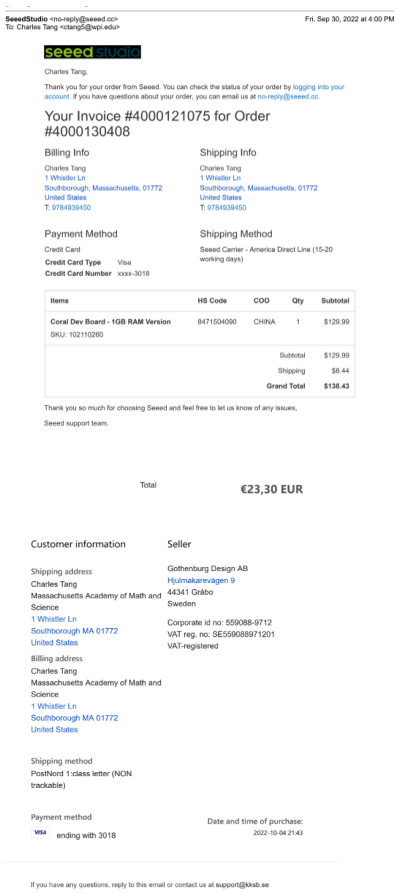
```
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example =
tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):

    writer = tf.python_io.TFRecordWriter(args.output_path)
    path = os.path.join(args.image_dir)
    examples = xml_to_csv(args.xml_dir)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
```

	`` writer.close() print('Successfully created the TFRecord file: {}'.format(args.output_path)) if args.csv_path is not None: examples.to_csv(args.csv_path, index=None) print('Successfully created the CSV file: {}'.format(args.csv_path)) if __name__ == '__main__': tf.app.run() ```
Results	The exported tensorflow record files are below. The train file is composed of around 30,000 images and the test/validation record file consists of around 500 images. Train.record file: (4.35Gb) https://drive.google.com/file/d/1c_fgqlS4OSyxScokKtgMwdIQk9wdGzX0/view?usp=sharing Test.record file: (354Mb) https://drive.google.com/file/d/1MuTFhz_WTYgI-IHtAju2RkHcSpHbzTMH/view?usp=sharing
Concluding Remarks	The next step is to configure a training pipeline from each pretrained model and to run some prototype training on them. Furthermore, I will need to determine more computational resources to train larger and more powerful models.

Daily Entry #4: Materials + Pipeline

Title	Order materials and prepared training pipeline
Date	October 6, 2022
Signature	*Charles Tang,* 
Introduction	Today, I ordered the Google Coral Dev Board and the KKSB Coral Dev Board Case for real-time application of the object detection model. At the same time, I modified the pipeline.config file for the SSD MobileNet V2 to configure this model for training. This is enough to create the training pipeline that can now train the model.
Methods	Materials ordered: Google Coral Dev Board + KKSB Coral Dev Board Case 1. Order materials. The Google Coral Board would provide a single board computer to run real-time object detection on a semi-trailer truck. Furthermore, the KKSB Coral Dev Board aluminum case would provide an encasing for protecting the google coral dev board. This is important because driving in a semi-trailer truck would not be smooth and the single board computer should be protected to prevent any damage to the components. 2. Setup training pipeline. I then configured the training pipeline for the SSD MobileNet V2 model for training. The model was set to train in 100,000 iterations of batch-size 8. The training pipeline configuration file is shown below. Note that from the default, the classes were changed to 8, the batch_size was changed to 8, and the warmup steps were changed to 500. To account for the test and train record files, the values were modified to be the file paths to the record files. (Code #1)

Receipts



SeeedStudio <no-reply@seeed.cc> Fri, Sep 30, 2022 at 4:00 PM
To: Charles Tang <ctang5@wpi.edu>

seeed studio

Charles Tang,

Thank you for your order from Seeed. You can check the status of your order by logging into your account. If you have questions about your order, you can email us at no-reply@seeed.cc.

Your Invoice #4000121075 for Order #4000130408

Billing Info

Charles Tang
1 Whistler Ln
Southborough, Massachusetts, 01772
United States
T: 9784939450

Shipping Info

Charles Tang
1 Whistler Ln
Southborough, Massachusetts, 01772
United States
T: 9784939450

Payment Method

Credit Card
Credit Card Type Visa
Credit Card Number xxxx-3018

Shipping Method

Seeed Carrier - America Direct Line (15-20 working days)

Items	HS Code	COO	Qty	Subtotal
Coral Dev Board - 1GB RAM Version SKU: 102110260	8471504090	CHINA	1	$129.99
			Subtotal	$129.99
			Shipping	$8.44
			Grand Total	$138.43

Thank you so much for choosing Seeed and feel free to let us know of any issues,

Seeed support team.

Total €23,30 EUR

Customer information

Shipping address
Charles Tang
Massachusetts Academy of Math and Science
1 Whistler Ln
Southborough MA 01772
United States

Billing address
Charles Tang
Massachusetts Academy of Math and Science
1 Whistler Ln
Southborough MA 01772
United States

Shipping method
PostNord 1:class letter (NON trackable)

Payment method
VISA ending with 3018

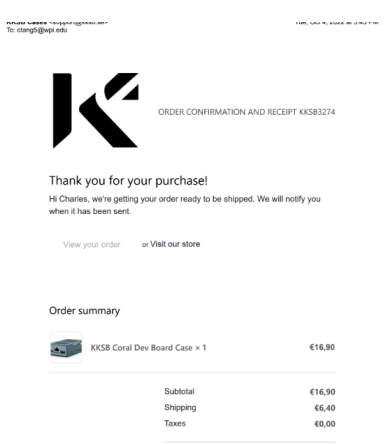
Seller

Gothenburg Design AB
Hjulmakarevägen 9
44341 Grästorp
Sweden

Corporate id no: 559088-9712
VAT reg. no: SE559088971201
VAT-registered

Date and time of purchase:
2022-10-04 21:43

If you have any questions, reply to this email or contact us at support@kksb.se



To: ctang5@wpi.edu

ORDER CONFIRMATION AND RECEIPT KKSB3274

Thank you for your purchase!

Hi Charles, we're getting your order ready to be shipped. We will notify you when it has been sent.

View your order or Visit our store

Order summary

KKSB Coral Dev Board Case × 1 €16,90

Subtotal	€16,90
Shipping	€6,40
Taxes	€0,00

Code #1

```
./training_demo/models/ssd_mobilenet_v2_fpnlite/pipeline.config
model {
  ssd {
    num_classes: 8
    image_resizer {
      fixed_shape_resizer {
        height: 640
        width: 640
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v2_fpn_keras"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
```

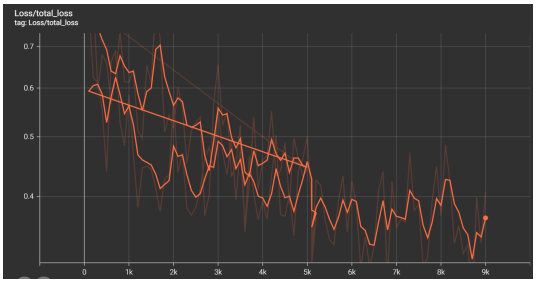
```
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.009999999776482582
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.996999979019165
        scale: true
        epsilon: 0.0010000000474974513
      }
    }
    use_depthwise: true
    override_base_feature_extractor_hyperparams: true
    fpn {
      min_level: 3
      max_level: 7
      additional_layer_depth: 128
    }
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
      use_matmul_gather: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  box_predictor {
    weight_shared_convolutional_box_predictor {
      conv_hyperparams {
        regularizer {
```

```
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      depth: 128
      num_layers_before_predictor: 4
      kernel_size: 3
      class_prediction_bias_init: -4.599999904632568
      share_prediction_tower: true
      use_depthwise: true
    }
  }
  anchor_generator {
    multiscale_anchor_generator {
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      scales_per_octave: 2
    }
  }
  post_processing {
    batch_non_max_suppression {
      score_threshold: 9.99999993922529e-09
      iou_threshold: 0.6
      max_detections_per_class: 100
      max_total_detections: 100
      use_static_shapes: false
    }
    score_converter: SIGMOID
  }
  normalize_loss_by_num_matches: true
```

```
    loss {
      localization_loss {
        weighted_smooth_l1 {
        }
      }
      classification_loss {
        weighted_sigmoid_focal {
          gamma: 2.0
          alpha: 0.25
        }
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 8
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.1
          total_steps: 100000
          warmup_learning_rate: 0.05
          warmup_steps: 500
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
```

	fine_tune_checkpoint: "models/ssd_mobilenet_v2_fpnlite/checkpoints/ckpt-95" num_steps: 100000 startup_delay_steps: 0.0 replicas_to_aggregate: 8 max_number_of_boxes: 100 unpad_groundtruth_tensors: false fine_tune_checkpoint_type: "detection" use_bfloat16: false fine_tune_checkpoint_version: V2 } train_input_reader { label_map_path: "annotations/label_map.pbtxt" tf_record_input_reader { input_path: "annotations/train2.record" } } eval_config { metrics_set: "coco_detection_metrics" use_moving_averages: false } eval_input_reader { label_map_path: "annotations/label_map.pbtxt" shuffle: true num_epochs: 1 tf_record_input_reader { input_path: "annotations/test2.record" } }
Concluding Remarks	These devices will be arriving in around ~20 days. The goal is to have a completed trained prototype model ready for deployment by the arrival of this device. Future work would include beginning to train the cyclist detection model and validating the training and validation dataset splits.

Daily Entry #5: Train Model (N)

Title	Train initial prototype model (Only Detect N) – Not Used
Date	October 7, 2022
Signature	*Charles Tang,*
Introduction	I trained a SSD MobileNet V2 for 9000 iterations with batch size 8 images per step. This model was trained for 16 hours and 6-8 seconds per step. I then evaluated this model up to this checkpoint. I also set up a training pipeline for a RetinaNet (SSD + ResNet) model.
Methods	1. Initialize Training To initialize the training procedure on a laptop, the following command was run. This command was outlined in the Object Detection API: ./training_demo (tensorflow) anaconda: python model_main_tf2.py --model_dir=models/ssd_mobilenet_v2_fpnlite --pipeline_config_path=models/ssd_mobilenet_v2_fpnlite/pipeline.config
Results	Figure 1. Loss for first 9,000 iterations.  *Note. This loss figure is from the Tensorboard application that is run locally on evaluation files.* Figure 2. Example Prediction #1.  *Note. This figure shows the first prediction of the model on a picture with numerous cyclists. The right-side represents the ground truth bounding boxes of the image, and the left side are the predictions from the model.*
Data Analysis	This model was configured incorrectly and only detected cyclists that were pointing North (1 out of the 8 classes). The function that mapped the label IDs

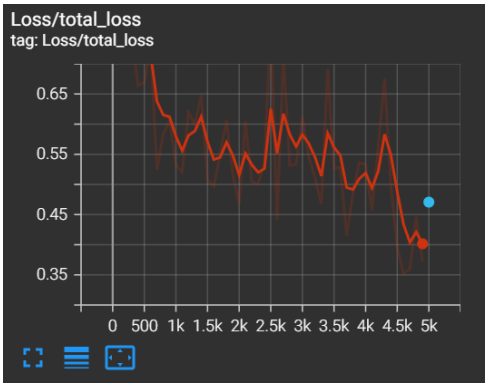
	from the label_map.pbtxt files was updated to correctly reflect the model. I reconfigured the training and testing record files and changed the postprocessing function from SIGMOID to SOFTMAX. The results were unpromising since the number of detections were extremely limited and nearly all of the predictions were only made on cyclists pointing North.
Concluding Remarks	Future work would include retraining the model and making sure that all eight orientations of cyclists are covered.

Daily Entry #6: Train Model (SOFTMAX)

Title	Train initial prototype model (SOFTMAX) - Not Used
Date	October 8, 2022
Signature	*Charles Tang,*
Introduction	Today, I trained another SSD MobileNet V2 model with a reconfigured training pipeline. I modified the non-linearity function for post-processing from sigmoid to softmax. I then trained this model for 5000 steps with batch size 8 overnight and recorded results. The results showed that accuracy was poor and the model made numerous false detections. I then recreated the tfrecord dataset files and updating the label map to prepare for other prototype training tests.
Methods	1. Modify pipeline files. (Code 1) The pipeline configuration model for configuring the parameters for training a MobileNet model was modified to accommodate for this change to a SOFTMAX post-processing function. A softmax post processing function would introduce a different type of nonlinearity into the feature extractor. The new pipeline configuration is shown below. 2. Train model This model was trained for 5,000 steps over 6 hours. The results were viewed through the Tensorboard environment.
Code #1	`./training_demo/models/ssd_mobilenet_v2_fpnlite/pipeline.config` `model {` `  ssd {` `    num_classes: 8` `...` `...  post_processing {` `      batch_non_max_suppression {` `        score_threshold: 9.99999993922529e-09` `        iou_threshold: 0.6` `        max_detections_per_class: 100` `        max_total_detections: 100` `        use_static_shapes: false` `      }` `      score_converter: SOFTMAX` `    }` `...` `}`

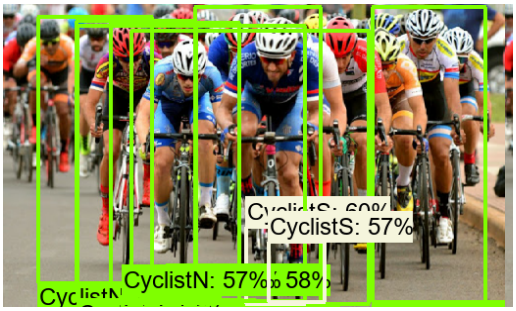
Results

Figure 1. Loss vs. Time for 5,000 Steps MobileNet.



Note. This figure shows the loss over the 5,000 steps trained for the MobileNet. It can be seen that the loss generally trends downwards, which is positively associated with accuracy of the model.

Figure 2. Example Prediction #1.

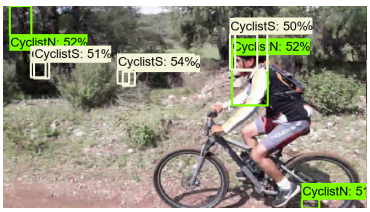


Note. This figure shows the first prediction of the newly trained model on a picture with numerous cyclists.

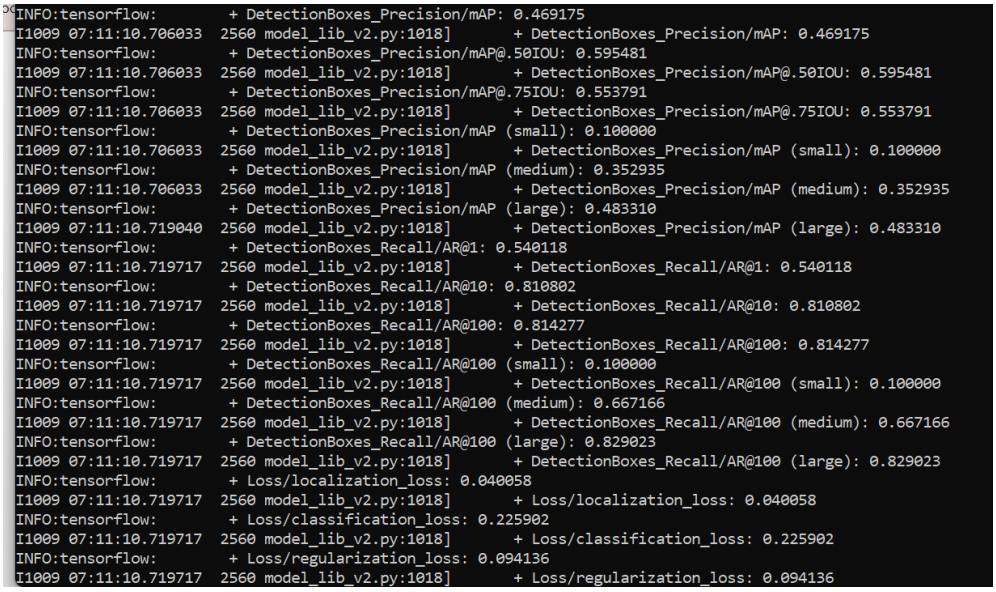
Figure 3. Example Prediction #2.

Note. This figure shows the second prediction of the newly trained model on a picture with a cyclist in the West direction.

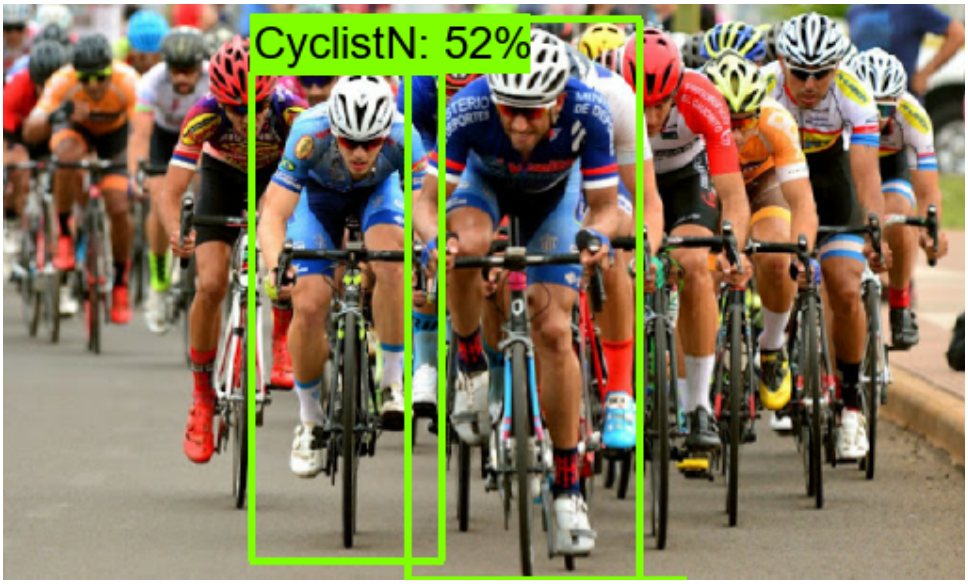
Figure 4. Example Prediction #3.

	 *Note. This figure shows the third prediction of the newly trained model on a picture with one cyclist in the East direction.*
Data Analysis	Although the softmax post-processing function showed more boxes, the accuracy was significantly worse. Furthermore, the loss was almost twice of the sigmoid post-processing function after training for around 5,000 iterations. This showed that the softmax post processing function is not the correct choice and the sigmoid function should be continued. The reason that it does not work is that this post-processing function performs the task of box regression, which basically outputs if the box is a cyclist is not. It would not make sense for each box to use a softmax classifier because a softmax classifier needs multiple regressors per box.
Concluding Remarks	The next task to work on is to fully train a MobileNet model with the newly corrected dataset and to further evaluate the models using mAP.

Daily Entry #7: Continue Training

Title	Continue training initial prototype model
Date	October 9, 2022
Signature	*Charles Tang,*
Introduction	Today, I trained another SSD MobileNet V2 model with a redesigned dataset. This model was trained with the post processing function reverted back to sigmoid.
Methods	1. Training I trained this model for 15,000 steps with batch size 8. I trained this model using the model_main python file which initializes batch tensorflow training. Furthermore, this model was trained with 500 warm up steps. Checkpoint files were created at every 1,000 steps.
Results	Checkpoint Files: https://drive.google.com/drive/folders/18_SXs7A2CpcGY3ZWNFGWc0V8j65lMz4Z?usp=sharing Figure 1. Precision Results  *Note. This figure shows the COCO object detection metrics run on the model with various Intersection Over Union parameters. The mAP of the model is 47%.* Figure 2. Example Prediction #1.

```
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.469175
I1009 07:11:10.706033  2560 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP: 0.469175
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 0.595481
I1009 07:11:10.706033  2560 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.50IOU: 0.595481
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 0.553791
I1009 07:11:10.706033  2560 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.75IOU: 0.553791
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (small): 0.100000
I1009 07:11:10.706033  2560 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (small): 0.100000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (medium): 0.352935
I1009 07:11:10.706033  2560 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (medium): 0.352935
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (large): 0.483310
I1009 07:11:10.719040  2560 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (large): 0.483310
INFO:tensorflow:        + DetectionBoxes_Recall/AR@1: 0.540118
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@1: 0.540118
INFO:tensorflow:        + DetectionBoxes_Recall/AR@10: 0.810802
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@10: 0.810802
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100: 0.814277
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100: 0.814277
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (small): 0.100000
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (small): 0.100000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (medium): 0.667166
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (medium): 0.667166
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (large): 0.829023
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (large): 0.829023
INFO:tensorflow:        + Loss/localization_loss: 0.040058
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + Loss/localization_loss: 0.040058
INFO:tensorflow:        + Loss/classification_loss: 0.225902
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + Loss/classification_loss: 0.225902
INFO:tensorflow:        + Loss/regularization_loss: 0.094136
I1009 07:11:10.719717  2560 model_lib_v2.py:1018]       + Loss/regularization_loss: 0.094136
```

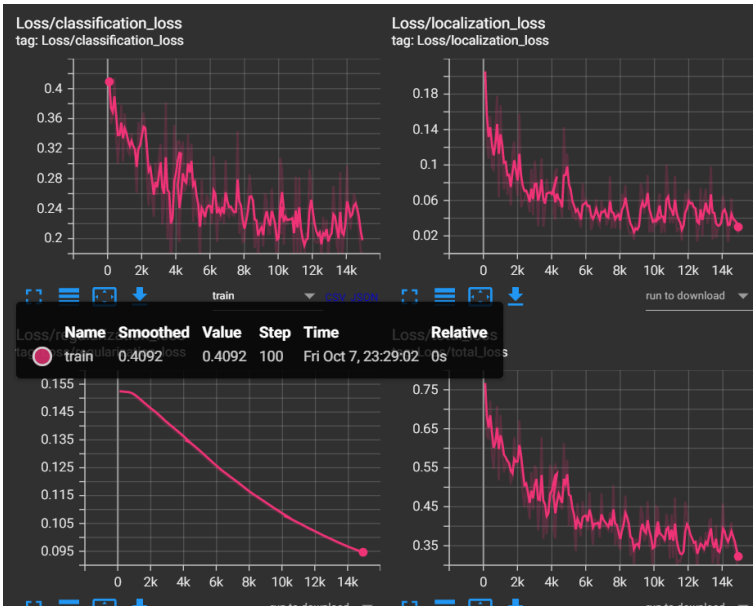


Note. This figure shows the first prediction of the prototype model on a picture with numerous cyclists in the North direction.

Figure 3. Example Prediction #2.

Note. This figure shows the second prediction of the prototype model on a picture with a cyclist in the West direction.

Figure 4. Loss Over Steps.

	 *Note. This figure shows the classification, localization, and total loss over time as training steps increase.*
Data Analysis	This model performs better than the previous model trained at 5,000 steps. The AR is very high but the model fails to detect many medium or smaller sized cyclists. To increase the performance, I will continue training this model until 25,000 or 50,000 steps. I will also need to develop a script for displaying evaluation images other than the 10 provided by tensorboard. It also can be observed that the loss is trending downwards continually, which means that further training would further improve results. Furthermore, the classification loss is very high and training with more iterations would help improve the loss. A limitation for the accuracy may be due to the limited images in a dataset to detect all 8 orientations of a cyclist. An improvement in this model is that it can detect cyclists in directions other than the North orientation.
Conclusion	Future work would be to continue training this model for more iterations and to use more data set images to improve the quality of the model.

Daily Entry #8: Continue Training

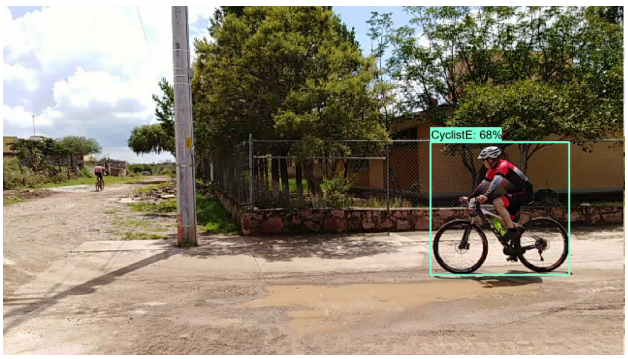
Title	Continue training initial prototype model
Date	October 9, 2022
Signature	*Charles Tang,*
Introduction	Today, I continued training the SSD MobileNet V2 model. I then trained this model for an additional 8000 steps with batch size 8 up to 23000 steps. This is a continuation of the previous entry's model. I used the same methods as the previous Daily Entry.
Results	Checkpoint Files: https://drive.google.com/drive/folders/1T8-WRKr_4E3LL7kx03sI5QMr1cBQbAOS?usp=sharing Figure 1. Example Prediction #1.  *Note. This figure shows the first prediction of the updated prototype model on a picture with a cyclist in the East direction.* Figure 2. Example Prediction #2.  *Note. This figure shows the second prediction of the updated prototype model on a picture with a cyclist in the NorthWest direction.* Figure 3. Example Prediction #3.

Note. This figure shows the third prediction of the updated prototype model on a picture with cyclists in the East and Southeast direction.

Figure 4. Example Prediction #4.

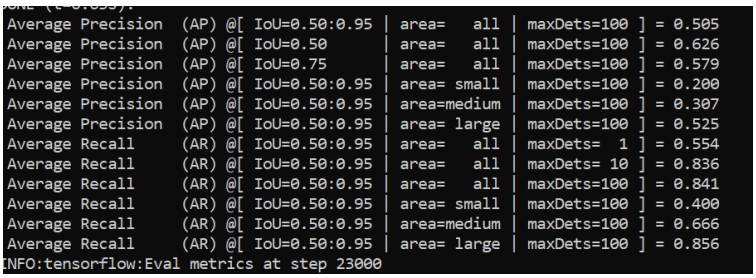
Note. This figure shows the fourth prediction of the updated prototype model on a picture with cyclists in the North direction.

Figure 5. Example Prediction #5.

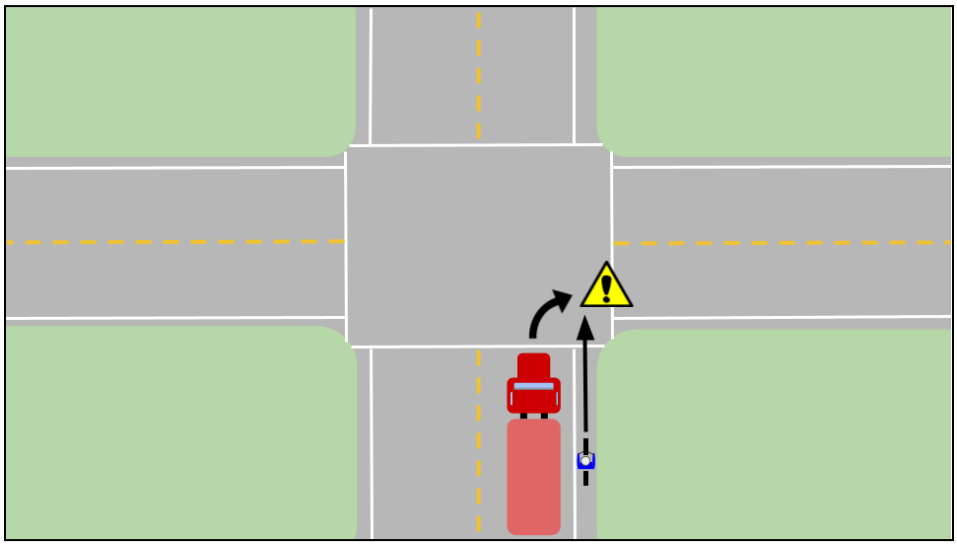


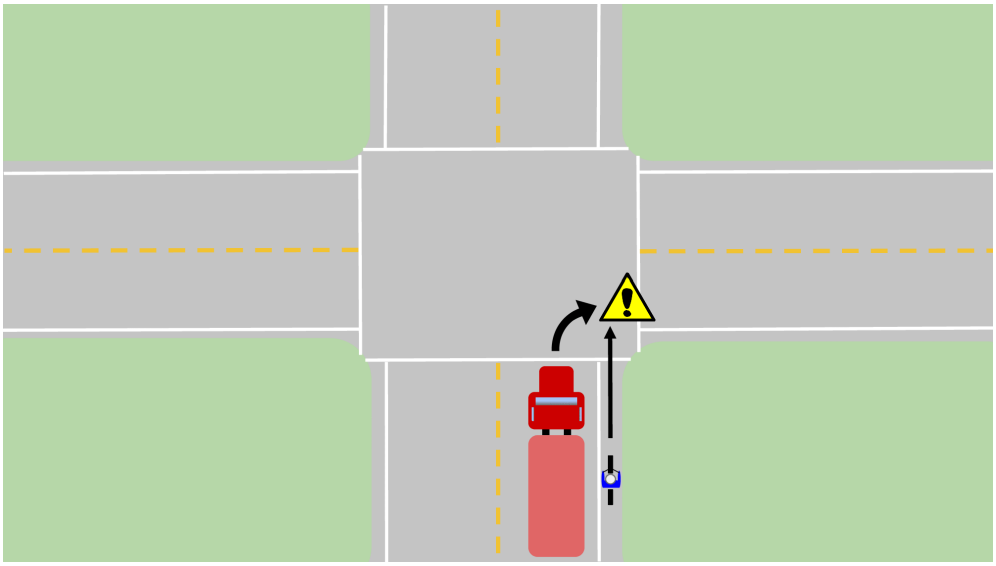
Note. This figure shows the fifth prediction of the updated prototype model on a picture with a cyclist in the East direction.

Figure 6. Object Detection Metrics.

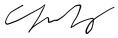
	 *Note. This figure shows the COCO object detection metrics on the updated prototype model. The mAP for this model is 51%, which is an improvement from the previous checkpoint.*
Data Analysis	This data was shuffled from throughout the test set to examine results more accurately. The current mAP is promising, which shows that longer training could show almost linear growth in mAP. The data should be trained to around 30-50000 iterations. The loss is steadily declining and the accuracy of the model is rising. Furthermore, it seems from Figure 17 that the model is performing better on smaller cyclist instances. The goal mAP of the model is to reach at least 0.5691 as described in the paper “On the safety of vulnerable road users by cyclist orientation detection using Deep Learning.” Other cyclist classes seem to be doing well, other than only CyclistN. This means that the model is generalizing well to other cyclist directions. However, the low mAP may not be usable for actual use in safety-critical design as the accuracy does not meet the objectives for this research.
Conclusion	The next step in the research process is to continue training the model and to reevaluate the objectives for their feasibility. Furthermore, it would be important to take into account safety critical design and managing redundancies within the system by improving technologies.

Daily Entry #9: Graphic Design

Title	Create graphic design modeling right-hook turns and crashes
Date	October 15, 2022
Signature	*Charles Tang,*
Introduction	The following graphic was created as a part of the systems diagram to visualize the problem this project addresses. The right-turning truck is shown to have a collision path with the thru-going cyclist, therefore causing an issue.
Design Process	The graphic design was created from google slides. Each frame of the GIF is a copy made in the same slideshow and the images were compiled to create the animation. Furthermore, the arrows were constructed to show the direction of movement of the vehicles and potential places of danger of the cyclist.
Graphics	Figure 1. Image of Right-Hook Turn.  *Note. This figure shows the scenario of a right-hook turn at a signalized intersection. The cyclist is moving on a straight course and the truck is attempting to make a right-hand turn. The warning label represents the location of risk of a right-hook collision.* Figure 2. Animated Collision Scenario.

	 *Note. This figure shows the scenario of a collision between a through-going cyclist and a right turning semi trailer truck.*
Conclusion	An extension to this graphic design would be to show the field of view of the driver or the blind spots of the semi-trailer truck. Furthermore, a graphic design showing the new field of view from the installed visual system would be helpful in describing the project.

Daily Entry #10: Finish Training

Title	Finish training initial prototype model
Date	October 17, 2022
Signature	*Charles Tang,* 
Introduction	Today, I finished training the SSD MobileNet V2 model on the CIMAT dataset for 8 orientation classification. This model was trained for 50,000 iterations with batch size 8 in the Object Detection API. I also set up a RetinaNet training pipeline in my computer and will train it in the following days.
Design Process	1. Update checkpoint. 2. Train model The model was trained on the new updated checkpoint from the previous entry. The training pipeline configuration for the SSD ResNet50 V1 is in Code 1.
Code 1	`./training_demo/models/ssd_resnet50_v1/pipeline.config` `model {` `  ssd {` `  num_classes: 8` `  image_resizer {` `  fixed_shape_resizer {` `  height: 640` `  width: 640` `  }` `  }` `  feature_extractor {` `  type: "ssd_resnet50_v1_fpn_keras"` `  depth_multiplier: 1.0` `  min_depth: 16` `  conv_hyperparams {` `  regularizer {` `  l2_regularizer {` `  weight: 0.00039999998989515007` `  }` `  }` `  initializer {` `  truncated_normal_initializer {` `  mean: 0.0` `  stddev: 0.029999999329447746` `  }` `  }` `  activation: RELU_6` `  batch_norm {`

```
        decay: 0.996999979019165
        scale: true
        epsilon: 0.0010000000474974513
      }
    }
    override_base_feature_extractor_hyperparams: true
    fpn {
      min_level: 3
      max_level: 7
    }
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
      use_matmul_gather: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  box_predictor {
    weight_shared_convolutional_box_predictor {
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 0.00039999998989515007
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
```

```
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      depth: 256
      num_layers_before_predictor: 4
      kernel_size: 3
      class_prediction_bias_init: -4.599999904632568
    }
  }
  anchor_generator {
    multiscale_anchor_generator {
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      scales_per_octave: 2
    }
  }
  post_processing {
    batch_non_max_suppression {
      score_threshold: 9.99999993922529e-09
      iou_threshold: 0.6000000238418579
      max_detections_per_class: 100
      max_total_detections: 100
      use_static_shapes: false
    }
    score_converter: SIGMOID
  }
  normalize_loss_by_num_matches: true
  loss {
    localization_loss {
      weighted_smooth_l1 {
      }
    }
    classification_loss {
      weighted_sigmoid_focal {
        gamma: 2.0
        alpha: 0.25
      }
    }
    classification_weight: 1.0
```

```
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 4
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.03999999910593033
          total_steps: 25000
          warmup_learning_rate: 0.013333000242710114
          warmup_steps: 2000
        }
      }
      momentum_optimizer_value: 0.8999999761581421
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "models/ssd_resnet50_v1/checkpoints/ckpt-7"
  num_steps: 25000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  use_bfloat16: false
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
```

```
  tf_record_input_reader {
    input_path: "annotations/train2.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  shuffle: true
  num_epochs: 1
  tf_record_input_reader {
    input_path: "annotations/test2.record"
  }
}
```

Results

Figure 1. COCO Object Detection Metrics.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.571
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.709
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.662
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.600
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.276
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.594
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.552
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.832
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.838
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.600
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.670
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.853
```

Note. This figure shows the COCO object detection metrics on the updated prototype model. The mAP for this model is 57%, which is an improvement from the previous checkpoint.

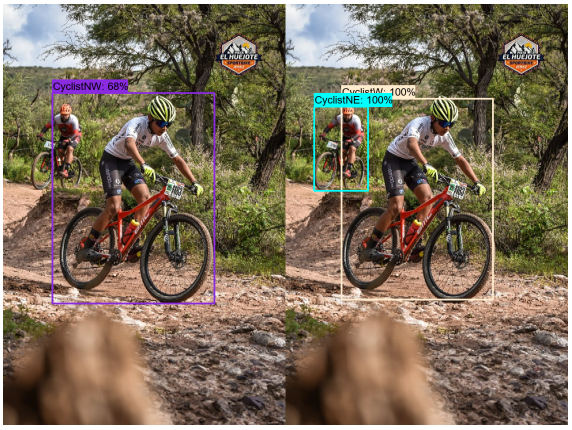
Figure 2. Example Prediction #1.

Note. This figure shows the first prediction of the updated prototype model on a picture with a cyclist in the North direction.

Figure 3. Example Prediction #2.

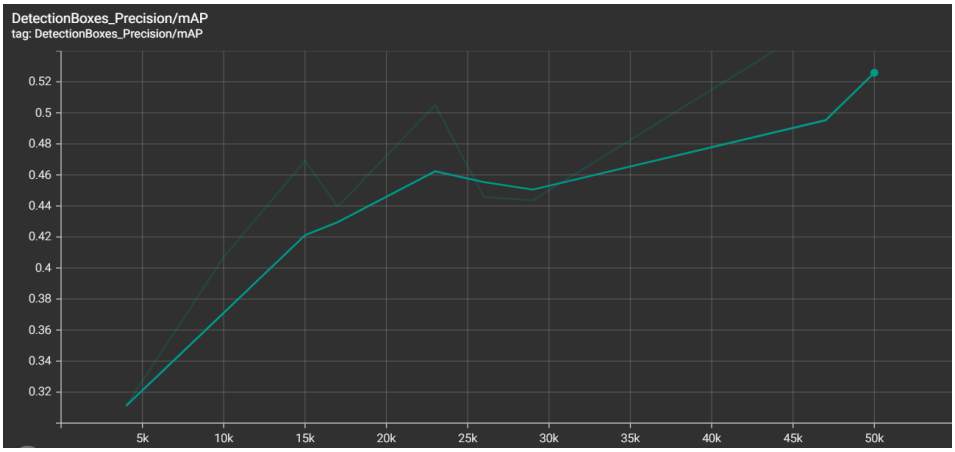
Note. This figure shows the second prediction of the updated prototype model on a picture with cyclists in the Northwest directions.

Figure 4. Example Prediction #3.



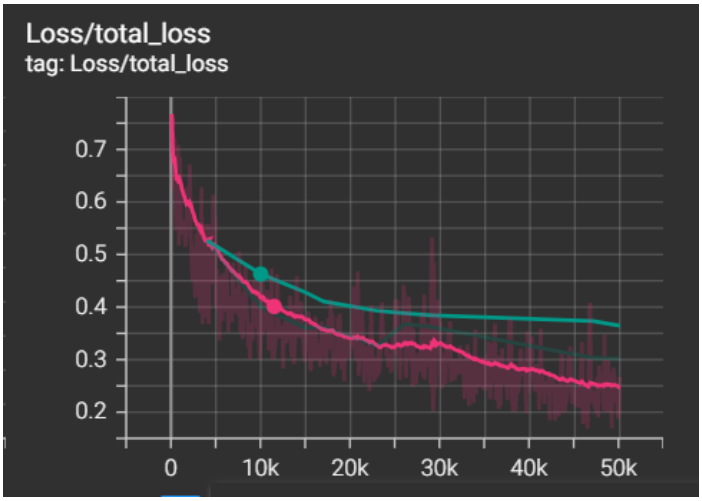
Note. This figure shows the third prediction of the updated prototype model on a picture with cyclists in the West directions.

Figure 5. mAP vs. Number of Steps.



Note. This figure shows the increase in the average precision of the updated prototype model on the test-validation set. This nearly linear increase shows that more training steps should be able to continue improving model performance.

Figure 6. Loss vs. Steps.



Note. This figure shows the decrease in the total loss (classification + localization loss) of the updated prototype model on the test-validation set.

Data Analysis	The AP has been continually increasing, which suggests that I should continue to train it to 75K or 100K iterations. In addition, loss has been continually decreasing. Some future work to do is to continue to train this and to try out some other SSD models, such as the SSD ResNet model. Furthermore, the mismatched predictions show issues when differentiating East and West cyclists, so the data augmentation parameter in the pipeline configuration file was removed. Something this model performed well on was detecting the directions of the North and NW facing cyclists. This might mean that there are more cyclist instances of those classes and the dataset would need to be rebalanced in order to improve accuracy on all classes.

Conclusion	Future work may also be to expand the dataset to prevent ML bias as well as covering various other cyclist instances.

Daily Entry #11: Begin Training ResNet

Title	Begin training SSD + ResNet50 second prototype model
Date	October 18, 2022
Signature	*Charles Tang,*
Introduction	Today, I started training the SSD ResNet50 model on the CIMAT dataset for 8 orientation classification.
Methods	This model was trained for 6,000 iterations with batch size 4 in the Object Detection API. The pipeline configuration file is shown in Code 1 of the previous entry.
Results	Figure 1. COCO Object Detection Metrics. Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100] = 0.302 Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100] = 0.498 Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100] = 0.345 Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100] = 0.075 Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.126 Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.316 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1] = 0.394 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10] = 0.572 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100] = 0.594 Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100] = 0.300 Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.444 Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.605 *Note. This figure shows the mAP and AR of the model from training 6,000 iterations.* Figure 2. Example Prediction #1  *Note. This figure shows the first prediction of the new prototype model on a picture with cyclists in the East directions.* Figure 3. Example Prediction #2

	 *Note. This figure shows the second prediction of the new prototype model on a picture with cyclists in the Northwest directions.*
Data Analysis	This was trained much slower with lower batch size because this model has twice the number of parameters than a SSD MobileNet. Each step of batch size 4 took around 13-15 seconds, compared to a batch size 8 and step time 7 seconds on the MobileNet. Since this model is larger than the MobileNet, I am hoping for this to perform much better. This model seems to be doing much better than the MobileNetV2 model after only 5,000 iterations. I will continue to train this model for another week or two up to 25,000 or 50,000 iterations. The high accuracy may be due to the greater model size.
Conclusion	In conclusion, this model seems to be able to differentiate between cyclist directions better than the MobileNetV2. A further consideration is to try FasterRCNN models, however they may be too slow for real-time inferences.

Daily Entry #12: Continue Training

Title	Continue training SSD ResNet50 prototype model
Date	October 20, 2022
Signature	*Charles Tang,*
Introduction	Today, I continued training the SSD ResNet50 model on the CIMAT dataset for 8 orientation classification. This model was trained for 25,000 iterations with batch size 4 in the Object Detection API. The configuration files were moved to a more powerful laptop with 32Gb RAM and AMD Ryzen Pro 5 CPU. Each training step was about twice as fast.
Methods	1. Train model. I used a new laptop computer with the specifications listed above. 2. Test model.
Results	Figure 1 Results from Object Detection COCO Metrics.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.664
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.823
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.765
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.500
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.427
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.688
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.560
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.819
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.824
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.500
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.661
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.840
```

Note. This table shows the precision of the trained model on detecting cyclists.

Figure 2. Example Prediction #1

Note. This prediction shows the results of the object detection model on an image with two north-facing cyclists.

Figure 3. Example Prediction #2

Note. This prediction shows the results of the object detection model on an image with a west-facing cyclist.

Figure 4. Example Prediction #3

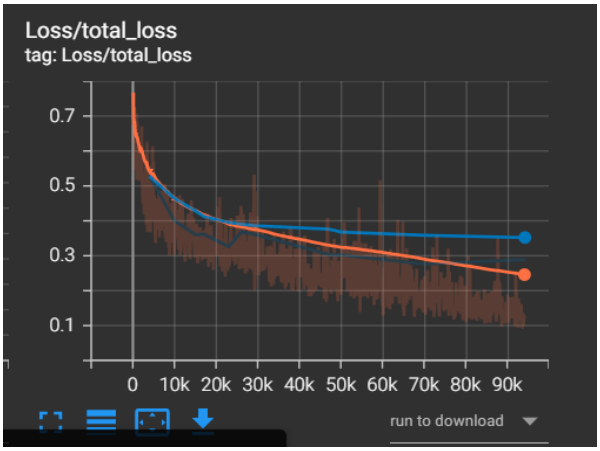
Note. This prediction shows the results of the object detection model on an image with northwest-facing cyclists.

Figure 5. Example Prediction #4

Note. This prediction shows the results of the object detection model on an image with two southeast-facing cyclists.

Data Analysis	These predictions shown by the model were very strong and could most likely be deployed into the real-time model. The mAP is very strong, especially for large objects, and training only 25,000 iterations proves high accuracy. In addition, this model is relatively small and portable, and may be able to be used for real-time inferences.
Conclusion	I will continue training this up to 50,000 iterations and will make a decision on continuing to train from there. Furthermore, testing on the Google Coral Dev Board with a tflite graph would help determine if this model can be used for real-time inferences or not.

Daily Entry #13: Continue Training MobileNet

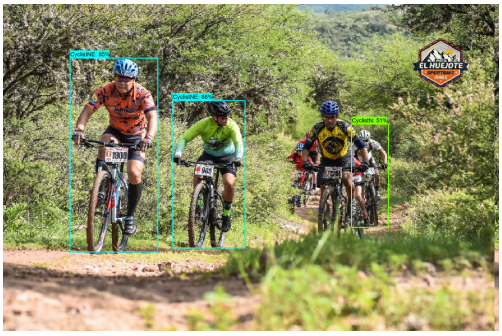
Title	Continue training SSD MobileNet V2 model
Date	October 22, 2022
Signature	*Charles Tang,*
Introduction	The SSD MobileNet V2 model was trained to 94,000 iterations. I continued to compare the results to previous evaluations.
Methods	1. Run the training commands. 2. Test on COCO Object Detection Metrics.
Results	Figure 1. Object Detection Accuracy Metrics on New Model. Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100] = 0.688 Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100] = 0.840 Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100] = 0.792 Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100] = 0.600 Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.334 Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.713 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1] = 0.566 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10] = 0.833 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100] = 0.839 Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100] = 0.600 Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.678 Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.853 *Note. This figure shows the precision metrics on the new model.* Figure 2. Loss Results.  *Note. This figure shows the new loss results up to the 100k step.* Figure 3. Example Prediction #1.

Note. This prediction shows the results of the object detection model on an image with southeast facing cyclists.

Figure 4. Example Prediction #2.

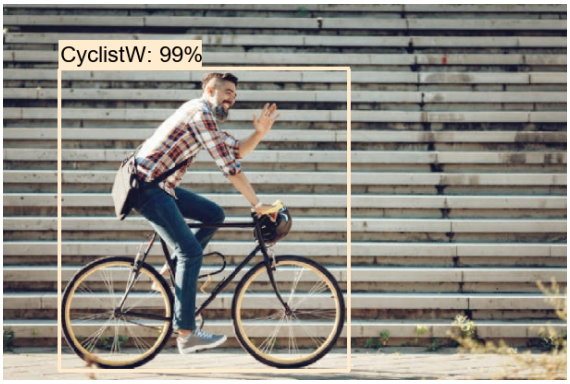
Note. This prediction shows the results of the object detection model on an image with north facing cyclists.

Figure 5. Example Prediction #3.



Note. This prediction shows the results of the object detection model on an image with northeast facings.

Figure 6. Example Prediction #4.

	 *Note. This prediction shows the results of the object detection model on an image with southeast facing cyclists.*
Data Analysis	The mAP improved slightly to 0.68 for IoU 0.5. The loss seems to begin to asymptote, showing that the model can be finished with training. The example detections are promising, as most cyclist instances were detected and classified correctly. The mAP metrics on this model surpassed that of the paper by Garcia-Venegaz et al (2020).
Conclusion	A further consideration is to try single-class cyclist detection in order to improve accuracy even further. I will begin setting up the Google coral board environment to start deploying the model. The example predictions showed strong performance as well.

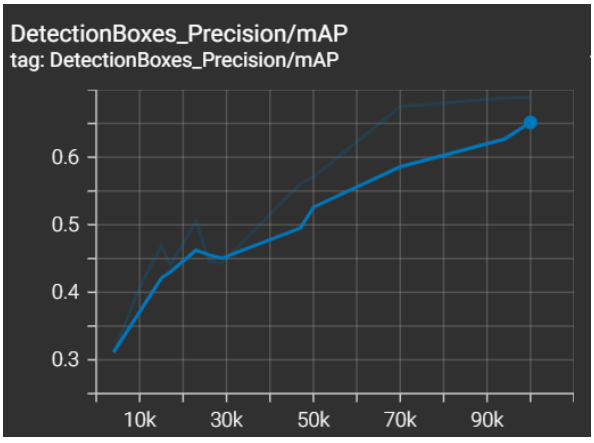
Daily Entry #14: Finish Training MobileNet

Title	Finish training SSD MobileNet V2 model
Date	October 24, 2022
Signature	*Charles Tang,*
Introduction	This SSD MobileNet V2 model was trained to 100,000 iterations. The model achieved 69% mAP on all bicycle objects from the testing dataset. This model was then exported and converted to a tensorflow lite runnable graph. Example predictions and loss results are listed below.
Methods	1. Train object detection model for 6,000 more steps. 2. Evaluate on COCO Object Detection Metrics.
Results	Figure 1. Object Detection Metrics Results.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.688
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.838
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.790
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.600
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.321
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.716
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.568
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.835
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.841
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.600
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.679
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.855
```

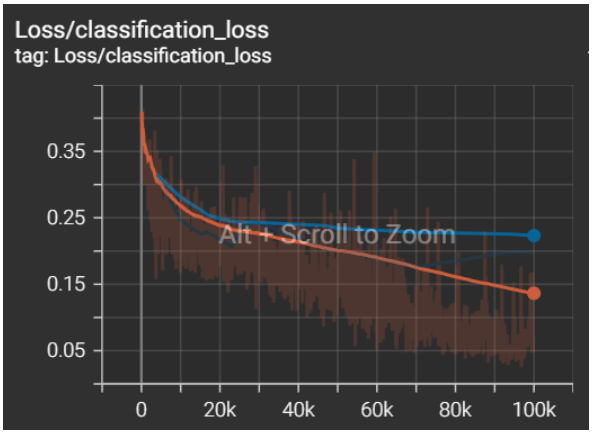
Note. This table shows the average precision metrics over a variety of IoUs and the average recall metrics.

Figure 2. mAP Over Step Count.



Note. This graph shows the growth in mAP over training steps.

Figure 3. Loss over 100,000 Steps.



Note. This graph shows the decrease in loss over the number of steps.

Figure 4. Example Prediction #1.

Note. This figure shows a prediction made by the object detection model on a cyclist moving in the West direction.

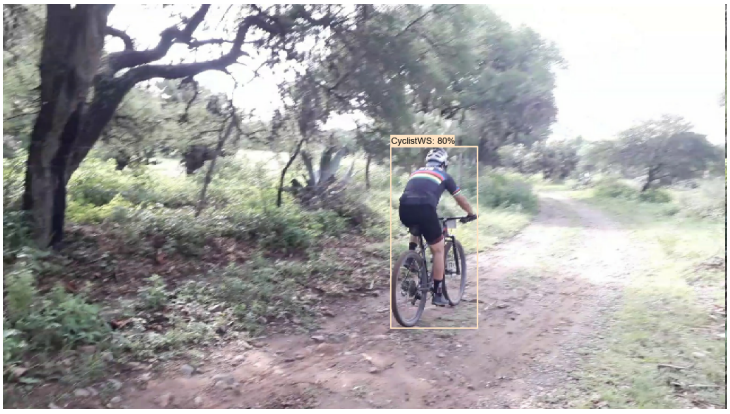
Figure 5. Example Prediction #2.

Note. This figure shows a prediction made by the object detection model on cyclists moving in the South direction.

Figure 6. Example Prediction #3.

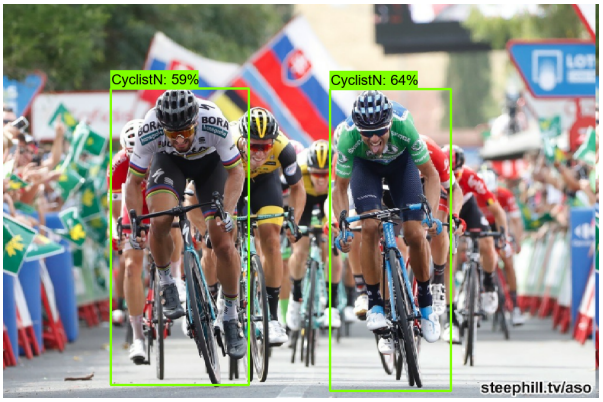
Note. This figure shows a prediction made by the object detection model on a cyclist moving in the North direction.

Figure 7. Example Prediction #4.

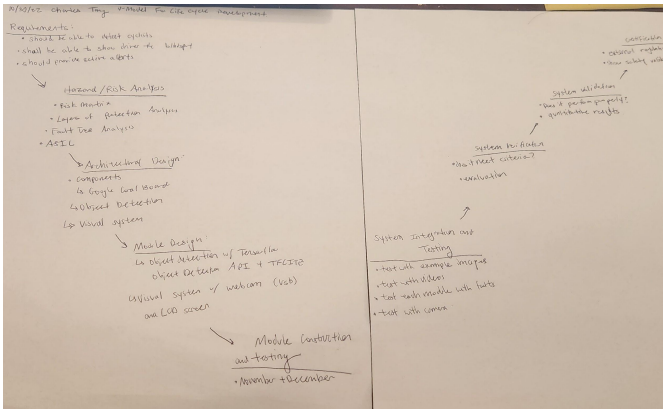


Note. This figure shows a prediction made by the object detection model on a cyclist moving in the Southwest direction.

Figure 8. Example Prediction #5.

	 *Note. This figure shows a prediction made by the object detection model on cyclists moving in the North direction.* Exported Model: https://drive.google.com/drive/folders/1qaDXZcYXr33GZ82jaFJYGgie8DeYSseb?usp=sharing
Data Analysis	The mAP performs up to standard as measured in the On the safety of vulnerable road users by cyclist orientation detection using Deep Learning. The detection results look strong, especially there is an improvement in South facing cyclists. However, the most important results would be the North facing cyclists as these would be the most vulnerable cyclists in semi-trailer truck blindspots. The example predictions also look promising as almost all cyclist instances were detected and classified correctly.
Conclusion	Future work to be done is to use the exported model on the Google Coral Dev Board and to test it for speed as well as efficacy in real-time.

Daily Entry #15: V-Model

Title	V-Model for Life Cycle Development
Date	October 31, 2022
Signature	*Charles Tang,*
Introduction	During the fourth STEM meeting of B-Term, Mr. Medeiros recommended that I investigate safety-critical systems and how the blind spot apparatus is a safety-critical system. The IEC 61508 specifications lay out guidelines for developing safety-critical systems.
Methods	The V-Model for Life Cycle Development was designed to layout the process of developing the blind-spot apparatus and testing it to meet the requirements of a safety-critical system. Furthermore, this model was developed with the support of IEC 61508 standards.
Results	Figure 1. V-Model for Lifecycle Development of Safety Critical Systems  *Note. This plan outlines the main parts of the development of the blind spot safety apparatus.*
Data Analysis	The stages fall into the order: Requirements, Risk Analysis, Design, Testing, and Validation. Currently, I am in the risk analysis through design stages. This model will support my process in outlining specific goals in what to achieve when developing my system.
Conclusion	Based on the IEC 61508 guidelines, I will need to analyze the safety integrity level of the system and what requirements and methods I should use to validate the integrity of my safety-critical system. I will use an estimated probability-based risk analysis or layers of protection analysis.

Daily Entry #16: ASIL

Title	ASIL Risk Analysis Estimation
Date	November 1, 2022
Signature	*Charles Tang,*
Introduction	Today I am estimating the potential risk this safety-critical system will deal with by using the ASIL safety index.
Methods	The ASIL safety index is assigned for every hazardous event that could occur with the device. In my application, the hazardous event would be the collision between a cyclist and a motor vehicle. The safety index would be determined in the following table based on the severity, controllability, and possibility of a collision. The range of severity, controllability, and exposure can be found below. ● In a hazardous event: ○ Severity - how severe of an injury a dangerous event can cause relating to a vehicle ■ S0 = no injury ■ S1 = minor injury ■ S2 = severe/life-threatening injury ■ S3 = Life-threatening/fatal injuries ○ Exposure - how likely is an injury likely to happen for any hazardous event ■ E0 = incredibly unlikely ■ E1 = Very low probability ■ E2 = Low probability ■ E3 = Medium probability ■ E4 = High probability (injury would happen under most events) ○ Controllability - how likely can a driver prevent this hazardous event ■ C0 - almost always controllable ■ C1 - easily controllable ■ C2 - normally controllable (most drivers can prevent injuries) ■ C3 - uncontrollable / very hard to control Table 1. ASIL Estimation Table.

Severity	Exposure	Controllability			
		C0	C1	C2	C3
S0	E1	QM	QM	QM	QM
	E2	QM	QM	QM	QM
	E3	QM	QM	QM	QM
	E4	QM	QM	QM	A
S1	E1	QM	QM	QM	QM
	E2	QM	QM	QM	QM
	E3	QM	QM	QM	A
	E4	QM	QM	A	B
S2	E1	QM	QM	QM	QM
	E2	QM	QM	QM	A
	E3	QM	QM	A	B
	E4	QM	A	B	C
S3	E1	QM	QM	QM	A
	E2	QM	QM	A	B
	E3	QM	A	B	C
	E4	A	B	C	D

Note. This table allows for estimation of an ASIL index based on the three main factors.

Results

Table 2. ASIL Estimation.

Severity	Exposure	Controllability	
		C2	C3
S2	E1	QM	QM
	E2	QM	A
	E3	A	B
	E4	B	C

Note. Filled out table columns and rows for determining ASIL.

Data Analysis

This showed that the ASIL level for this safety-critical design is B, which corresponds to Safety Index Level (SIL) level 2.

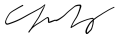
Conclusion

The results were comparable to the classifications of various other automotive components as outlined in ISO 26262. Furthermore, this ASIL level outlined the specifications and coding standards that I will have to follow throughout this project. The next step in this project is to align the current work done so far and to revise the objectives in order to correspond with this risk evaluation.

Daily Entry #17: Setup TF1

Title	Setup new Object Detection API training pipeline for Tensorflow 1
Date	November 6, 2022
Signature	*Charles Tang,*
Introduction	To train models that will run on a Coral Dev Board Edge TPU, it is necessary to use the Tensorflow 1 Object Detection API with 8 bit quantized models.
Methods	I followed the instructions as outlined in the Tensorflow 1 Object Detection API as linked below. https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/tensorflow-1.14/ I configured a file tree as shown below. C:. └───training_demo ├───annotations ├───pre-trained-model │ ├───ssdlite_mobiledet_edgetpu │ └───ssd_mobilenet_v2_quantized └───training Furthermore, I configured the Tensorflow 1 Object Detection module with the respective python packages and downloaded two 8 bit quantized TPU compatible tflite object detection models.
Conclusion	The next step in the development process is to attempt to train a model with this training environment. If this environment poses an obstacle, training a model using Google Colab may be the next step.

Daily Entry #18: Colab Training

Title	Train on Google Colab training notebook
Date	November 9, 2022
Signature	*Charles Tang,* 
Introduction	Since training quantized models is no longer supported in the Object Detection Tensorflow 1 training pipeline, I will follow the instructions as outlined in the Google Coral website and use the Colab training tutorials to construct a tflite-quantized trained model graph.
Methods	1. I set up the training environment using Google Colaboratory to train a SSD Efficient Det Tensorflow Lite model. 2. I uploaded the training and testing record files to google drive and mounted the colab notebook to be able to access Gdrive files. 3. I then configured the model to train for 15 epochs and batch-size 4. I froze the layers containing "efficientnet" in the layer title to speed up the training process. 4. Then, I trained the model over the course of two hours. Additionally, Google Colab cores were purchased to use higher-end GPUs to speed up training. The colaboratory file is linked below. https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-orientation/Try_1_EfficientDet_Lite_detector_for_the_Edge_TPU.ipynb 5. Finally, the tflite graph file was converted to an Edge TPU tflite quantized model for compatibility with the Coral Dev Board.
Results	Figure 1. Object Detection Metrics On The New Model.

	{'AP': 0.45852143, 'AP50': 0.56443346, 'AP75': 0.517839, 'APs': 0.005194805, 'APm': 0.20941214, 'APl': 0.4815102, 'ARmax1': 0.5119225, 'ARmax10': 0.7649619, 'ARmax100': 0.78073084, 'ARs': 0.4, 'ARm': 0.49727932, 'ARl': 0.807394, 'AP_/CyclistN': 0.64601207, 'AP_/CyclistNE': 0.23612733, 'AP_/CyclistE': 0.6230924, 'AP_/CyclistSE': 0.25653276, 'AP_/CyclistS': 0.49380535, 'AP_/CyclistWS': 0.47254133, 'AP_/CyclistW': 0.4827137, 'AP_/CyclistNW': 0.45734638} *Note. This figure shows the AP precision metrics on the newly trained model.* TFLite Model Graph - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-orientation/efficientdet-lite-bikes.tflite Edge TPU Tflite quantized graph - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-orientation/efficientdet-lite-bikes.tflite
Data Analysis	The results and accuracy of the model seem poor, especially in cyclists facing directions other than N, E, S, and W. Furthermore, the Colaboratory training process is much slower and costly than training locally. A much higher accuracy would be needed in producing a final product, which means that future work should be to retrain other models with different hyperparameters and training procedures to improve accuracy.
Conclusion	Future work would be to consider single-class cyclist detection to improve accuracy. Furthermore, it would be future work to run this google colab file on local computational resources or that of the ARC at WPI.

Daily Entry #19: Further Colab Training

Title	Train on Google Colab training notebook part 2
Date	November 13, 2022
Signature	*Charles Tang,*
Introduction	Today, I trained another prototype model using transfer learning with the same methods from the previous daily entry.
Methods	I set up a new but similar training environment to the previous daily entry. The model was trained on the eight orientations CIMAT cyclist dataset benchmark with the EfficientDetLite1 model. The model was trained for 15 epochs, batch size 4, and frozen initial layers. This would mean that the transfer learning would only occur on the final layers. Colab File: https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-orientation/Try_1_EfficientDet_Lite_detector_for_the_Edge_TPU.ipynb Finally, the tflite output file was converted to an edgetpu compatible eight-bit quantized graph for deployment.
Results	Figure 1. Object Detection Metrics on New Model. 738/1000 [==========] {'AP': 0.43024522, 'AP50': 0.52810025, 'AP75': 0.49537733, 'APs': 0.011764706, 'APm': 0.18955281, 'APl': 0.45132405, 'ARmax1': 0.3983129, 'ARmax10': 0.5613, 'ARmax100': 0.5622888, 'ARs': 0.4, 'ARm': 0.2860019, 'ARl': 0.5821052, 'AP_/CyclistN': 0.5923776, 'AP_/CyclistNE': 0.47888288, 'AP_/CyclistE': 0.27259472, 'AP_/CyclistSE': 0.23183042, 'AP_/CyclistS': 0.5153335, 'AP_/CyclistWS': 0.46759295, 'AP_/CyclistW': 0.31650394, 'AP_/CyclistNW': 0.56684583} *Note. This figure shows the AP precision metrics on the newly trained model.* TFLITE File: https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-orientation/efficientdet-lite-bikes.tflite

	Edge TPU Optimized File: https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-orientation/efficientdet-lite-bikes_edgetpu.tflite
Data Analysis	This new model trained only on the last layers of the model performed slightly less precisely than the previously trained on the previous model. The mAP of 45% would not be adequate to deploy onto safety critical systems, which would mean that either the model needs to be improved or the dataset needs to be refined. The significantly higher mAP in the CyclistN class shows that the dataset may not be balanced well. Furthermore, since neither the mAP 50:95 average nor the mAP at .5 IoU is very high, it means that the model did not learn to generalize well over the cyclists.
Conclusion	Future work would include implementing single-class cyclist detection as well as getting example predictions on the cyclists from these poor performing models.

Daily Entry #20: Single-Class Dataset

Title	Create dataset with one class – 'Cyclist'
Date	November 15, 2022
Signature	*Charles Tang,*
Introduction	Today, I created a new tfrecord dataset file that would only be dependent on one class, which would be called 'Cyclist'. Since the models performed poorly on the eight orientation dataset, it may be a better choice to train the models with only one class to make detections more accurate.
Methods	I used the Tensorflow Object Detection API to generate a new dataset from the CIMAT orientations dataset. The new generate_tfrecord.py file is shown below, which was modified to generate a new tfrecord dataset file.
Code 1	./training_demo/generate_tfrecord.py

```
import os
import glob
import pandas as pd
import io
import xml.etree.ElementTree as ET
import argparse

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow
logging (1)
import tensorflow.compat.v1 as tf
from PIL import Image
from object_detection.utils import dataset_util, label_map_util
from collections import namedtuple

# Initiate argument parser
parser = argparse.ArgumentParser(
    description="Sample TensorFlow XML-to-TFRecord converter")
parser.add_argument("-x",
                    "--xml_dir",
                    help="Path to the folder where the input .xml
files are stored.",
                    type=str)
parser.add_argument("-l",
                    "--labels_path",
                    help="Path to the labels (.pbtxt) file.",
type=str)
parser.add_argument("-o",
                    "--output_path",
```

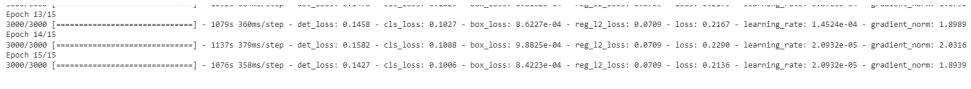
```
                    help="Path of output TFRecord (.record) file.",
type=str)
parser.add_argument("-i",
                    "--image_dir",
                    help="Path to the folder where the input image
files are stored. "
                         "Defaults to the same directory as
XML_DIR.",
                    type=str, default=None)
parser.add_argument("-c",
                    "--csv_path",
                    help="Path of output .csv file. If none
provided, then no file will be "
                         "written.",
                    type=str, default=None)

args = parser.parse_args()

if args.image_dir is None:
    args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
    """Iterates through all .xml files (generated by labelImg) in a
given directory and combines
    them in a single Pandas dataframe.

    Parameters:
    ----------
    path : str
        The path containing the .xml files
    Returns
    -------
    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        filename = root.find('filename').text
        width = int(root.find('size').find('width').text)
        height = int(root.find('size').find('height').text)
```

```
        for member in root.findall('object'):
            bndbox = member.find('bndbox')
            value = (filename,
                     width,
                     height,
                     member.find('name').text,
                     int(bndbox.find('xmin').text),
                     int(bndbox.find('ymin').text),
                     int(bndbox.find('xmax').text),
                     int(bndbox.find('ymax').text),
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def class_text_to_int(row_label):
    # return label_map_dict[row_label]
    if row_label in ["CyclistN", "CyclistNE", "CyclistE",
"CyclistSE", "CyclistS", "CyclistWS", "CyclistW", "CyclistNW"]:
        return 1

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    print(group.filename)
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename) + ".jpg"), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
```

```
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example =
tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):

    writer = tf.python_io.TFRecordWriter(args.output_path)
    path = os.path.join(args.image_dir)
    examples = xml_to_csv(args.xml_dir)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    print('Successfully created the TFRecord file:
{}'.format(args.output_path))
    if args.csv_path is not None:
```

	examples.to_csv(args.csv_path, index=None) print('Successfully created the CSV file: {}'.format(args.csv_path)) if __name__ == '__main__': tf.app.run()
Results	Training Dataset Record File: https://drive.google.com/file/d/1CIpz0w2N_1P-W1pRZPTRWS_byMqpuMqr/view?usp=sharing Testing Dataset Record File: https://drive.google.com/file/d/1POnTHTYMOQpgyqE2lzHtVeP9XrNC8y-Y/view?usp=sharing
Conclusion	The next step in the training process is to train models on this new dataset using the Colab training environment. Then, I can begin the deployment process with the Google Coral Dev Board.

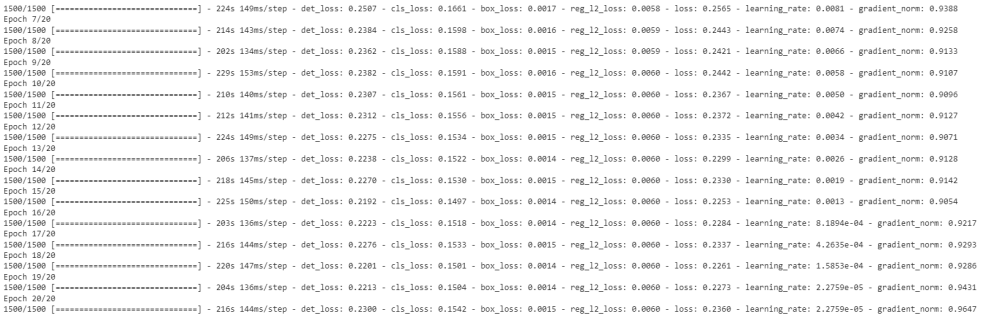
Daily Entry #21: Train Single-Class Model

Title	Train object detection model on one class dataset
Date	November 17, 2022
Signature	*Charles Tang,*
Introduction	Today, I trained an object detection model on the new dataset to detect cyclists without orientation classes.
Methods	I set up the training environment using Google Colaboratory to train a SSD Efficient Det Tensorflow Lite model. I used the new training and testing record files on google drive and mounted the colab notebook to be able to access Gdrive files. I then configured the model to train for 15 epochs and batch-size 4. I froze the non-final-layers to speed up the transfer learning process. Then, I trained the model over the course of four hours. Additionally, Google Colab cores were purchased to use higher-end GPUs to speed up training. Finally, the produced TFLite graph was converted to be Edge TPU compatible. Google Colab File - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-noorientation/Try%203_%20No%20orientation%20-%20last%20layers
Results	Figure 1. Object detection metrics. {'AP': 0.80249125, 'AP50': 0.97707677, 'AP75': 0.9283836, 'APs': 0.011864407, 'APm': 0.4651496, 'APl': 0.8242985, 'ARmax1': 0.4780303, 'ARmax10': 0.83325756, 'ARmax100': 0.83560604, 'ARs': 0.7, 'ARm': 0.6148515, 'ARl': 0.854023, 'AP_/Cyclist': 0.80249125} *Note. This figure shows the accuracy metrics outlined by the COCO Object Detection paper.* Figure 2.  Epoch 13/15 3000/3000 [==========] - 1079s 360ms/step - det_loss: 0.1458 - cls_loss: 0.1027 - box_loss: 8.6227e-04 - reg_l2_loss: 0.0709 - loss: 0.2167 - learning_rate: 1.4524e-04 - gradient_norm: 1.8989 Epoch 14/15 3000/3000 [==========] - 1137s 379ms/step - det_loss: 0.1582 - cls_loss: 0.1088 - box_loss: 9.8825e-04 - reg_l2_loss: 0.0709 - loss: 0.2290 - learning_rate: 2.0932e-05 - gradient_norm: 2.0316 Epoch 15/15 3000/3000 [==========] - 1076s 358ms/step - det_loss: 0.1427 - cls_loss: 0.1006 - box_loss: 8.4223e-04 - reg_l2_loss: 0.0709 - loss: 0.2136 - learning_rate: 2.0932e-05 - gradient_norm: 1.8939

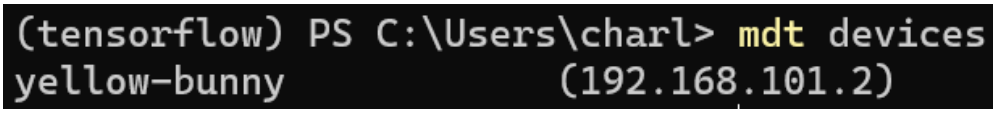
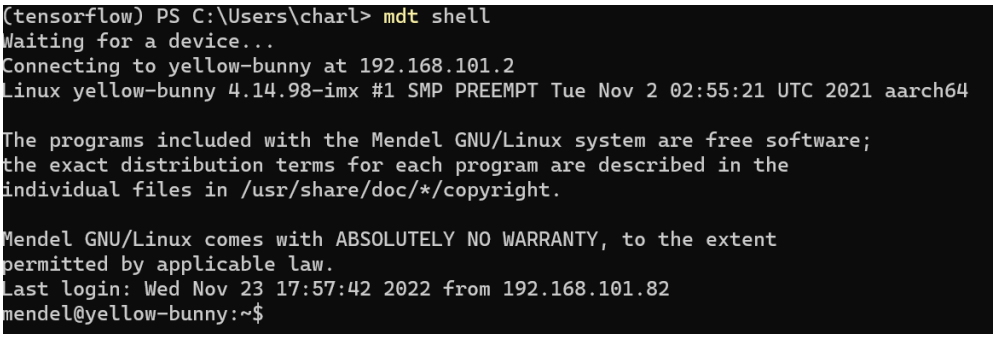
	Note. This figure shows the classification, localization, and total loss of the model over the last three epochs of the training process. Edge TPU Compiled TF Lite Graph - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-final-layers-train-noorientation/efficientdet-lite-bikes_edgetpu.tflite
Data Analysis	This new model proves much more accurate and satisfies the criteria of being at least 80% accurate. This increased accuracy is most likely due to the decrease in the number of classes, thus making the object detection task simpler and much more effective. Furthermore, the mAP for larger objects is almost 100%, which shows that this model would perform exceptionally well if a cyclist was very close to the blind spot apparatus.
Conclusion	The next steps in this project is to deploy this prototype model into the Google Coral Dev Board and test its capabilities with a WebCam. The Google Coral Dev Board needs to be set up so that test predictions can be made. Furthermore, the FPS and speed of the model needs to be tested on the Coral Dev Board, which should be very high due to the efficiency of the Edge TPU.

Daily Entry #22: Train Another Single-Class Detection

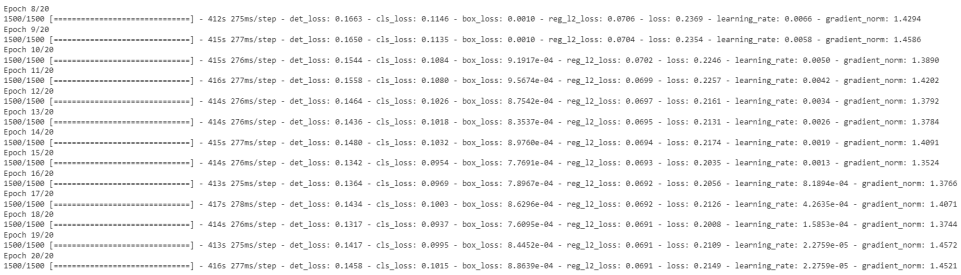
Title	Train second object detection model on one class dataset
Date	November 21, 2022
Signature	*Charles Tang,*
Introduction	Today, I trained another object detection model for the single-class cyclist detection task.
Methods	I set up the training environment using Google Colaboratory to train a SSD Efficient Det Tensorflow Lite Spec 2 model. I used the single-class training and testing record files on google drive and mounted the colab notebook to be able to access Gdrive files. I then configured the model to train for 20 epochs and batch-size 8. I froze the non-final-layers to speed up the transfer learning process. Then, I trained the model over the course of four hours. Additionally, Google Colab cores were purchased to use higher-end GPUs to speed up training. Finally, the produced TFLite graph was converted to be Edge TPU compatible. Google Colab File - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-spec2-final-layers-train-noorientation/Try%204_%20No%20orientation%20-%20all%20layers%20-%20spec2
Results	Figure 1. Object Detection Metrics. `12/16 [====================>........] - ETA: 2s` `{'AP': 0.77642953,` `'AP50': 0.9714287,` `'AP75': 0.90911853,` `'APs': 0.013333334,` `'APm': 0.46575785,` `'APl': 0.7975721,` `'ARmax1': 0.46698046,` `'ARmax10': 0.8257784,` `'ARmax100': 0.844895,` `'ARs': 0.2,` `'ARm': 0.67706424,` `'ARl': 0.85979545,` `'AP_/Cyclist': 0.77642953}` *Note. This figure shows the accuracy metrics outlined by the COCO Object Detection paper.*

	Figure 2. Loss Over Time of the Training Process.  1500/1500 [==========] - 224s 149ms/step - det_loss: 0.2507 - cls_loss: 0.1661 - box_loss: 0.0017 - reg_l2_loss: 0.0058 - loss: 0.2565 - learning_rate: 0.0081 - gradient_norm: 0.9388 Epoch 7/20 1500/1500 [==========] - 214s 143ms/step - det_loss: 0.2384 - cls_loss: 0.1598 - box_loss: 0.0016 - reg_l2_loss: 0.0059 - loss: 0.2443 - learning_rate: 0.0074 - gradient_norm: 0.9258 Epoch 8/20 1500/1500 [==========] - 202s 134ms/step - det_loss: 0.2362 - cls_loss: 0.1588 - box_loss: 0.0015 - reg_l2_loss: 0.0059 - loss: 0.2421 - learning_rate: 0.0066 - gradient_norm: 0.9133 Epoch 9/20 1500/1500 [==========] - 229s 153ms/step - det_loss: 0.2382 - cls_loss: 0.1591 - box_loss: 0.0016 - reg_l2_loss: 0.0060 - loss: 0.2442 - learning_rate: 0.0058 - gradient_norm: 0.9107 Epoch 10/20 1500/1500 [==========] - 210s 140ms/step - det_loss: 0.2307 - cls_loss: 0.1561 - box_loss: 0.0015 - reg_l2_loss: 0.0060 - loss: 0.2367 - learning_rate: 0.0050 - gradient_norm: 0.9096 Epoch 11/20 1500/1500 [==========] - 212s 141ms/step - det_loss: 0.2312 - cls_loss: 0.1556 - box_loss: 0.0015 - reg_l2_loss: 0.0060 - loss: 0.2372 - learning_rate: 0.0042 - gradient_norm: 0.9127 Epoch 12/20 1500/1500 [==========] - 224s 149ms/step - det_loss: 0.2275 - cls_loss: 0.1534 - box_loss: 0.0015 - reg_l2_loss: 0.0060 - loss: 0.2335 - learning_rate: 0.0034 - gradient_norm: 0.9071 Epoch 13/20 1500/1500 [==========] - 206s 137ms/step - det_loss: 0.2238 - cls_loss: 0.1522 - box_loss: 0.0014 - reg_l2_loss: 0.0060 - loss: 0.2299 - learning_rate: 0.0026 - gradient_norm: 0.9128 Epoch 14/20 1500/1500 [==========] - 218s 145ms/step - det_loss: 0.2270 - cls_loss: 0.1530 - box_loss: 0.0015 - reg_l2_loss: 0.0060 - loss: 0.2330 - learning_rate: 0.0019 - gradient_norm: 0.9142 Epoch 15/20 1500/1500 [==========] - 225s 150ms/step - det_loss: 0.2192 - cls_loss: 0.1497 - box_loss: 0.0014 - reg_l2_loss: 0.0060 - loss: 0.2253 - learning_rate: 0.0013 - gradient_norm: 0.9054 Epoch 16/20 1500/1500 [==========] - 203s 136ms/step - det_loss: 0.2223 - cls_loss: 0.1518 - box_loss: 0.0014 - reg_l2_loss: 0.0060 - loss: 0.2284 - learning_rate: 8.1894e-04 - gradient_norm: 0.9217 Epoch 17/20 1500/1500 [==========] - 216s 144ms/step - det_loss: 0.2276 - cls_loss: 0.1533 - box_loss: 0.0015 - reg_l2_loss: 0.0060 - loss: 0.2337 - learning_rate: 4.2635e-04 - gradient_norm: 0.9293 Epoch 18/20 1500/1500 [==========] - 220s 147ms/step - det_loss: 0.2201 - cls_loss: 0.1501 - box_loss: 0.0014 - reg_l2_loss: 0.0060 - loss: 0.2261 - learning_rate: 1.5853e-04 - gradient_norm: 0.9286 Epoch 19/20 1500/1500 [==========] - 204s 136ms/step - det_loss: 0.2213 - cls_loss: 0.1504 - box_loss: 0.0014 - reg_l2_loss: 0.0060 - loss: 0.2273 - learning_rate: 2.2759e-05 - gradient_norm: 0.9431 Epoch 20/20 1500/1500 [==========] - 216s 144ms/step - det_loss: 0.2300 - cls_loss: 0.1542 - box_loss: 0.0015 - reg_l2_loss: 0.0060 - loss: 0.2360 - learning_rate: 2.2759e-05 - gradient_norm: 0.9647 *Note. This figure shows the classification, localization, and total loss of the model over the training process*. Edge TPU Compiled TF Lite Graph - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-spec2-final-layers-train-noorientation/efficientdet-lite-bikes_edgetpu.tflite
Data Analysis	Although this model is larger and was trained with the same number of epochs, the accuracy was slightly worse than the EfficientDet Lite Spec 1. Furthermore, this model would not perform as fast on the Coral Dev Board because the size of the model exceeds the onboard ram. Out of the loss categories, the detection loss seemed to be the highest, which means that the cyclist features are hard to detect and would need more epochs of training.
Conclusion	The next step in the design process is to deploy these prototype object detection models onto the Google Coral Dev Board and perform some initial module tests with the webcam.

Daily Entry #23: Setup Coral Board

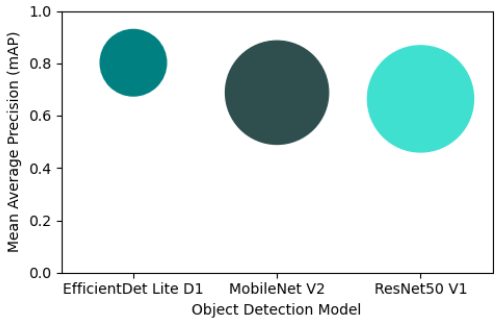
Title	Set up Google Coral Dev Board
Date	November 23, 2022
Signature	*Charles Tang,*
Introduction	To begin deployment and testing with the Google Coral Dev Board, I set up the Google Coral Dev Board.
Methods	I followed the setup instructions as outlined by the Google Coral Dev Board Documentation. First, I flashed the board with Mendel Linux with a MicroSD card. Then, I installed MDT which is used to SSH into the board from another computer. I then connected to the internet and ran some demo applications to prove that the setup was successful. Figure 1. Establishing SSH Connection.  *Note. The MDT package was downloaded to establish an ssh connection over USB C.* Figure 2. Connected to Coral Dev Board via MDT.  *Note. This shows a successful connection.*
Conclusion	The next step in the process is to download the trained model from previous daily entries and set up the camera module and test the combined system.

Daily Entry #24: Train Third Single-Class Detection

Title	Train third object detection model on single-class cyclist detection task
Date	November 24, 2022
Signature	*Charles Tang,*
Introduction	Today, I trained a third object detection model for the single-class cyclist detection task.
Methods	I set up the training environment using Google Colaboratory to train a SSD Efficient Det Tensorflow Lite Spec 1 model. I used the single-class training and testing record files on google drive and mounted the colab notebook to be able to access Gdrive files. I then configured the model to train for 20 epochs and batch-size 8. I did not freeze any layers and trained the model over the course of four hours. Additionally, Google Colab cores were purchased to use higher-end GPUs to speed up training. Finally, the produced TFLite graph was converted to be Edge TPU compatible. Google Colab File - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-all-layers-train-noorientation/Try%205%20-%20no%20orientation%2C%20all%20layers%20-%20spec1
Results	Trained Edge TPU Compatible TFLITE model - https://github.com/charlestang06/BikeDetector/blob/main/efficient-det-all-layers-train-noorientation/efficientdet-lite-bikes_edgetpu.tflite Figure 1. Loss Results.  *Note. This shows the classification, localization, and total loss of the model over the last 11 epochs of the training process.* Figure 2. Accuracy Results.

	12/16 [=====================>.....] - ETA: 2s {'AP': 0.82103777, 'AP50': 0.9777668, 'AP75': 0.9295557, 'APs': 0.012121212, 'APm': 0.50447285, 'APl': 0.84338975, 'ARmax1': 0.4859522, 'ARmax10': 0.85792905, 'ARmax100': 0.8705286, 'ARs': 0.4, 'ARm': 0.6587156, 'ARl': 0.8890637, 'AP_/Cyclist': 0.82103777} *Note. The precision results use the metrics outlined by COCO Object Detection. This is the best performing model so far.*
Data Analysis	This newly performed model is the same size as the model from Daily Entry #21 and performs 2% greater precision in terms of mAP. This model performs stronger in all categories of precision and satisfies the 80% accurate criteria for a cyclist detection model. However, this model did not perform well on smaller objects with a mAP of small objects at 0.012. This could either mean that the dataset did not incorporate many smaller cyclists or was trained enough.
Conclusion	Further work with this model is to deploy it onto the Google Coral Dev Board and test for latency, frames per second, and set up a testing environment.

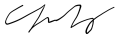
Daily Entry #25: Figure Creation

Title	Create figures and graphs of mAP per model
Date	November 25, 2022
Signature	*Charles Tang,*
Introduction	Today, I created the figures for the December fair and the grant proposal assignment that describe each model trained.
Methods	I used Matplotlib and Pyplot to create the figures. I chose to use a bubble graph representation so it can represent the model size and the accuracy of the model on each given task.
Code 1	`import matplotlib.pyplot as plt` `import numpy as np` `x = ['EfficientDet Lite D1', 'MobileNet V2', 'ResNet50 V1']` `y = [0.802, 0.688, 0.664]` `z = np.array([7, 17, 18], dtype=float)` `colors = ['teal', 'darkslategray' ,'turquoise']` `plt.ylim([0, 1])` `plt.xlim([-0.5, 2.5])` `plt.scatter(x, y, s=z*300,c=colors)` `plt.xlabel("Object Detection Model")` `plt.ylabel("Mean Average Precision (mAP)")` `plt.show()`
Results	Figure 1. Performance of Prototype Cyclist Detection Models.  *Note. The bubble size represents the file size of the final trained model.*

Daily Entry #26: Deploy Model

Title	Deploy initial single-class cyclist detection model onto Coral Dev Board
Date	November 26, 2022
Signature	*Charles Tang,*
Introduction	Today, I deployed one of the cyclist detection models onto the Coral Dev Board and tested it to see if it worked with a webcam.
Methods	I followed the instructions from the Coral Board Documentation to set up an object detection model. 1. Connect to the Coral Dev Board via USB-C connector and MDT shell commands. 2. Download the models from Github on the Dev Board using the following commands. $ export DEMO_FILES="$HOME/demo_files" $ wget -P ${DEMO_FILES}/ https://github.com/charlestang06/BikeDetector/raw/main/efficient-det-final-layers-train-noorientation/bikes-labels.txt $ wget -P ${DEMO_FILES}/ https://github.com/charlestang06/BikeDetector/raw/main/efficient-det-final-layers-train-noorientation/efficientdet-lite-bikes_edgetpu.tflite 3. Connected the camera by USB to the board. 4. Ran the following command v4l2-ctl --list-formats-ext --device /dev/video1 This showed that the input USB webcam ran 30 frames per second at 640x480 view size. 5. Ran the following command. edgetpu_detect \ --source /dev/video1:YUY2:640x480:30/1 \ --model $HOME/demo_files/efficientdet-lite-bikes_edgetpu2.tflite \ --labels $HOME/demo_files/bikes-labels.txt \ --threshold 0.5 6. Observed results on a monitor connected by HDMI.
Conclusion	The results from the monitor seemed strong and the model could run around 10 frames per second at estimate. Some future work would include measuring the frames per second of the output and the latency. Furthermore, I will need to test this model apparatus in real-time with a portable monitor.

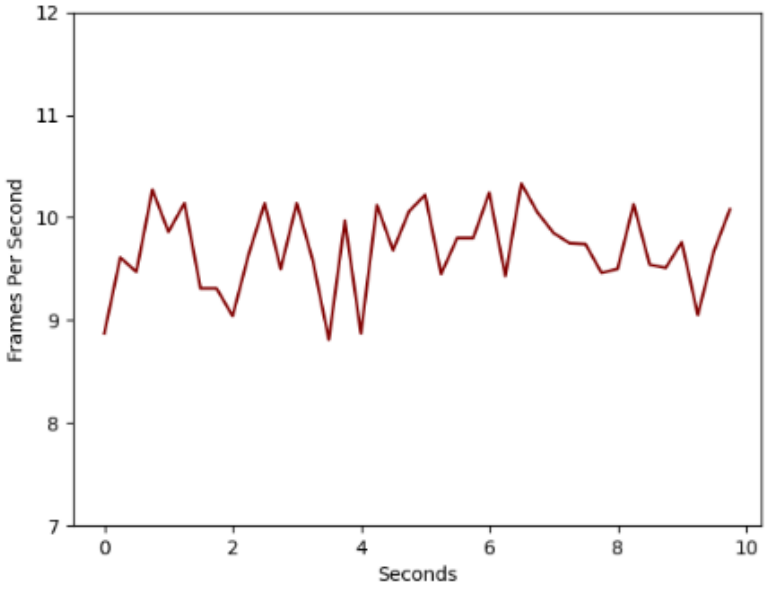
Daily Entry #27: Run Model on Video

Title	Run single-class cyclist detection model on Video File on Coral Dev Board
Date	November 29, 2022
Signature	*Charles Tang,* 
Introduction	Today, I tested the single-class cyclist detection model on a video of cyclists and viewed the results on a monitor.
Methods	1. Pushed video to Coral Dev Board over SSH with `$ mdt push` command 2. Ran below command to show results. edgetpu_detect \ --source $HOME/bikevid.mp4 \ --model $HOME/demo_files/efficientdet-lite-bikes_edgetpu2.tflite \ --labels $HOME/demo_files/bikes-labels.txt \ --threshold 0.2 3. Recorded FPS and Inference Time/Latency over a 10 second window and stored in Table 4. 4. I then formatted this table as a line chart using the following code with the matplotlib package.
Code 1	`import matplotlib.pyplot as plt` `import numpy as np` `# https://pythonguides.com/matplotlib-two-y-axes/` `x = np.arange(0, 10, 0.25)` `fps =` `[8.87,9.61,9.47,10.27,9.86,10.14,9.31,9.31,9.04,9.64,10.14,9.5,10.14,9.58,8.81,9.97,8.87,10.12,9.68,10.06,10.22,9.45,9.8,9.8,10.24,9.43,10.33,10.05,9.85,9.75,9.74,9.46,9.5,10.13,9.54,9.51,9.76,9.05,9.67,10.08]` `plt.xlabel('Seconds')` `plt.ylabel('Frames Per Second')` `plt.plot(x, fps, color = 'maroon', label="Frames Per Second")` `plt.ylim([7, 12])` `plt.show()`
Results	Table 1. FPS and Inference Times.

Time	Inference time (ms)	FPS
0	112.8	8.87
0.25	104.04	9.61
0.75	105.55	9.47
1	97.39	10.27
1.25	101.37	9.86
1.5	98.6	10.14
1.75	107.41	9.31
2	107.44	9.31
2.25	110.6	9.04
2.5	103.79	9.64
2.75	98.58	10.14
3	105.3	9.5
3.25	98.59	10.14
3.5	104.41	9.58
3.75	113.52	8.81
4	100.26	9.97
4.25	112.76	8.87
4.5	98.86	10.12
4.75	103.35	9.68
5	99.38	10.06
5.25	97.84	10.22
5.5	105.77	9.45
5.75	102.05	9.8
6	102	9.8
6.25	97.62	10.24
6.5	106.1	9.43
6.75	96.77	10.33
7	99.55	10.05

Average FPS: 9.7
Average Inference Time: 103.3

Figure 1. FPS of the Single-Class Object Detection Model.

	 *Note. This figure shows the frames per second of the model evaluating on a video file over a 10 second period.*
Data Analysis	The criteria of detecting cyclists within 2 seconds was satisfied by a large margin with an inference of time of around 100 milliseconds. This means that this prototype model successfully satisfies the base criteria and can be tested further in various scenarios. The variability in the frames per second seemed minimal and could be used in real-time scenarios.
Conclusion	The next step in the process is to test it with a webcam and develop a real-time testing scenario.

Daily Entry #28: Run On Startup

Title	Run detection model automatically on startup
Date	November 30, 2022
Signature	*Charles Tang,* 
Introduction	One feature of the device that should be implemented as a blind spot apparatus is automatically running the detection model on the startup. This would prevent any potential users from needing to SSH into the Coral Board and running the commands manually.
Methods	The process of inserting the necessary scripts was outlined by the GitHub link below. https://github.com/google-coral/edgetpu/issues/120 First, I created a file in the Coral Board using VI text editor called detects.service and I pasted in the following contents. [Unit] Description=systemd object detection service After=weston.target [Service] PAMName=login Type=simple User=mendel WorkingDirectory=/home/mendel Environment=DISPLAY=:0 ExecStart=/bin/bash /usr/bin/detect_service.sh Restart=always [Install] WantedBy=multi-user.target I then moved the file to /lib/systemd/system with the following command. $ sudo mv detects.service /lib/systemd/system/detects.service Next, I created a new bash script file called detects.service with the contents below.

<table>
<tr><td></td><td>edgetpu_detect --source /dev/video1:YUY2:640x480:30/1 --model $HOME/demo_files/efficientdet-lite-bikes_edgetpu2.tflite --labels $HOME/demo_files/bikes-labels.txt

I finally ran the below commands to make it executable on startup and also in the write system configuration location.

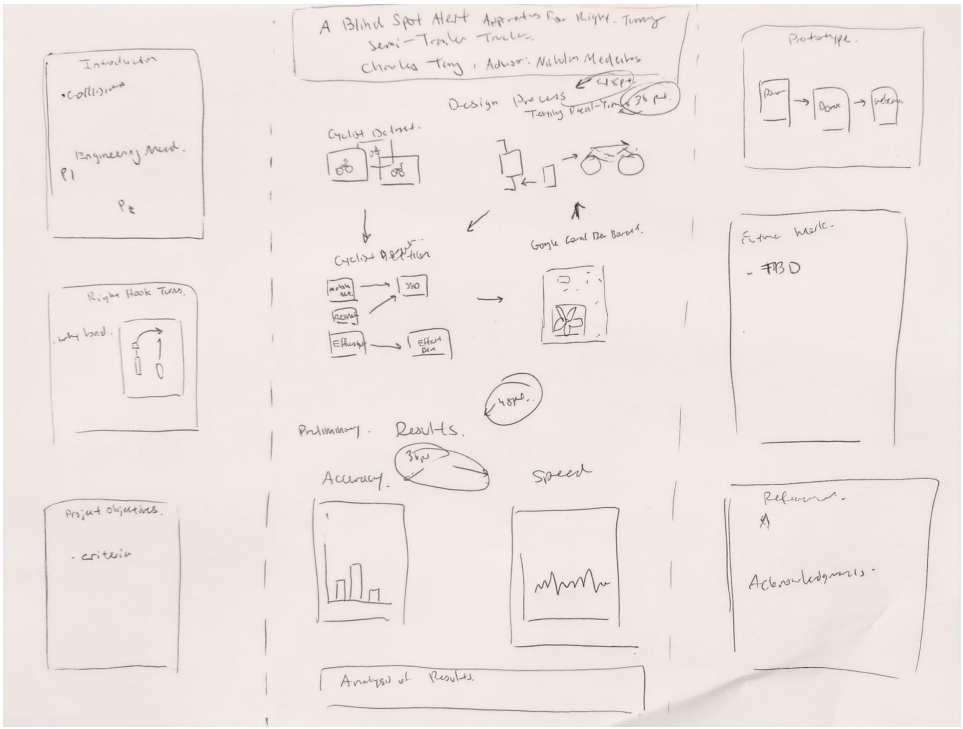
$ sudo chmod u+x detect_service.sh
$ sudo mv detect_service.sh /usr/bin

Finally, to enable or disable this start up service, I can run the following commands.

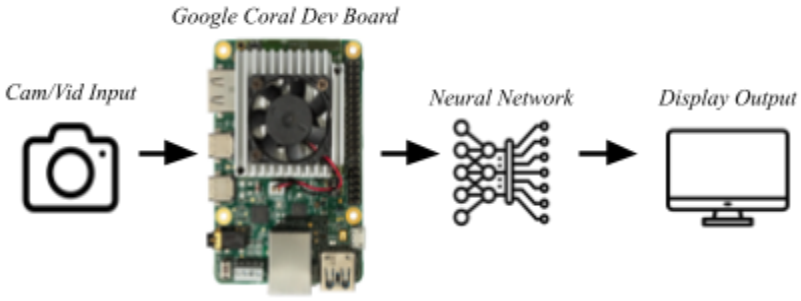
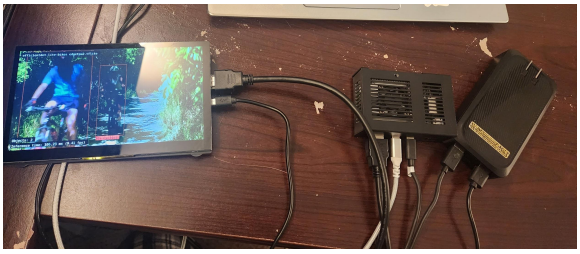
$ sudo systemctl enable detects.service
$ sudo systemctl disable detects.service</td></tr>
<tr><td>Results</td><td>The time to start up is measured from the moment power is sent to the device to the moment the object detection model appears on the GUI.

Table 1. Startup Times.
<table><tr><td>Trial #</td><td>1</td><td>2</td><td>3</td></tr><tr><td>Time to Start Up (s)</td><td>23.2</td><td>21.9</td><td>23.0</td></tr></table>
Average Time to Start Up – 22.4 seconds</td></tr>
<tr><td>Conclusion</td><td>This daily entry’s work made it possible to run auto detection from the power on of the device. The next investigation would include developing a CAD model for the mini-monitor case.</td></tr>
</table>

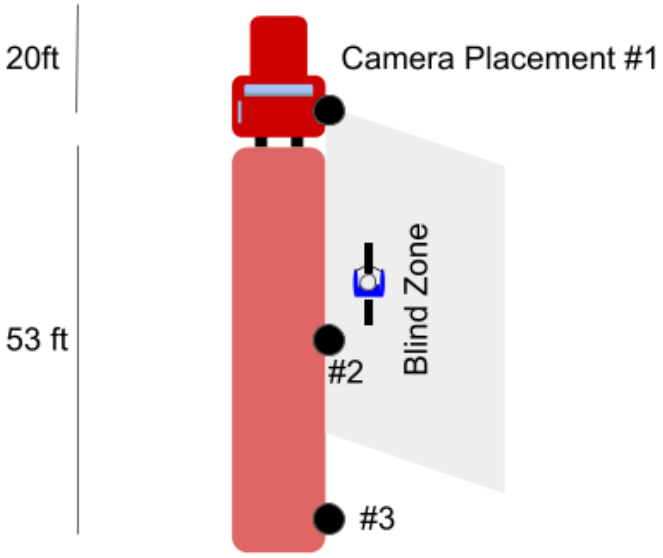
Daily Entry #29: December Fair Design

Title	December Fair poster brainstorm design
Date	December 1, 2022
Signature	*Charles Tang,*
Introduction	In preparation for the December Fair at Mass Academy where we share some preliminary work of our project, we are tasked to create a tri-fold poster.
Designs	Figure 1. Initial Poster Sketch.  *Note. This is a brainstorm sketch of the December Fair poster.*
Conclusion	The next steps would be to design the contents of the poster and assemble the pieces to create the final product.

Daily Entry #30: Prototyping

Title	Prototype design and initial test run
Date	December 3, 2022
Signature	*Charles Tang,*
Introduction	This is the first fully integrated test of all of the modules designed so far.
Methods	Figure 1. High Level Design of Integrated Modules.  We tested each module to determine if it was operational (Yes) or not (No).
Results	Table 1. Module Integration Test. Module | Operational? Video Input | Yes Coral Dev Board | Yes Object Detection CNN | Yes Figure 2. Prototype Apparatus Picture. 
Conclusion	This initial prototype construction and test proves that the prototype apparatus works as expected. Further work would include training better performing deep learning models, creating a CAD case for the monitor, and analyzing each part of the semi-trailer truck in determining the best place to put the camera.

Daily Entry #31: Blind Zone Estimation

Title	Estimate right-hand blind zone of semi-trailer truck and camera placement effectiveness
Date	December 5, 2022
Signature	*Charles Tang,*
Introduction	This daily entry tries to estimate the right-hand blind zone of a semi-trailer truck.
Estimation	Figure 1. Graphical Representation.  *Note. This shows the three possible camera placements to cover the blind zone as shown in the graphical design.* Table 1. Design Matrix to Determine Optimal Camera Placement.

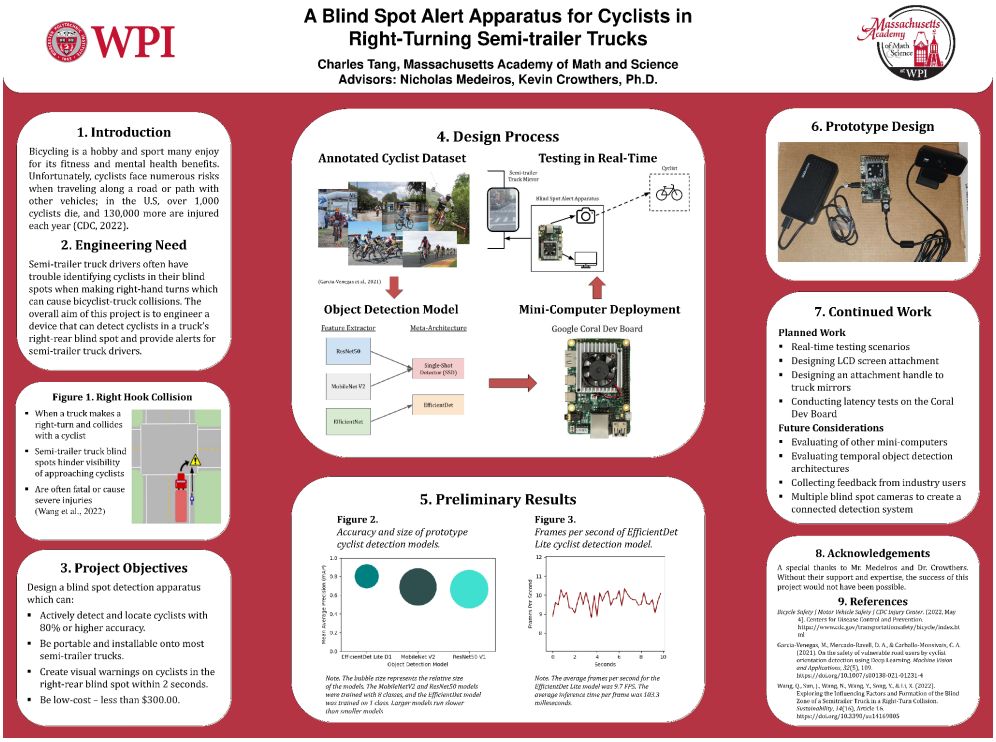
(0-10)	Placement #1	Placement #2	Placement #3
Cover blind zone	7	10	5
Accurately detect cyclists	7	6	6
Ease of installation	9	5	6
TOTAL	23/30	21/30	17/30

This shows that the first and second placements are the better decision choices to implement the camera.

Data Analysis	The design matrix proved that the first and second placements are the strongest options for camera placement. In the case where the camera is placed on top, the cyclist detection model may not be able to detect cyclists from an aerial view well because the training set did not accommodate for this angle. A potential solution to this is to expand the dataset for more angles of the cyclists and more edge case pictures. This placement would also prevent the camera from differentiating whether cyclists are very close to the front (more vulnerable) by using cyclist size. The camera placement near the front of the truck would allow easier wiring and would better work with the CIMAT cyclist dataset benchmark because this dataset uses more images of cyclists facing the camera.
Conclusion	Some considerations for each of the camera placements is the ability to accurately detect cyclists. This may call for a trained model using a different or collected dataset based on aerial views of cyclists.

Daily Entry #32: December Fair Poster

Title	Final poster design
Date	December 7, 2022
Signature	*Charles Tang,*
Poster	Figure 1. Final Poster Design.

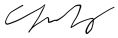


Note. This shows the final December fair design for the poster.

Daily Entry #33: Cyclist Images

Title	Find new cyclist dataset images
Date	December 8, 2022
Signature	*Charles Tang,*
Introduction	To account for more cyclist angles that the object detection model could encounter, I performed a search for other usable cyclist datasets and image collections.
Links	https://cyclinguphill.com/photos-cycling-oxford/ https://storage.googleapis.com/openimages/web/extended.html#crowdsourced https://ieeexplore.ieee.org/abstract/document/8813814 https://universe.roboflow.com/pawel-brzozowski/cyclists_database Each of these links lead to a cyclist dataset with bounding box annotations.
Conclusion	To continue to expand on the CIMAT cyclist dataset, one potential dataset out of the above is the roboflow cyclist dataset because it uses Pascal VOC annotation format. This allows it to be integrated with current dataset images. However, the file names are not usable in the training process and have to be renamed to a number.

Daily Entry #34: New TFRecord

Title	Create tensorflow record files out of new dataset images
Date	December 10, 2022
Signature	*Charles Tang,* 
Introduction	To allow these images to be trained on in the object detection models, I will convert these models to tensorflow record files.
Methods	First, I will need to rename all of the files in the directories into a number format to allow being trained on using Code 1. Then, I will convert these files to a tensorflow record format with this modified script from the Tensorflow Object Detection API. I changed the file name format and the class records for each object using Code 2.
Code 1	`import os` `# Specify the folder containing the files to be renamed` `folder =` `"C:\\Users\\charl\\Documents\\Tensorflow\\workspace\\training_demo\` `\images_collected\\roboflow"` `# Get a list of all files in the folder` `files = os.listdir(folder)` `# Iterate over the files and rename each one based on its file` `extension` `counter = 2` `for file in files:` `    old_name = os.path.join(folder, file)` `    file_name, file_ext = os.path.splitext(file)` `    new_name = os.path.join(folder, "{}{}".format(str(counter//2),` `file_ext))` `    print(old_name, new_name)` `    os.rename(old_name, new_name)` `    counter += 1`
Code 2	`import os` `import glob` `import pandas as pd` `import io` `import xml.etree.ElementTree as ET` `import argparse` `os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow` `logging (1)` `import tensorflow.compat.v1 as tf`

```
from PIL import Image
from object_detection.utils import dataset_util, label_map_util
from collections import namedtuple

# Initiate argument parser
parser = argparse.ArgumentParser(
    description="Sample TensorFlow XML-to-TFRecord converter")
parser.add_argument("-x",
                    "--xml_dir",
                    help="Path to the folder where the input .xml
files are stored.",
                    type=str)
parser.add_argument("-l",
                    "--labels_path",
                    help="Path to the labels (.pbtxt) file.",
type=str)
parser.add_argument("-o",
                    "--output_path",
                    help="Path of output TFRecord (.record) file.",
type=str)
parser.add_argument("-i",
                    "--image_dir",
                    help="Path to the folder where the input image
files are stored. "
                         "Defaults to the same directory as
XML_DIR.",
                    type=str, default=None)
parser.add_argument("-c",
                    "--csv_path",
                    help="Path of output .csv file. If none
provided, then no file will be "
                         "written.",
                    type=str, default=None)

args = parser.parse_args()

if args.image_dir is None:
    args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
    """Iterates through all .xml files (generated by labelImg) in a
given directory and combines
    them in a single Pandas dataframe.
```

```
    Parameters:
    ----------
    path : str
        The path containing the .xml files
    Returns
    -------
    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        filename = xml_file.replace('.xml', '')
        filename =
filename.replace("C:\\Users\\charl\\\\Tensorflow\\workspace\\trainin
g_demo\\images_collected\\roboflow", '')
        width = int(root.find('size').find('width').text)
        height = int(root.find('size').find('height').text)
        for member in root.findall('object'):
            bndbox = member.find('bndbox')
            value = (filename, #filename.replace('.jpg', ''),
                     width,
                     height,
                     member.find('name').text,
                     int(bndbox.find('xmin').text),
                     int(bndbox.find('ymin').text),
                     int(bndbox.find('xmax').text),
                     int(bndbox.find('ymax').text),
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def class_text_to_int(row_label):
    return 1

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]
```

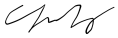
```
def create_tf_example(group, path):
    print(group.filename)
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename) + ".jpg"), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example =
tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
```

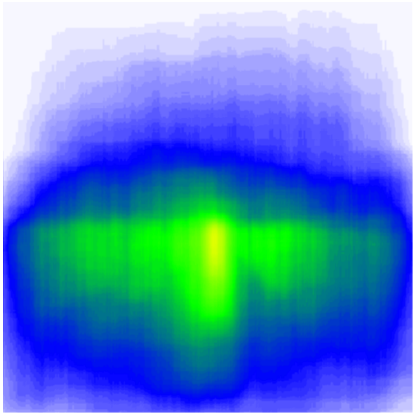
	dataset_util.int64_list_feature(classes), })) return tf_example def main(_): writer = tf.python_io.TFRecordWriter(args.output_path) path = os.path.join(args.image_dir) examples = xml_to_csv(args.xml_dir) grouped = split(examples, 'filename') for group in grouped: tf_example = create_tf_example(group, path) writer.write(tf_example.SerializeToString()) writer.close() print('Successfully created the TFRecord file: {}'.format(args.output_path)) if args.csv_path is not None: examples.to_csv(args.csv_path, index=None) print('Successfully created the CSV file: {}'.format(args.csv_path)) if __name__ == '__main__': tf.app.run()
Results	200 Images collected from the internet + annotations: https://wpi0-my.sharepoint.com/:u:/g/personal/ctang5_wpi_edu/EfVzrv7xwe5HuPhkJpT7-xMBdxyJ38L11aVsOJ48HYr7xQ?e=YwCetD Roboflow 1000 images + annotations: https://wpi0-my.sharepoint.com/:u:/g/personal/ctang5_wpi_edu/ET2Yl_3P5_pApU1krxIDHYcBijVX_dgMtHCYRSu9bkV8aA?e=nyhSL7
Conclusion	Further model training will be performed using these combined datasets and the results will be compared to that of the CIMAT cyclist dataset.

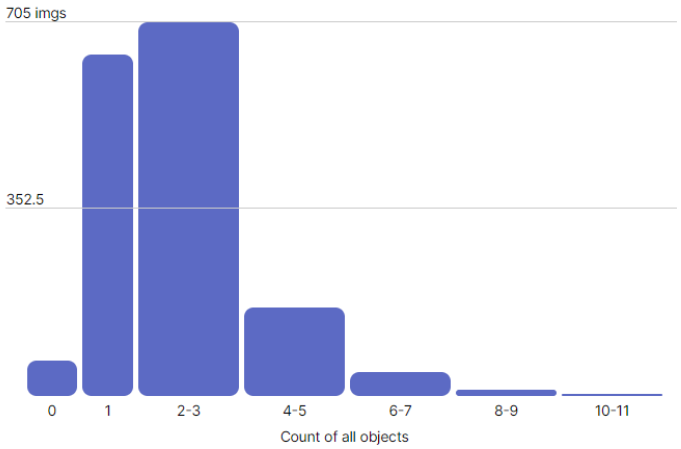
Daily Entry #35: Training Procedure

Title	Training Procedure
Date	December 12, 2022
Signature	*Charles Tang,* 
Introduction	In order to standardize the training procedure for training single-class cyclist detection models, today I will develop a training procedure with outlined hyperparameters and dataset usage. This is also to diversify the training data and allow the model to understand different viewing angles of cyclists.
Methods	For any model to be trained: 1. Train on the CIMAT cyclist dataset of 12,000 images for 20-30 epochs to achieve at least a loss of 0.3 and mAP of 75%. Using learning rate starting at 0.08 with less decay than the preset values. Save the final checkpoint file. Evaluate this model and record the mAP. 2. Load the checkpoint file and train on the Roboflow + synthetic images dataset for 10 epochs with learning_rate = 0.05 with minimal decay. Save this new checkpoint file. Evaluate this model and record the mAP. 3. Evaluate this final model using the original CIMAT cyclist validation set and record results. 4. Evaluate this final model using the dataset and record results. 5. Evaluate this final model using the Personal dataset and record results.
Conclusion	The next step in this project is to train a model using this design process and to modify it as needed to improve results.

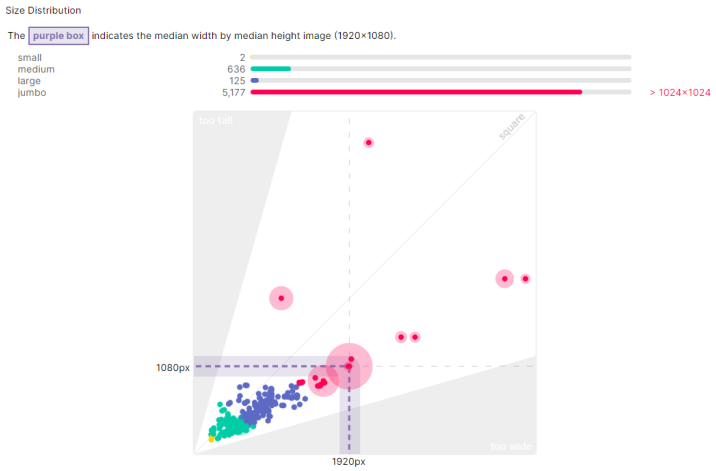
Daily Entry #36: Annotate Images.

Title	Annotate and collect new synthetic images of cyclists
Date	December 14, 2022
Signature	*Charles Tang,* 
Introduction	To better account for different angles and more edge cases of cyclist detection, I plan to use a synthetic cyclist detection dataset offered by Parallel Domain.
Methods	To use a smaller subset of the images provided, I manually parsed through scenes 0-60 of each image PD-K folder and a random selection of a fourth of the dataset. For the case of large wide images, I used the following script to cut the image up into three parts. After this was all completed, I ended up with a set of 1000+ images with cyclists.
Code 1	`import os` `from PIL import Image` `# Set the directory where the images are stored` `image_dir = 'C:\\Users\\charl\\OneDrive - Worcester Polytechnic Institute (wpi.edu)\\pd-open-datasets-bicycle-detection\\images-PD-K-scenes-0-60'` `# Loop through all the files in the directory` `for file in os.listdir(image_dir):` `  # Open the image file` `  img = Image.open(os.path.join(image_dir, file))` `  # Get the width and height of the image` `  width, height = img.size` `  # Calculate the x coordinate for the left and right halves` `  x1 = 0` `  x2 = width // 4` `  x3 = width // 2` `  x4 = 3 * width // 4` `  # Crop the left and right halves of the image` `  left = img.crop((x1, 0, x3, height))` `  middle = img.crop((x2, 0, x4, height))` `  right = img.crop((x3, 0, width, height))` `  # Save the left right and middle halves to separate files` `  left.save(os.path.join(image_dir, 'left_' + file))`

	`middle.save(os.path.join(image_dir, 'middle_' + file))` `right.save(os.path.join(image_dir, 'right_' + file))`
Results	Link to dataset: https://wpi0-my.sharepoint.com/:f:/g/personal/ctang5_wpi_edu/Etv5a42JzIBMioA2kvsN8XkBlhxQiIS89AWDXjaUhhpJzA?e=8WdvLH Figure 1.. Average Image Size of New Dataset.  *Note. This figure shows the average size of images and the size distributions. Currently, my image lean on the large side.* Figure 2. Image Annotation Heatmap.  *Note. This figure shows the average location of cyclists in the annotated bounding boxes.* Figure 3. Number of Cyclists Per Image.

	 *Note. This figure is a histogram of the object count in each image. Note that the average number of cyclists in an image is about 2.*
Data Analysis	In Figure 18, the average image size is very high, which may not perform well on small 300x300 input size models such as the EfficientDet or MobileNetV2. Thus, it may be necessary to combine the CIMAT cyclist and the Roboflow dataset in order to prevent training only on difficult cyclist instances. In Figure 69, the image annotations show good variability in where the cyclists are located in the images. Furthermore, in Figure 3, there is on average more than one cyclist per image (disregarding null), which would be adequate for training strong performing models.
Conclusion	I initially annotated 500 images and need to continue annotating the rest. Furthermore, I need to parse through the rest of the images in the dataset to choose which would be suitable for training the deep learning model on. Furthermore, I reached out to the Roboflow team for a research license.

Daily Entry #37: Continue Annotating

Title	Continue to annotate and collect new synthetic images of cyclists
Date	December 15, 2022
Signature	*Charles Tang,*
Introduction	In this daily entry, I continued adding onto the synthetic images dataset and merged it with the other roboflow cyclist detection dataset to be trained alongside the CIMAT cyclist dataset. This means that the supplemental dataset will have over 5,000 images.
Methods	1. Add the previous Roboflow cyclist dataset to this dataset. https://wpi0-my.sharepoint.com/:f:/g/personal/ctang5_wpi_edu/EtkVIDrAkcFLtI8KfEQXDKwBvojwu5jqrN-FkrFG0kbazQ?e=bx6UuA 2. Split it 80% training, 20% testing. 3. Annotate the unannotated images and check all images for correct annotations.
Results	Figure 1. Size distribution of images.  *Note. This image shows the distribution of the sizes of images in the dataset.* Figure 2. Annotation heatmap.

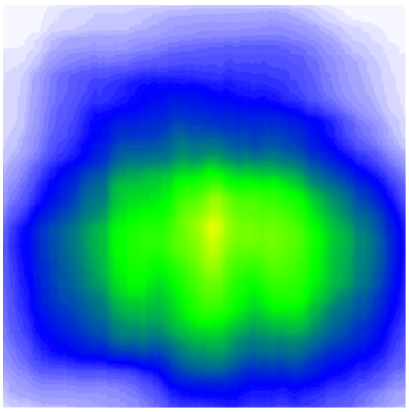
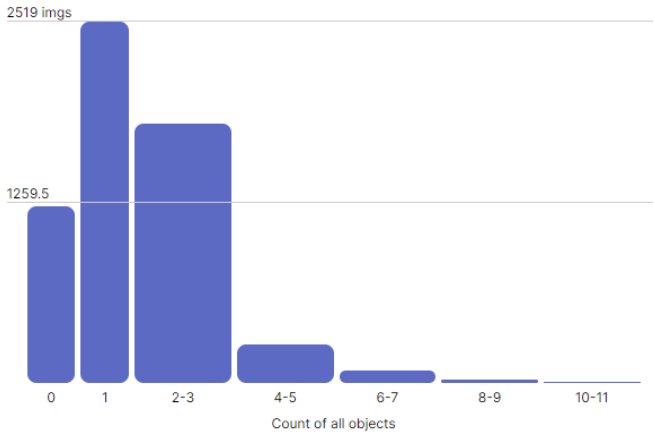
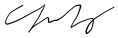
<table>
<tr><td></td><td> 
Note. This shows the distribution of cyclists within the images.

Figure 3. Histogram of cyclist instances.
 

Note. This shows the number of images with each number of objects.</td></tr>
<tr><td>Data Analysis</td><td>The annotation heatmap shows that mostly the images of the cyclists are located near the center, which would represent real-time scenarios. Furthermore, the number of cyclist instances in each image is good as most images have a non-zero number of cyclists. However, this would not impact the training accuracy much because of the cropping and region proposal algorithms within the object detection models. Furthermore, this dataset has a good number of images with 2 or more cyclists, which will support the training process. In Figure 1, since the image sizes seem to be much larger than the input of the object detection models, the annotated cyclists (especially the smaller ones) may lose features as a result.</td></tr>
<tr><td>Conclusion</td><td>The next step in this project is to continue adding synthetic images to this dataset and continue annotating them.</td></tr>
</table>

Daily Entry #38: New TFRecord

Title	Export as TFrecord and train on new dataset
Date	December 16, 2022
Signature	*Charles Tang,* 
Introduction	In this daily entry, I compiled all of the images and annotations from the Roboflow dataset and fine-tuned a previously trained cyclist detection model with this new data.
Methods	1. Convert the Roboflow + Synthetic Dataset into a tfrecord file. 2. Modify previously existing code to train the new model with the previous checkpoint weights from the 80% accuracy model. 3. Train the new model for 15 epochs on the new dataset. 4. Evaluate this new model on the CIMAT cyclist test set and the new dataset's test set. 5. Record and analyze results.
Results	Train TFRecord File: https://drive.google.com/file/d/14D4LR3pHhiw3jmbrW_revEAAYC32g2CI/view?usp=sharing Test TFRecord File: https://drive.google.com/file/d/1siZBSLadvNMLcQV3puxjNCkZxmPmgrhS/view?usp=sharing Latest Checkpoint Weights File (EfficientDet Lite 1): https://drive.google.com/file/d/115eXfkecTQ5UcapVbABPKxgmUUaMlAdI/view?usp=sharing https://drive.google.com/file/d/11BFTRcFqa49F3ySv4rwyuQ0K8eUaF185/view?usp=sharing Exported Model File: https://drive.google.com/drive/folders/1V_hZkYG_gt3KyQfqMk6m5ZptwpMr9grQ?usp=sharing Figure 1. Loss of the trained model.

```
Epoch 45: saving model to /content/drive/My
Drive/STEM_1/dataset/try6/last_layers/ckpt-45
445/445 [==============================] - 422s 949ms/step - det_loss:
0.2881 - cls_loss: 0.1743 - box_loss: 0.0023 - reg_l2_loss: 0.0981 - loss:
0.3862 - learning_rate: 0.0023 - gradient_norm: 1.0598
<keras.callbacks.History at 0x7fc237d4f790>
```

Note. This is the loss calculations of the newly trained object detection model on the new synthetic + roboflow dataset.

Link to loss results:
https://docs.google.com/document/d/1rCO60kGjIPjh_B-lVkdO1BHbXmuXBB5FyKLQarTTu9o/edit?usp=sharing

Figure 2. Accuracy metrics of the trained model on Roboflow test set.

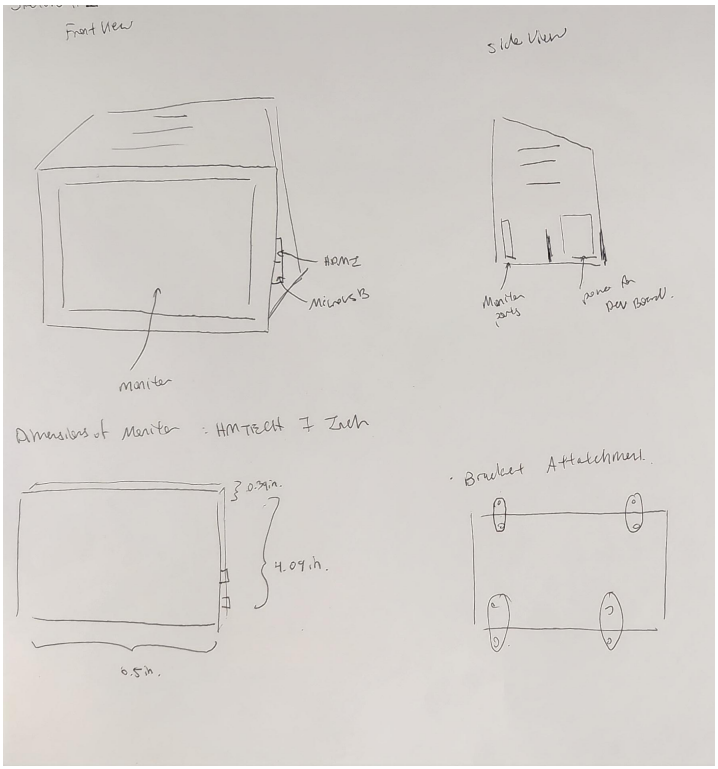
```
'AP': 0.57744664,
'AP50': 0.9282869,
'AP75': 0.62850314,
'APs': 0.1993011,
'APm': 0.5327555,
'APl': 0.64206433,
'ARmax1': 0.35225278,
'ARmax10': 0.64218843,
'ARmax100': 0.6475717,
'ARs': 0.33831775,
'ARm': 0.61055046,
'ARl': 0.7080169,
'AP_/Cyclist': 0.57744664}
```

Note. This is the mAP of the model on the test set of the Roboflow + Synthetic Data dataset.

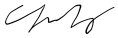
Figure 3. Accuracy metrics of the trained model on the CIMAT test benchmark.

	{'AP': 0.6652071, 'AP50': 0.95401025, 'AP75': 0.811753, 'APs': 0.0, 'APm': 0.33794585, 'APl': 0.68926585, 'ARmax1': 0.41520637, 'ARmax10': 0.7160753, 'ARmax100': 0.72027516, 'ARs': 0.0, 'ARm': 0.47247708, 'ARl': 0.74209285, 'AP_/Cyclist': 0.6652071} *Note. This shows the mAP and AR metrics of the trained model on the test set of the CIMAT cyclist orientations dataset.*
Data Analysis	Although the mAP decreased, it seemed to have generalized decently well to both datasets. Both mAP metrics range around 60%, with AP at IoU level of 50% being around 90%, meaning that further training may improve accuracy significantly. The high AP values at IoU of 0.5 mean that the detections are usually made but the bounding box is not exactly the right place. This data could be unrepresentative of its actual accuracy because the smaller instances from the synthetic image dataset may be hard to detect exactly when the images are shrunken to the input size of the object detection model.
Conclusion	Future work would be to expand the roboflow dataset with more synthetic images and to continue training with the new datasets to improve accuracy. One potential option is to continue training for 30-40 more epochs. Furthermore, it would be necessary to use CAD to bring together all of the components and to test it in real-time scenarios.

Daily Entry #39: CAD Sketch

Title	Develop a prototype sketch for a CAD design for the final apparatus
Date	December 17, 2022
Signature	*Charles Tang,*
Introduction	The purpose of this daily entry is to prototype a sketch CAD design to be able to encase a screen and the Google Coral Dev Board.
Proposed Sketches	Figure 1. CAD Design.  *Note. This shows the front and side views of the proposed cad design.* Prototype Model: https://www.tinkercad.com/things/lecNcjAdXeC-case/edit?sharecode=Y8bTdvZ7BaHjHaUDLYJdeZ6ob165zNOlAIfW3z4hZlU
Conclusion	The next step is to fine tune and check that this prototype will match the dimensions of the screen.

Daily Entry #40: Train on New Dataset

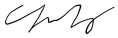
Title	Continue training previous model on new synthetic dataset
Date	December 18, 2022
Signature	*Charles Tang,* 
Introduction	In this daily entry, I continued training from the previous checkpoint and trained on the new dataset.
Methods	1. Convert the Roboflow + Synthetic Dataset into a tfrecord file. 2. Modify previously existing code to train the new model with the previous checkpoint weights from Daily Entry #36. 3. Train the new model for 15 epochs on the new dataset. 4. Evaluate this new model on the CIMAT cyclist test set and the new dataset's test set. 5. Record and analyze results.
Results	Latest Checkpoint Weights File (EfficientDet Lite 1): https://drive.google.com/file/d/11UX8KyAmm-pYGyNoQRGAX0WY8j3lBTZN/view?usp=sharing https://drive.google.com/file/d/11kJOh7cUXUE-kTsv9UukMitZ8iTwbWyn/view?usp=sharing Exported Model File: https://drive.google.com/file/d/11n1mIXJNsDJk_kMzzA3dqTeQwRyL_wjK/view?usp=sharing Figure 1. Accuracy metrics of the trained model on Roboflow test set. `{'AP': 0.5311011,` `'AP50': 0.90847266,` `'AP75': 0.5617804,` `'APs': 0.15283729,` `'APm': 0.48018998,` `'APl': 0.60342044,` `'ARmax1': 0.32791105,` `'ARmax10': 0.5983616,` `'ARmax100': 0.6039204,` `'ARs': 0.27850467,` `'ARm': 0.55489296,` `'ARl': 0.6744726,` `'AP_/Cyclist': 0.5311011}`

	Note. This is the mAP of the model on the test set of the Roboflow + Synthetic Data dataset. Figure 2. Accuracy metrics of the trained model on the CIMAT test benchmark. {'AP': 0.6013433, 'AP50': 0.9394732, 'AP75': 0.7217308, 'APs': 0.0, 'APm': 0.2412228, 'APl': 0.62867576, 'ARmax1': 0.38146272, 'ARmax10': 0.6608255, 'ARmax100': 0.6659667, 'ARs': 0.0, 'ARm': 0.35688072, 'ARl': 0.69299763, 'AP_/Cyclist': 0.6013433} *Note. This shows the mAP and AR metrics of the trained model on the test set of the CIMAT cyclist orientations dataset.*
Data Analysis	The new accuracy results show that the model at the 60 epoch benchmark was most likely overfitted to the training set and the 45 epoch checkpoint was the best generalized accuracy. The low mAP score over IoU from 50 to 95 may be because of the edge case scenarios in the dataset. Further testing is required to determine if the model actually performs well in edge case scenarios and if it improved at all compared to the CIMAT cyclist dataset on other bicyclist instances.
Conclusion	The next step of this model is to further train it on either the CIMAT cyclist dataset or the Roboflow dataset until the 100 epoch mark. Furthermore, I could also deploy this newly trained model onto the Google Coral Dev Board for real-time testing.

Daily Entry #41: Deploy New Model

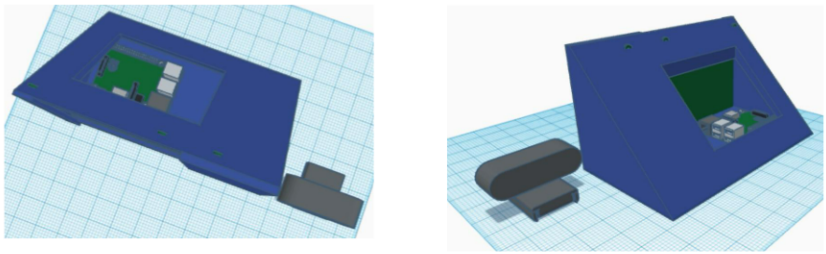
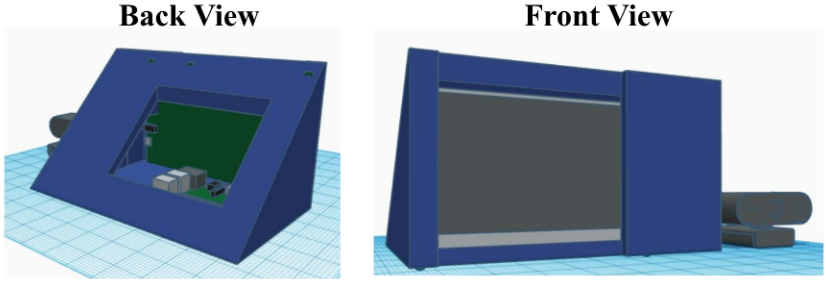
Title	Deployment of Single-Class Cyclist Detection EfficientDet Lite 1 object detection models onto the Google Coral Dev Board
Date	December 20, 2022
Signature	*Charles Tang,* 
Introduction	In this daily entry, I deployed the newly trained EfficientDe Lite 1 onto the Google Coral Dev Board, which was trained on the newly proposed dataset as well as the CIMAT cyclist dataset.
Methods	1. Convert Tflite graph to Edge TPU Quantized Graph 2. Upload EdgeTPU graph to Github. 3. SSH into the Dev Board and wget the model. 4. Test to see if the model ran.
Conclusion	The model ran successfully on the Google Coral Dev Board. Furthermore, qualitative observations showed that the FPS data matches that of the previous EfficientDet Lite 1 model. Future work would be to create figures for the final research paper deliverable as well as train a SSD MobileNetV2 for the cyclist detection task.

Daily Entry #42: Docker Environment

Title	Create training environment for SSD MobileNet V2
Date	December 22, 2022
Signature	*Charles Tang,* 
Introduction	After meeting with Grant Perkins (Mass Academy Alumni + WPI Senior) in discussing the training SSD MobileNetV2 architectures, I set up a docker container for training with TF1.
Methods	To set up the docker container (Retrain an object detection model | Coral) 1. Create a folder /google-coral 2. cd google-coral 3. git clone https://github.com/google-coral/tutorials.git 4. cd tutorials/docker/object_detection 5. docker build . -t detect-tutorial-tf1 6. docker run --name edgetpu-detect --rm -it --privileged -p 6006:6006 --mount type=bind,src=C:\Users\charl\Documents\google-coral\tutorials\docker\object_detection,dst=/tensorflow/models/research/learn_pet detect-tutorial-tf1 When inside the docker container: 7. ./prepare_checkpoint_and_dataset.sh --network_type mobilenet_v2_ssd --train_whole_model false 8. Delete the Oxford pets training records and upload the single-class cyclist detection tfrecord into the learn_pet directory 9. Configure the pipeline file to single-class object detection with training for 50000 steps 10. NUM_TRAINING_STEPS=50000 && NUM_EVAL_STEPS=2000 11. ./retrain_detection_model.sh --num_training_steps ${NUM_TRAINING_STEPS} --num_eval_steps ${NUM_EVAL_STEPS} When training is finished: 12. ./convert_checkpoint_to_edgetpu_tflite.sh --checkpoint_num 50000 13. Copy tflite file to local computer outside of the docker container 14. Record mAP values from the auto evaluation call on the last 1000 steps.
Results	The results from this training process were invalid due to an error possibly in the dataset or label map values. It caused the mAP values to all take on -1, and the loss values were exploding between 0 and 50. Furthermore, this training process is deprecated since TF1 is not supported.

Conclusion	The next step in this project is to set up another training environment for the SSD MobileNet V2 using potentially the Tensorflow 2 object detection API as well as debug the issues found in the single-class cyclist detection dataset and rebuilding the tfrecord files.

Daily Entry #43: CAD Design

Title	Develop CAD design for final device
Date	December 23, 2022
Signature	*Charles Tang,*
Introduction	The purpose of this daily entry is to develop a CAD model for the final blind spot device with wiring and spaces for the monitor and the Coral Dev Board.
Methods	1. Use OnShape CAD. 2. Create a large rectangular box to contain all of the components. 3. Create cutouts for the monitor and the cable outputs. 4. Verify that dimensions fit the apparatus. 5. Print it using a 3-D printer.
Results	Figure 1. 4 perspectives of blind spot CAD apparatus. **Top View** **Side View**  **Back View** **Front View** 
Conclusion	The next step of the design process is to 3-D print the design. Furthermore, more object detection models can be trained and should be analyzed on the Coral Dev Board as well as in real-time environments.

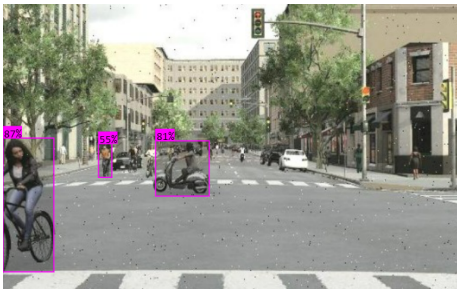
Daily Entry #44: Train MobileNet TF2

Title	Train MobileNetV2 using TF2 Object Detection API
Date	December 26, 2022
Signature	*Charles Tang,*
Introduction	The main objective of this daily entry is to train a fully functioning single-class cyclist detection model using the Tensorflow 2 Object Detection API.
Methods	1. Download MobileNetV2 COCO checkpoint files and architecture. 2. Update the pipeline and the label map files. 3. Train for 40,000 steps on the CIMAT cyclist dataset and 10,000 steps on the new proposed cyclist dataset.. 4. Export as a tflite file and analyze mAP results.
Results	Table 1. mAP of the Cyclist Detection Model after 50,000 Epochs. CIMAT Val Results mAP (50 IoU) 0.926 mAP (75 IoU) 0.764 mAP (50:05:95) 0.656 Roboflow Val Results mAP (50 IoU) 0.616 mAP (75 IoU) 0.269 mAP (50:05:95) 0.309 *Note. These accuracy results use mAP metrics proposed by the MS COCO dataset.*
Data Analysis	The accuracy from this model is slightly worse than that of the EfficientDet Lite 1 and 2 models. There is a 3% decrease in accuracy from the EfficientDet 1 and 2 in the mAP at 0.5 IoU threshold. Furthermore, the decrease in accuracy in the other categories, which describe the accuracies of most accurate cyclist predictions, show that this model does not perform up to the engineering objectives of this paper. This could be due to the small architecture of this model and its inability to generalize to complex features in urban environments. When trained on the newly proposed Roboflow dataset, the accuracy on the CIMAT dataset dropped which showed that the hard features of the synthetic images hurt performance on easier detections.
Conclusion	Future work of this study would be to take a look into validation results on real images and determine which model performs the best from a qualitative standpoint. Furthermore, it would be advisable to manually create the validation set in order to balance various scenarios, such as rain, dark conditions, and easier cyclist detentions. Lastly, future work would include testing in real-time and testing the right-hook turn environment.

Daily Entry #45: Test Models

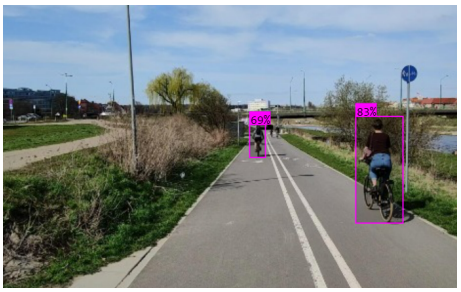
Title	Test models on validation set and record example predictions
Date	December 28, 2022
Signature	*Charles Tang,*
Introduction	The purpose of this daily entry is to perform some initial testing with select pictures of the validation set and analyze the models for effectiveness.
Methods	Run the tflite model on the image file using the example script (Code 1).
Code 1	

```
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
import cv2
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import tflite_runtime.interpreter as tflite
from pycoral.adapters import common
from pycoral.adapters import detect
from pycoral.utils.dataset import read_label_file
from IPython.display import clear_output


def draw_objects(draw, objs, scale_factor, labels):
  """Draws the bounding box and label for each object."""
  COLORS = np.random.randint(0, 255, size=(len(labels), 3),
dtype=np.uint8)
  for obj in objs:
    bbox = obj.bbox
    color = tuple(int(c) for c in COLORS[obj.id])
    draw.rectangle([(bbox.xmin * scale_factor, bbox.ymin *
scale_factor),
                    (bbox.xmax * scale_factor, bbox.ymax *
scale_factor)],
                   outline=color, width=3)
    # font = ImageFont.truetype("LiberationSans-Regular.ttf",
size=15)
    draw.text((bbox.xmin * scale_factor + 4, bbox.ymin *
scale_factor + 4),
              '%s\n%.2f' % (labels.get(obj.id, obj.id), obj.score),
              fill=color)

# Load the TF Lite model
labels = read_label_file('bikes-labels.txt')
```

	`` interpreter = tflite.Interpreter('efficient-det-bikes.tflite') interpreter.allocate_tensors() # Resize the image for input def resize_img(image): image = image.resize((display_width, int(display_width * height_ratio))) return image image = Image.open('bikebirdeye.jpg') _, scale = common.set_resized_input(interpreter, image.size, lambda size: image.resize(size, Image.ANTIALIAS)) image = resize_img(image) #display sizes display_width = 500 scale_factor = display_width / image.width height_ratio = image.height / image.width # Run inference interpreter.invoke() objs = detect.get_objects(interpreter, score_threshold=0.2, image_scale=scale) draw_objects(ImageDraw.Draw(image), objs, scale_factor, labels) image ```
Results	Figure 1. Example prediction #1.  *Note. This image is from the Open Synthetic Dataset offered by Parallel Domain.* Figure 2. Example prediction #2.

Note. This image is of a smaller cyclist instance from an aerial view.

Figure 3. Example prediction #3.

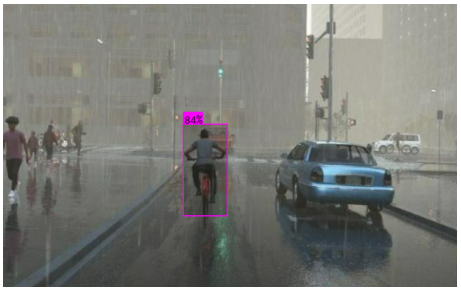


Note. This image is of a rear view of cyclists.

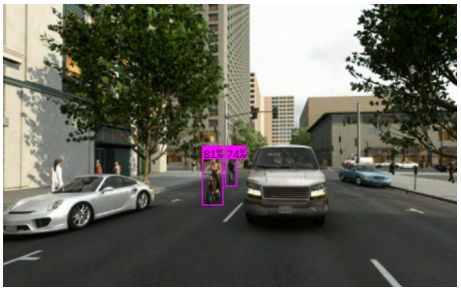
Figure 4. Example prediction #4.

Note. This image is from the Open Synthetic Dataset for Cyclists.

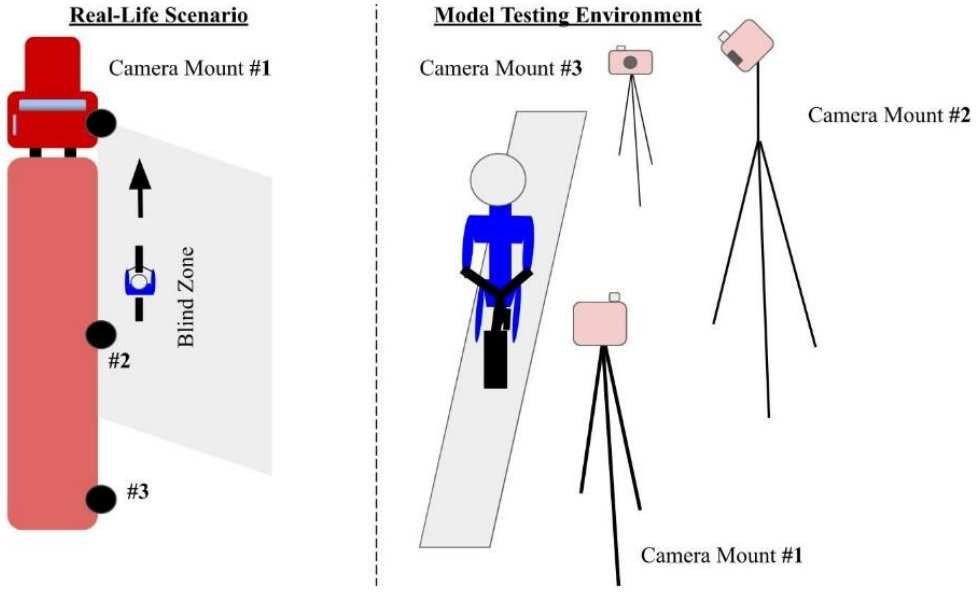
Figure 5. Example prediction #5.

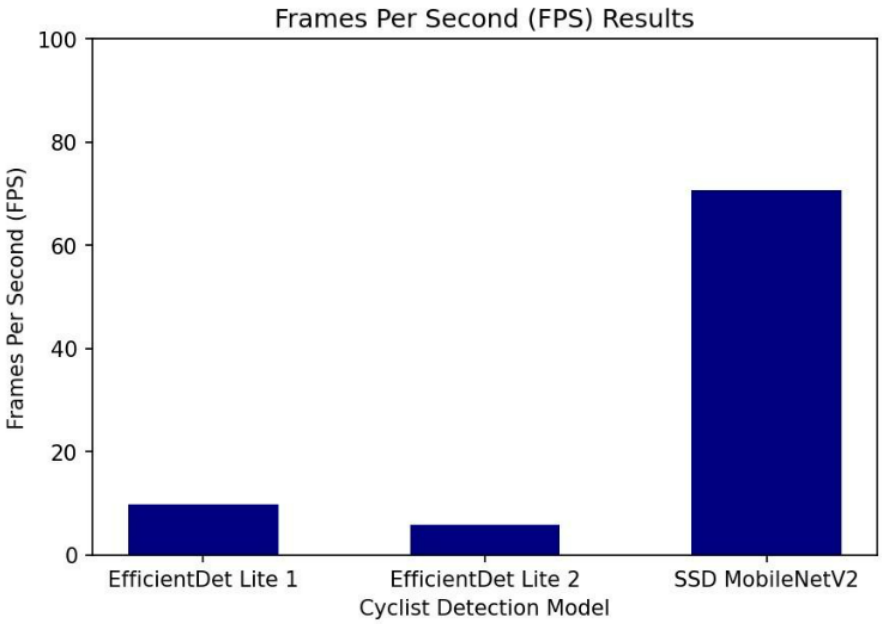


Note. This image is from the Open Synthetic Dataset for Cyclists.

	Figure 6. Example prediction #6.  *Note. This image is from the Open Synthetic Dataset for Cyclists.*
Data Analysis	The example predictions shown above show that the EfficientDet Lite model works well in detecting cyclists from a variety of environments. In Figure 1, the model is able to detect three cyclists: one partially visible on the left, one in a moped bicycle, and one waiting at the intersection. All of these cyclist instances are relatively difficult to detect as they are small and in various positions. In Figure 2, the small cyclist in the background was detected with 80% confidence, meaning that the model performs well on smaller cyclist instances. In Figures 3 and 6, the same is held true as the model successfully detects faraway cyclists with high accuracy. In Figure 5, the rainy environment and successful detections illustrate the good performance of the model even with variable lighting and weather conditions.
Conclusion	The next step in the design process is to test this device in real-time with model traffic environments as well as conduct further speed tests with the new EfficientDet Lite 2 and MobileNetV2 models.

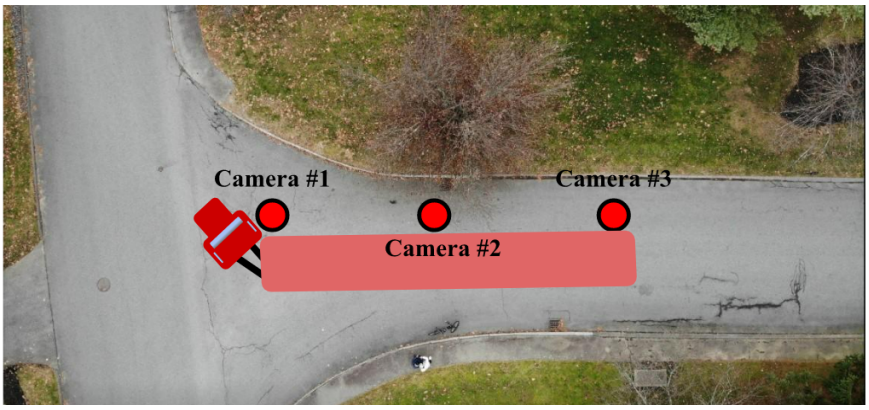
Daily Entry #46: Test in Real-Time

Title	Test models in real-time right-hook turn model traffic scenario
Date	December 30, 2022
Signature	*Charles Tang,*
Introduction	The purpose of this daily entry is to test the cyclist detection models in a real traffic scenario with a model semi-trailer truck and a through-going cyclist on the right-hand side.
Methods	A real-time testing environment was then constructed, which included a cyclist, a fixed mount for the blind spot device, and the blind spot device. The cyclist was guided down a straight path to simulate a through-going cyclist, and the cyclist detection apparatus was then fixed at three locations to simulate a semi-trailer truck at an intersection (Figure 1): in front of the cyclist, above the cyclist, and behind the cyclist. The first camera was placed at a height of 5 feet, the second camera was placed at a height of 13 feet, and the third camera was placed at a height of 13 feet. The device was tested qualitatively for its effectiveness in detecting cyclists from the viewpoint of a truck cab as well as for inference speed. Figure 1. Model testing environment.  *Note. This model will test three camera placements: in front of the cyclist, above the cyclist, and behind the cyclist.*
Results	Figure 2. Frames per second results for each object detection model.



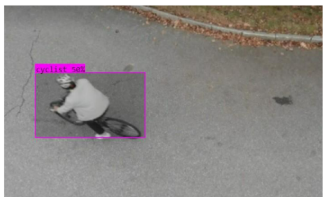
Note. The frames per second were obtained by calculating the average inference time per frame.

Figure 3. Aerial view of real-time testing environment.



Note. The camera placements match that of Figure 1.

Figure 4. Example detections from cyclist detection models.

	   *Note. The percentages shown on the bounding boxes are the confidence scores assigned by the EfficientDet Lite 1 cyclist detection model. These predictions are made by the EfficientDet Lite 1 and 2 model. (Left) Camera placement #1 shows an oncoming cyclist. (Middle) Camera placement #2 shows an aerial view of a cyclist. (Right) Camera placement #3 shows a rear view of the cyclist.*
Data Analysis	This testing method is unique and uses real-time tests rather than computer simulations used in various other literature. In Figure 2, the highest frames per second was produced by the MobileNetV2 at 75 FPS, which means it can make inferences at nearly 1/100 of a second. However, this model is compromised at accuracy, which is crucial in safety-critical devices. The EfficientDet Lite 1 and 2 both perform well under these conditions, with FPS of 10 and 7, respectively. Figure 4 displays some of the predictions from these tests. Based on qualitative analysis of the camera placements in Figure 3, camera #2 had the worst performance in terms of accuracy, potentially due to the lack of images in the training dataset that resemble this angle of cyclists. When comparing cameras #1 and #3, they had similar performance, however camera #1 would be more easily installable onto a semi-trailer truck, which makes it the optimal camera placement on the semi-trailer truck.
Conclusion	The next step in this study is to further expand the training set and to merge the CIMAT and newly proposed dataset for better training results.

Daily Entry #47: Object Tracking Design

<table>
<tr><td>Title</td><td>Object Tracking Options</td></tr>
<tr><td>Date</td><td>January 5, 2023</td></tr>
<tr><td>Signature</td><td>Charles Tang,</td></tr>
<tr><td>Introduction</td><td>The purpose of this daily entry is to create a design matrix to evaluate the different object tracking algorithms.</td></tr>
<tr><td>Design Matrix</td><td>Table 1. Design matrix for object tracker algorithms.
<table>
<tr><th>Object Tracker</th><th>Code Availability</th><th>Speed</th><th>Accuracy</th></tr>
<tr><td>Simple Online and Realtime Tracking (SORT)</td><td>1</td><td>10</td><td>6</td></tr>
<tr><td>DeepSORT</td><td>1</td><td>5</td><td>8</td></tr>
<tr><td>ByteTrack</td><td>0</td><td>8</td><td>10</td></tr>
</table>
Since the SORT algorithm provides the simplest and least computationally heavy method for real-time tracking, it is the optimal tracking algorithm based on this design matrix.</td></tr>
<tr><td>Conclusion</td><td>For this application, the design matrix showed that the SORT algorithm would prove best for the cyclist detection task.</td></tr>
</table>

Daily Entry #48: Implement SORT Algorithm

Title	Implement SORT algorithm on Coral Dev Board
Date	January 10, 2023
Signature	*Charles Tang,*
Introduction	The purpose of this daily entry is to implement the tracking algorithm onto the Google Coral Dev Board and run a test run using it.
Methods	1. Clone github repository onto the Google Coral Dev Board git clone https://github.com/google-coral/example-object-tracker.git 2. CD into the gstreamer folder. cd example-object-tracker cd gstreamer 3. Modify the detect.py (Code 1) file to be compatible with the EfficientDet Lite model and change tensor outputs. 4. Run detect.py with the appropriate inputs with the EfficientDet Lite 1 model. python3 detect.py --model ../../efficientdet1.tflite --labels ../../demo_files/bike-labels-tracker.txt --videosrc /dev/video1 --top_k 1 --tracker sort --threshold 0.5
Code 1	/example-object-tracker/gstreamer/detect.py `import argparse` `import collections` `import common` `import gstreamer` `import numpy as np` `import os` `import re` `import svgwrite` `import time` `from tracker import ObjectTracker` `Object = collections.namedtuple('Object', ['id', 'score', 'bbox'])` `def load_labels(path):` `  p = re.compile(r'\s*(\d+)(.+)')` `  with open(path, 'r', encoding='utf-8') as f:` `  lines = (p.match(line).groups() for line in f.readlines())`

```
        return {int(num): text.strip() for num, text in lines}


def shadow_text(dwg, x, y, text, font_size=20):
    dwg.add(dwg.text(text, insert=(x+1, y+1), fill='black',
font_size=font_size))
    dwg.add(dwg.text(text, insert=(x, y), fill='white',
font_size=font_size))


def generate_svg(src_size, inference_size, inference_box, objs,
labels, text_lines, trdata, trackerFlag):
    dwg = svgwrite.Drawing('', size=src_size)
    src_w, src_h = src_size
    inf_w, inf_h = inference_size
    box_x, box_y, box_w, box_h = inference_box
    scale_x, scale_y = src_w / box_w, src_h / box_h

    for y, line in enumerate(text_lines, start=1):
        shadow_text(dwg, 10, y*20, line)
    if trackerFlag and (np.array(trdata)).size:
        for td in trdata:
            x0, y0, x1, y1, trackID = td[0].item(), td[1].item(
            ), td[2].item(), td[3].item(), td[4].item()
            overlap = 0
            for ob in objs:
                dx0, dy0, dx1, dy1 = ob.bbox.xmin.item(),
ob.bbox.ymin.item(
                ), ob.bbox.xmax.item(), ob.bbox.ymax.item()
                area = (min(dx1, x1)-max(dx0, x0))*(min(dy1,
y1)-max(dy0, y0))
                if (area > overlap):
                    overlap = area
                    obj = ob

            # Relative coordinates.
            x, y, w, h = x0, y0, x1 - x0, y1 - y0
            # Absolute coordinates, input tensor space.
            x, y, w, h = int(x * inf_w), int(y *
                                             inf_h), int(w *
inf_w), int(h * inf_h)
            # Subtract boxing offset.
            x, y = x - box_x, y - box_y
            # Scale to source coordinate space.
            x, y, w, h = x * scale_x, y * scale_y, w * scale_x, h *
scale_y
            percent = int(100 * obj.score)
```

```
            label = '{}% {} ID:{}'.format(
                percent, labels.get(obj.id, obj.id), int(trackID))
            shadow_text(dwg, x, y - 5, label)
            dwg.add(dwg.rect(insert=(x, y), size=(w, h),
                             fill='none', stroke='red',
stroke_width='2'))
    else:
        for obj in objs:
            x0, y0, x1, y1 = list(obj.bbox)
            # Relative coordinates.
            x, y, w, h = x0, y0, x1 - x0, y1 - y0
            # Absolute coordinates, input tensor space.
            x, y, w, h = int(x * inf_w), int(y *
                                             inf_h), int(w *
inf_w), int(h * inf_h)
            # Subtract boxing offset.
            x, y = x - box_x, y - box_y
            # Scale to source coordinate space.
            x, y, w, h = x * scale_x, y * scale_y, w * scale_x, h *
scale_y
            percent = int(100 * obj.score)
            label = '{}% {}'.format(percent, labels.get(obj.id,
obj.id))
            shadow_text(dwg, x, y - 5, label)
            dwg.add(dwg.rect(insert=(x, y), size=(w, h),
                             fill='none', stroke='red',
stroke_width='2'))
    return dwg.tostring()


class BBox(collections.namedtuple('BBox', ['xmin', 'ymin', 'xmax',
'ymax'])):
    """Bounding box.
    Represents a rectangle which sides are either vertical or
horizontal, parallel
    to the x or y axis.
    """
    __slots__ = ()


def get_output(interpreter, score_threshold, top_k,
image_scale=1.0):
    """Returns list of detected objects."""
    boxes = common.output_tensor(interpreter, 1)
    category_ids = np.array([1])
    scores = common.output_tensor(interpreter, 2)
```

```
    def make(i):
        ymin, xmin, ymax, xmax = boxes[i]
        return Object(
            id=int(category_ids[i]),
            score=scores[i],
            bbox=BBox(xmin=np.maximum(0.0, xmin),
                      ymin=np.maximum(0.0, ymin),
                      xmax=np.minimum(1.0, xmax),
                      ymax=np.minimum(1.0, ymax)))
    return [make(i) for i in range(top_k) if scores[i] >=
score_threshold]


def main():
    default_model_dir = '../models'
    default_model =
'mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite'
    default_labels = 'coco_labels.txt'
    parser = argparse.ArgumentParser()
    parser.add_argument('--model', help='.tflite model path',
                        default=os.path.join(default_model_dir,
default_model))
    parser.add_argument('--labels', help='label file path',
                        default=os.path.join(default_model_dir,
default_labels))
    parser.add_argument('--top_k', type=int, default=3,
                        help='number of categories with highest
score to display')
    parser.add_argument('--threshold', type=float, default=0.1,
                        help='classifier score threshold')
    parser.add_argument('--videosrc', help='Which video source to
use. ',
                        default='/dev/video0')
    parser.add_argument('--videofmt', help='Input video format.',
                        default='raw',
                        choices=['raw', 'h264', 'jpeg'])
    parser.add_argument('--tracker', help='Name of the Object
Tracker To be used.',
                        default=None,
                        choices=[None, 'sort'])
    args = parser.parse_args()

    print('Loading {} with {} labels.'.format(args.model,
args.labels))
    interpreter = common.make_interpreter(args.model)
    interpreter.allocate_tensors()
    labels = load_labels(args.labels)
```

```
    w, h, _ = common.input_image_size(interpreter)
    inference_size = (w, h)
    # Average fps over last 30 frames.
    fps_counter = common.avg_fps_counter(30)

    def user_callback(input_tensor, src_size, inference_box,
mot_tracker):
        nonlocal fps_counter
        start_time = time.monotonic()
        common.set_input(interpreter, input_tensor)
        interpreter.invoke()
        # For larger input image sizes, use the
edgetpu.classification.engine for better performance
        objs = get_output(interpreter, args.threshold, args.top_k)
        end_time = time.monotonic()
        detections = []  # np.array([])
        for n in range(0, len(objs)):
            element = []  # np.array([])
            element.append(objs[n].bbox.xmin)
            element.append(objs[n].bbox.ymin)
            element.append(objs[n].bbox.xmax)
            element.append(objs[n].bbox.ymax)
            element.append(objs[n].score)  # print('element=
',element)
            detections.append(element)  # print('dets: ',dets)
        # convert to numpy array #      print('npdets: ',dets)
        detections = np.array(detections)
        trdata = []
        trackerFlag = False
        if detections.any():
            if mot_tracker != None:
                trdata = mot_tracker.update(detections)
                trackerFlag = True
            text_lines = [
                'Inference: {:.2f} ms'.format((end_time -
start_time) * 1000),
                'FPS: {} fps'.format(round(next(fps_counter))), ]
        if len(objs) != 0:
            return generate_svg(src_size, inference_size,
inference_box, objs, labels, text_lines, trdata, trackerFlag)

    result = gstreamer.run_pipeline(user_callback,
                                    src_size=(640, 480),
                                    appsink_size=inference_size,
                                    trackerName=args.tracker,
                                    videosrc=args.videosrc,
```

	videofmt=args.videofmt) if __name__ == '__main__': main()
Results	The test ran successfully.
Conclusion	The next step in this project is to test this object tracking algorithm and measure the FPS of the SORT algorithm. Furthermore, more model testing within urban environments would be advisable.

Daily Entry #49: Test Sort Algorithm

Title	Test SORT algorithm on Coral Dev Board
Date	January 25, 2022
Signature	*Charles Tang,*
Introduction	The sort algorithm was tested on the videos from Daily Entry #46 for effectiveness. The purpose of this daily entry is to determine whether object tracking algorithms are useful for cyclist detection purposes.
Methods	1. Download videos from Daily Entry #46. 2. Run the model with SORT algorithm from Daily Entry #48 on the videos. 3. Record FPS results and inference latencies.
Results	Table 1. Time | Inference time (ms) | FPS 0 | 132.45 | 7.55 1 | 149.7 | 6.68 2 | 152.97 | 6.54 3 | 131.26 | 7.62 4 | 137.31 | 7.28 Average | 140.74 | 7.13 *Note. This table shows the average inference times and FPS using the SORT tracker algorithm.*
Data Analysis	Although the tracking algorithm seems well used in the journal article testing it on object tracking metrics, it does not seem to perform well in this implementation. Since the cyclist detection models can already run at 10-30 frames per second, the object tracking method slows down the computation speeds by using CPU time on the Google Coral Dev Board. All operations done on the CPU significantly slow down the Dev Board performance, thus the models were compiled for use with the TPU. Furthermore, qualitative testing showed that the cyclist tracking methods were significantly worse than the object detection methods. Thus, it would not be advisable to pursue further cyclist tracking methods in further daily entries.
Conclusion	Future work regarding this project is to put together a final device and to conduct more real-time tests and implement audible alerts to this system.