IADT DUN LAOGHAIRE

DL835 BSc (Hons) in Creative Media Technologies – 2022/2023

Exploring Machine Learning as a Tool to Revolutionize Electrical Circuitry

Final Project Thesis

by Shane Smyth

Exam No: N00193418

Supervisor: Dr. Paul Comiskey

Acknowledgements

I am very grateful and fortunate for the continuous support I have received in the journey of this project throughout the years. I could not have done this project without the invaluable support, guidance and expertise of my supervisor, Dr Paul Comiskey. Thank you for the enjoyable and productive lectures, which have taught me so much in my preferred field. Thank you for being positive and optimistic about teaching.

Thank you to my second reader, Dr Sivakumar Ramachandran, for the constant guidance, support and help throughout the years. Thank you for enhancing my interest in my preferred field and providing me invaluable opportunities throughout my years in college. Your continuous positive and optimistic attitude has made college a fun and enjoyable atmosphere to be in. I am forever grateful.

I cannot thank all my family enough for the continuous moral support and motivation to pursue my interest in this field, along with the emotional support and care throughout this journey. Without them, I could not have done this.

I also want to thank my classmates for the emotional support and help when I needed it most, and always being there for me. Thank you to all my lecturers for guiding and caring for me and thank you to IADT for providing a friendly, supportive facility and community that I enjoyed every day.

Abstract

This thesis explores the novel area of using machine learning to identify electrical circuitry for producing schematic diagrams. Machine learning in electrical circuitry is becoming increasingly important for providing accurate results in the design, analysing and optimization processes; as well as performing these tasks faster than humans. Machine learning specializes in analysing large data for detecting objects and patterns within imagery, which makes it an ideal technology for electrical circuitry automation. The procedure this thesis follows involves capturing and labelling images, training and testing a custom model, exporting the model to a public storage cloud and deploying the model to a website. The detection of electrical components and schematic output within this thesis proves the project should be recognized for the potential of this area for further research.

Declaration

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism. If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/ collaboration is acceptable, you should consult your lecturer or the Course Director. WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not to leave copies of your own files on a hard disk where they can be accessed by others. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended. Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline. Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties. Failure to complete and submit this form may lead to an investigation into your work.

Please sign and include the following at the start of your report:

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student: Shane Smyth

Signed: Shane Smyth

Contents

1 Introduction

Breadboard2Schematic is a website that converts electrical circuitry into a schematic diagram. The aim of this project is to build a new machine learning model that is then deployed to a website and used to detect electrical circuitry within an image. These detections are then used to automatically output a schematic diagram for the user using Artificial Intelligence. The purpose of this project is to prove that artificial intelligence can speed up the process of creating schematic diagrams.

The project plan chapter highlights the project process and methodology, timeline, technologies, purpose of the project and research. The main methodologies of the project include artificial intelligence, machine learning and deep learning. Computer vision is an algorithm that stems from deep learning and is used in image and video processing. Object detection is an algorithm that is a product of computer vision. Object detection can detect objects in images and execute tasks based on these detections. The purpose of this project is to use machine learning to enhance electrical circuitry, to make it faster, accurate and efficient. Artificial intelligence can perform specific tasks faster than humans can. The overall limitation of this project is machine learning. Machine learning is dependent on the data that is provided to the algorithm. The data that is used by machine learning but be of high quality and in high quantities. This is an issue for time management, as this takes a long time for one person and usually requires a team.

In the construction and test of the prototype section, there are three prototypes that were produced during this project. The first prototype was created to store images of electrical components within a folder and label them using a graphical image annotation tool. The second prototype was made to create a new custom model for training and testing. The third prototype exports the custom model to a Tensorflow JavaScript file and then the model is posted on IBM Cloud in a public object storage.

The final project development chapter highlights the main findings of the project. In this chapter, a website is created using a hypertext markup language (HTML) document, cascading style sheet (CSS) file and a JavaScript (JS) file. The HTML document is the main structure of the website that includes most of the content. The CSS file is used to style HTML file. The JavaScript file is used to complete interactions between the user and the website. This is where animations and interactions take place. The final design choices are meant for simplicity and effectiveness. The results conclude that the project does work but is inconsistent with the predictions. The inconsistency can be fixed in the future with model optimization and extra training. The website does not produce a schematic diagram, however, the Jupyter Notebook prototype does. The next steps of the project include optimizing the model and producing schematic diagrams from the outputs.

The user and testing chapter focuses on gathering feedback from the users. There were three users tested in the user testing phase. These users took a pre-test questionnaire that asked them demographic questions and questions that answer if they are familiar with technologies. The users then took the test and the reactions to the website were recorded with their permission. The users then took a post-test qualitative survey. Overall, the first user rated the website a good experience, the second user rated a good experience, and the third user rated the website a bad experience.

2 Project Plan

2.1 Introduction

The project plan outlines the purpose of this project, as well as the limitations. This section aims to answer specific questions and discuss the objectives in relation to the project. The methodology highlights the methods used to achieve results for the project, as well as discuss the technologies used. A timeline and risk assessment are prepared to allow for a scheduled plan of the work that was completed over the year. A discussion of existing projects is also included in this section.

2.2 Methodology

This section describes the methodology and main technologies that were used to prepare and develop this project. The main methodologies include artificial intelligence, machine learning and computer vision. Under these sub-headings describes the fundamentals of these methodologies and technologies.

2.2.1 Artificial Intelligence

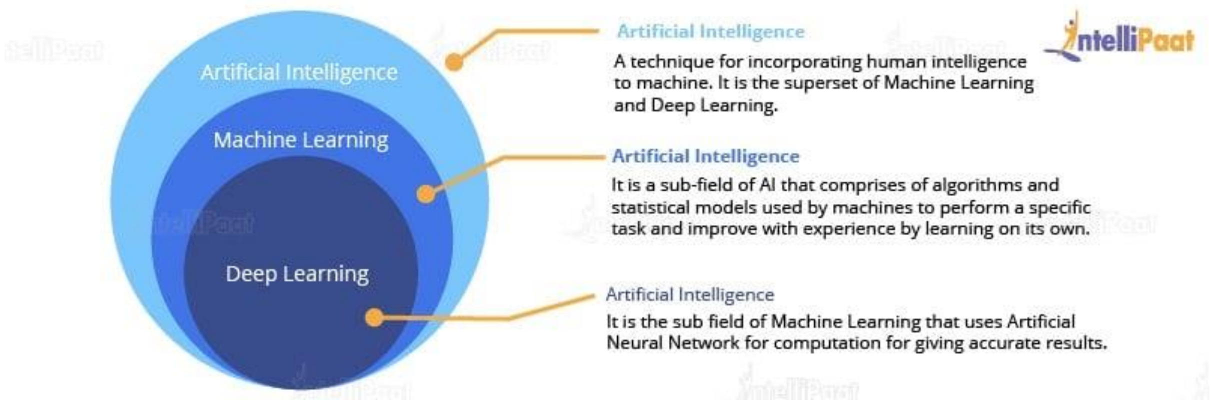

Artificial Intelligence
Machine Learning
Deep Learning
Artificial Intelligence
A technique for incorporating human intelligence to machine. It is the superset of Machine Learning and Deep Learning.
Artificial Intelligence
It is a sub-field of AI that comprises of algorithms and statistical models used by machines to perform a specific task and improve with experience by learning on its own.
Artificial Intelligence
It is the sub field of Machine Learning that uses Artificial Neural Network for computation for giving accurate results.
IntelliPaat

Figure 1.0: The branches of Artificial Intelligence (*Retrieved from https://intellipaat.com/blog/wp-content/uploads/2018/07/ALMLDL-01.jpg*)

Figure 1.0 shows a diagram that represents artificial intelligence (AI) and the branches of AI, machine learning and deep learning. Artificial intelligence is the main technology that has two subsets of computer learning, these subsets are known as machine learning and deep learning. Artificial intelligence (AI) is the use of computers to automate tasks that commonly need human intelligence (Russell, S. J., & Norvig, P. (2010)). Machine learning is derived from AI and can be described as a technology that can learn from the data the developer gives it. Machine learning allows for the computer to recognize patterns in images and video. Giving more data to the machine increases the machine's knowledge and optimizes the machine to complete tasks faster and more effectively (Samuel, A. L. (1959)). Machine learning is main technology used in this project and offers algorithms such as object detection.

Deep learning is a subset of machine learning that uses neural networks. These neural networks are designed to replicate the human brain's neural network. This technology is responsible for processes such as image processing, computer vision and speech recognition (Goodfellow, I., Bengio, Y., & Courville, A. (2016)). Computer vision allows for machines to analyse and gather useful data and meaning from images and video. This technology is responsible for face recognition, object detection and for medical image detection (Russell, S. J., & Norvig, P. (2010)).

2.2.2 Object Detection

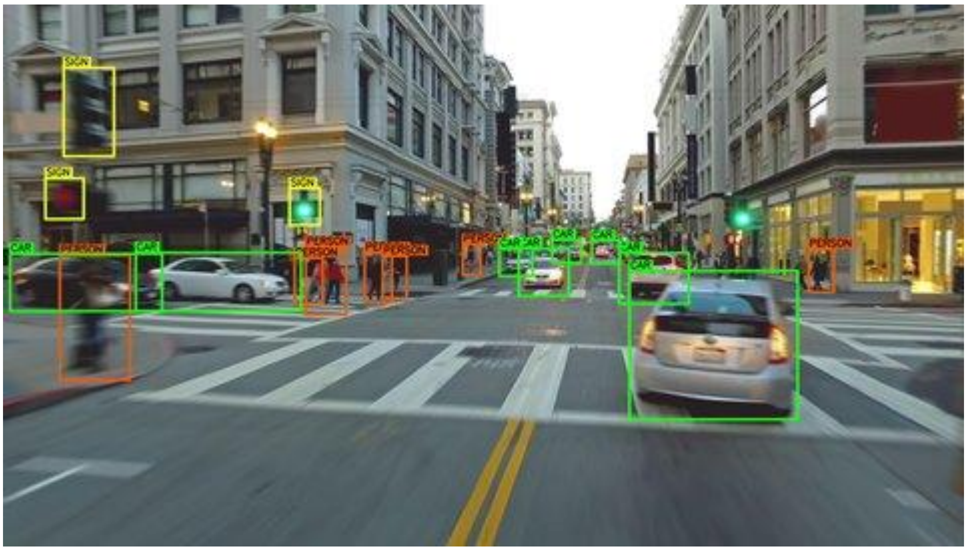


Figure 1.1: Object Detection (*Image retrieved from* https://www.shutterstock.com/shutterstock/videos/1064002759/thumb/1.jpg?ip=x480)

Object detection involves the computer detecting objects within an image or video. The location of the objects, presence and type of objects are detectable in object detection. As can be seen in Figure 1.1, object detection can be used with autonomous vehicles to detect the presence of street objects, type of street object and location of the street object, which can be used by the vehicle to perform tasks with caution, such as driving safely and by itself (Szeliski, R. (2010)).

2.3 The Purpose of Breadboard2Schematic

The purpose of this project is to use a novel technique of applying machine learning to electrical circuitry to produce schematic diagrams. In relation to electrical circuitry, a schematic diagram presents a graphical representation of a circuit (National Instruments. (n.d.)). In the context of this project, machine learning allows for the automation of schematic diagrams in an effective and accurate process. Machine learning allows for the designing of schematic diagrams to be faster than humans.

2.4 Project Limitations

The project limitations revolve around the limitations of machine learning. Machine learning relies on large amounts of quality data. This causes difficulties because it is time consuming to create a new model would commonly require a team. The machine learning model can produce a biased outcome, which can lead to the model remembering data and basing the outcomes on that and producing incorrect data in the future (Piatetsky-Shapiro, G. (2021)).

2.5 Research and Existing Similar Projects

2.5.1 Tensorflow Object Detection Tutorial by NickNochNack

The Github tutorial from GitHub - nicknochnack/TFODCourse (Renotte, 2022) is a guide for object detection in Tensorflow. The aim of this tutorial is to create a program that detects any object on a camera once the object is trained. The first step of training these objects involves creating a virtual environment on the computer. Images of the object are required for training. These images are captured using a camera and then labelled and placed inside a folder in the virtual environment. The labelled images are trained, and the objects are then detected from the given image. This tutorial is related to the project because it follows similar steps in training images, using a computer neural network. It is also related because one of the outcomes of training in the tutorial detects the anode and cathode legs of an LED.

2.5.2 Tensorflow Tutorial

The Tensorflow website TensorFlow 2 Object Detection API tutorial — TensorFlow 2 Object Detection API tutorial documentation (tensorflow-object-detection-api-tutorial.readthedocs.io) has a full documentation related to computer neural networks and artificial intelligence in general. This documentation is related to the project because the same procedure is followed for training a new model. This website has the official Tensorflow library documentation and explains each step to training a model. This documentation suggests using LabelImg, a graphical image annotation tool, which allows for drawing a boundary box around the area of the object. Similarly in this project, LabelImg is the chosen tool for this task.

2.5.3 About Tensorflow Lite

Tensorflow Lite (TFLite) https://www.tensorflow.org/lite/guide is a package that allows for transferring and using a model on Android, iOS and other devices such as microcontrollers. This is useful for this project because the completed model can be used on mobiles devices, allowing for use of taking pictures of circuits on a mobile phone camera for convenience. This also allows for high performance and model optimization within the project.

2.5.4 Users of Breadboard2Schematic

The users of this application include students for general education purposes. This project allows for the learning of circuitry and electronic components. It is also useful for users in the manufacturing industry and areas of technology as it provides a fast way of drawing schematics. Schematics can be large and difficult to understand so information is useful for the user to make sense of them. Large-scale and small-scale schematics can be designed faster than other ways by using artificial intelligence, which is useful for users that aim to design schematics in a short period of time. It can also be difficult to visualize a circuit as a schematic drawing when looking at a breadboard, so this project can help the users visualize and understand schematics.

2.5.5 Project Requirements

This project aims to process an image of a breadboard circuit, and then convert the image from breadboard circuit to a schematic drawing. The outcome is designed in website format with a graphical user interface that can be navigated with ease. When an image is taken or a pre-existing image is used, this image is processed and converted by the press of a button. After processing, the outcome is displayed on a screen for the user to use. The aim of this project is to make the graphical user interface navigable and for the processing of the image to be quick and effective for the user. The first aim is to create a virtual environment that isolates a file section for using different versions of packages such as downloading and using a previous version of Tensorflow, which can be used in this virtual environment only.

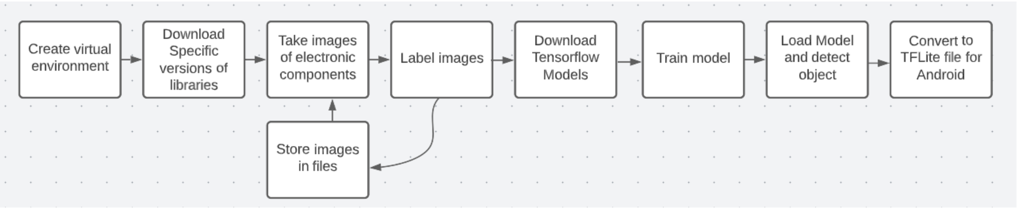

Create virtual environment
Download Specific versions of libraries
Take images of electronic components
Label images
Download Tensorflow Models
Train model
Load Model and detect object
Convert to TFLite file for Android
Store images in files

Figure 1.2: Code requirements

Figure 1.2 shows the code requirements. The aim of the code is to take images of electronic components and draw a boundary box around the object in each image. These images are then labelled and stored in a file inside the virtual environment. A Tensorflow Pretrained Model is downloaded along with different Tensorflow models. These labelled images are used for training the model. After training, the model is loaded, and an object is detected. Finally, the model is converted to TFLite and used in app format on Android.

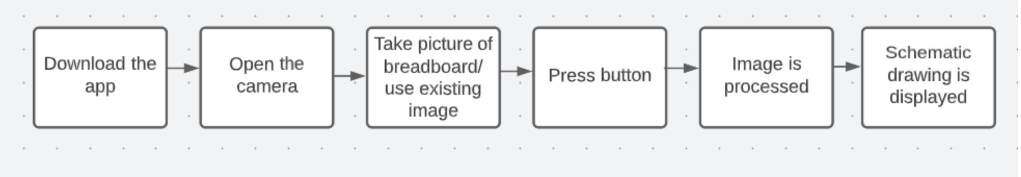

Download the app
Open the camera
Take picture of breadboard/ use existing image
Press button
Image is processed
Schematic drawing is displayed

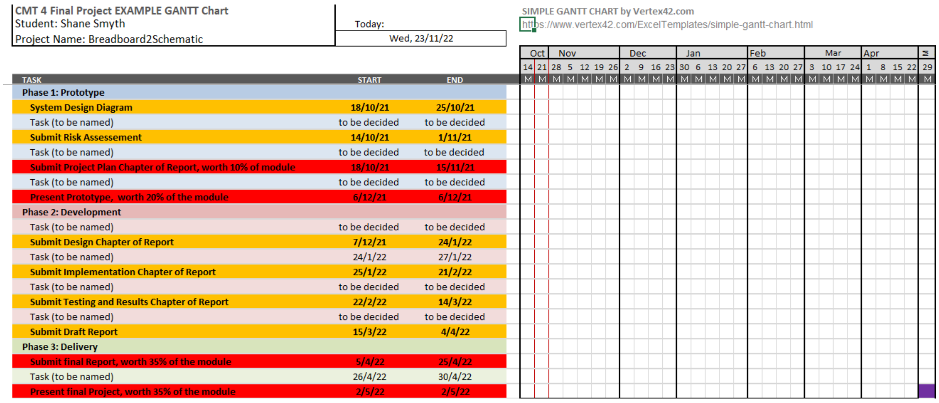
Figure 1.3: User experience requirements

Figure 1.3 shows the client side of the application. First, the user downloads the application. The user is then prompted to open the camera and then take a picture of a circuit or use an existing image. A button is pressed on the graphical user interface. This button takes the selected image and processes it by using the model. The outcome is then displayed as a schematic drawing of the circuit. The user can then select a different image to process or end the application. Information is also given relating to the circuit.

2.5.6 Technologies

The technologies used in this project include a camera, Python, Tensorflow, LabelImg, os, matplotlib, numpy and schemdraw. Python is a useful programming language for machine learning. Python is easy to use and accessible, making it a popular choice for programming language. Tensorflow is a machine learning library that is supported by Python, commonly used for artificial intelligence. LabelImg is a graphical image annotation tool that can be used to label images and surround objects in an image with boundary boxes, helping the machine learn a new object. Os is a library that is used for multiple purposes related to operating system programming. This library can be used for creating new directories and files on a computer. This is useful for virtual environments, where the files are isolated from the rest of the computer's files. Matplotlib is a scientific library used for creating graphs in Python. These graphs can be used in artificial intelligence to understand and visualize tensors. Numpy is a large library used for scientific computation. This library is used for multiple purposes and many aspects from Tensorflow depend on numpy for its use of more effective arrays than the regular arrays in Python. Schemdraw is a library that can draw schematics using Python.

2.5.7 Project Timeline Plan



CMT 4 Final Project EXAMPLE GANTT Chart
Student: Shane Smyth
Project Name: Breadboard2Schematic

Today: Wed, 23/11/22

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

TASK	START	END
Phase 1: Prototype		
System Design Diagram	**18/10/21**	**25/10/21**
Task (to be named)	to be decided	to be decided
Submit Risk Assessement	**14/10/21**	**1/11/21**
Task (to be named)	to be decided	to be decided
Submit Project Plan Chapter of Report, worth 10% of module	**18/10/21**	**15/11/21**
Task (to be named)	to be decided	to be decided
Present Prototype, worth 20% of the module	**6/12/21**	**6/12/21**
Phase 2: Development		
Task (to be named)	to be decided	to be decided
Submit Design Chapter of Report	**7/12/21**	**24/1/22**
Task (to be named)	24/1/22	27/1/22
Submit Implementation Chapter of Report	**25/1/22**	**21/2/22**
Task (to be named)	to be decided	to be decided
Submit Testing and Results Chapter of Report	**22/2/22**	**14/3/22**
Task (to be named)	to be decided	to be decided
Submit Draft Report	**15/3/22**	**4/4/22**
Phase 3: Delivery		
Submit final Report, worth 35% of the module	**5/4/22**	**25/4/22**
Task (to be named)	26/4/22	30/4/22
Present final Project, worth 35% of the module	**2/5/22**	**2/5/22**

Oct Nov Dec Jan Feb Mar Apr M
14 21 28 5 12 19 26 2 9 16 23 30 6 13 20 27 6 13 20 27 3 10 17 24 1 8 15 22 29
M M M M M M M M M M M M M M M M M M M M M M M M M M M M M

Figure 1.4: GANTT Chart for Final Project

(*Simple Gantt Chart*, n.d.)

The GANTT chart in Figure 1.4 is the timeline plan for the project during this year. The GANTT chart consists of a series of materials to submit over time. These first stage of the materials to complete include creating a system design diagram and completing a risk assessment. The next step is to write the project plan chapter. The final stage of phase one is to present the project. The second phase involves completing the design chapter, implementation chapter, testing and results chapter and preparing a draft report. The last phase involves completing the final report and presenting the final project.

2.6 Conclusion

This section summarizes the plan of the project to be conducted during the year. The project plan consists of a methodology, which discusses the main technological methods: artificial intelligence, machine learning and deep learning. The limitations and difficulties were discussed, and the project timeline was planned to use a GANTT chart to keep track of the aims of the project during the year.

3 Construction and Test of the Prototype

3.1 Introduction

This section describes the construction and testing of the three prototypes that were built before the final project development. The first prototype focuses on the image collection and labelling before the custom model is trained and tested. The main technologies of this include OpenCV, uuid, os, time, PyQt and LabelImg. OpenCV is used in this prototype to open the external camera and capture images. Uuid was used to give each image a unique identification, to prevent each image from overwriting names. The os module was used to create, move and manipulate folders and files to specific locations within the file manager. The time module was used to give the developer a chance to adjust the camera between each photo if needed. LabelImg is an annotation tool that was used to label each object in an image, to prepare them for training and testing. This technology required PyQt5 to run, which is a graphical user interface module that provides developers with tools to build software applications. The main procedure of the first prototype included importing libraries, creating image labels, setting up the folders, capturing the images and labelling the images.

The second prototype explores the process of creating the custom model. The main procedure of this prototype included setting up the paths for the custom model, downloading the pre-trained model and Tensorflow Object Detection, creating the label map, creating TF records for training and testing, moving the model configuration to the testing folder, updating the configuration, training the model, loading the model and checkpoints, and detecting the objects in the image. The main technologies in this prototype are Schemdraw, OpenCV, Numpy, Tensorflow, Keras, OS and Matplotlib. Schemdraw drew the schematics in the post-processing. Numpy created a Numpy array of the image for post-processing. OpenCV allowed for image pre-processing. Tensorflow allowed for the custom model object detection, training and testing. OS allowed for file creation and navigation. Matplotlib created a visualization of the result.

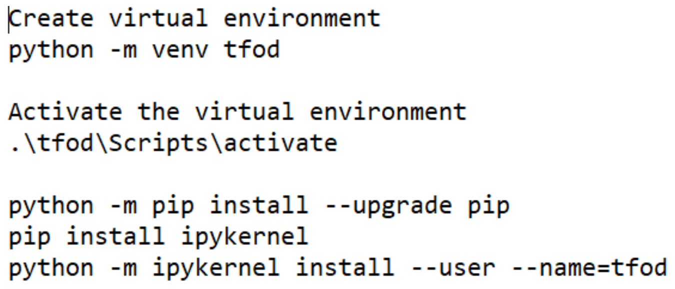
The third prototype freezes the model graph for optimization and converts the custom model into a Tensorflow JavaScript file for exporting. The model.json file and model bin files are then posted to IBM Cloud to the public, inside a bucket. A command is then executed to allow the model file to be shared on other applications, such as the JavaScript websites. This bucket link is then used to make the model function on the website.

3.2 Prototype 1

The aim of the first prototype was to achieve an application that can collect images, store them in specific file locations and label these images on an interactive computing platform. An interactive computing platform is a technology that allows the user to write code within an environment, as well as interpret and visualize data (Kluyver et al., n.d.). Jupyter Notebook (Jupyter, 2019) was the chosen interactive computing platform, which was used for executing and building the architecture of the code. Jupyter notebook is a web-based interface, used for scientific computing (Kluyver et al., n.d.).

3.2.1 Virtual Environment

Before accessing Jupyter Notebook, the first step of the prototype was to create a virtual environment. A virtual environment is a computer programming tool used to isolate packages and libraries in a project (Chen, 2020). This allows for the creation of multiple projects on one system that may require different library packages and package versions, without the library versions and packages conflicting with each other (Chen, 2020).

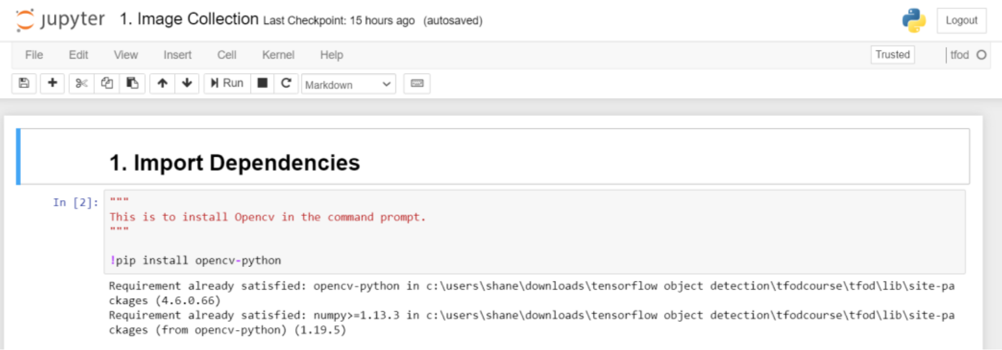


```
Create virtual environment
python -m venv tfod

Activate the virtual environment
.\tfod\Scripts\activate

python -m pip install --upgrade pip
pip install ipykernel
python -m ipykernel install --user --name=tfod
```

Figure 1.5: Creating a virtual environment.

Figure 1.5 shows the necessary commands to create a virtual environment using the command prompt. First, a virtual environment is declared using venv, which stands for virtual environment. The virtual environment is then given the name ‘tfod’, which stands for ‘Tensorflow Object Detection’. After this, the virtual environment is activated inside the Scripts folder. To add this virtual environment to the Python kernel, ipykernel (Team, n.d.) and pip (Pip, 2019) must be installed, and pip must be upgraded to the latest version. The final command adds the environment to the Python kernel. A Python kernel from ipykernel is required for Jupyter to interact with Python code (Kluyver et al., n.d.). Pip is used as an easy way for developers to install and uninstall packages for Python. It can be used to list all packages installed and their versions (Python Packaging Authority, n.d.). The ‘m’ command means module and specifies that the module must be ran as a script (Python Software Foundation. (n.d.)). The ‘--user’ command is used for pip and allows for the installation of packages to the specified user on the computer. This prevents the need for administrator access (Python Packaging Authority, 2022). After installing Jupyter Notebook, typing ‘jupyter notebook’ inside this virtual environment launches Jupyter Notebook (Renotte, 2022).

3.2.2 Jupyter Notebook



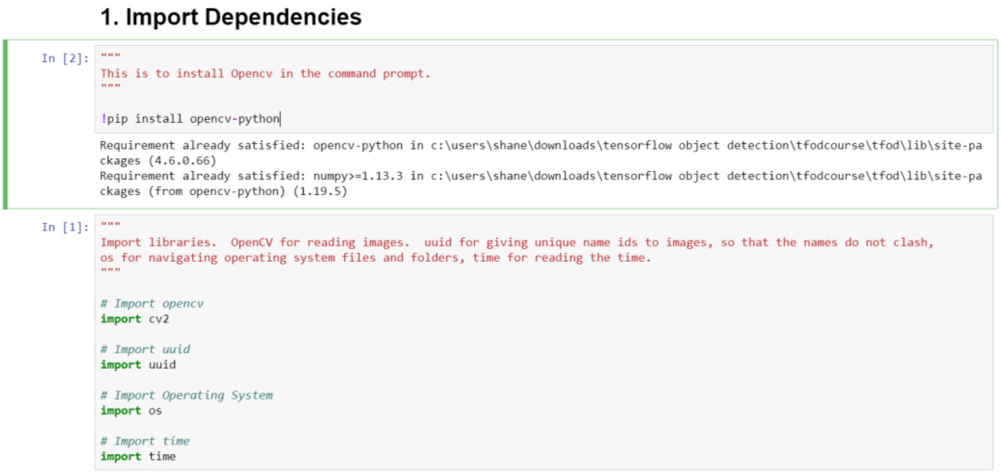
1. Import Dependencies

```
In [2]: """
        This is to install Opencv in the command prompt.
        """

        !pip install opencv-python

        Requirement already satisfied: opencv-python in c:\users\shane\downloads\tensorflow object detection\tfodcourse\tfod\lib\site-pa
        ckages (4.6.0.66)
        Requirement already satisfied: numpy>=1.13.3 in c:\users\shane\downloads\tensorflow object detection\tfodcourse\tfod\lib\site-pa
        ckages (from opencv-python) (1.19.5)
```

Figure 2: Jupyter Notebook Tools and interface

Figure 2 showcases the Jupyter Notebook interface. This application is especially useful for machine learning because it can execute code in individual blocks, as well as visualize and change data effectively. This prototype used a new Jupyter Notebook file, ‘Image Collection’, for the purpose of gathering images to prepare for training and testing in the further prototypes. At the top right corner of Figure 2, ‘tfod’ indicates that the Jupyter Notebook file was successfully created inside the virtual environment that uses a Python kernel. Jupyter Notebook uses blocks of code that are known as cells. These cells are used to write the code and any code inside a cell block runs together as one line of code. The top left side of Figure 2 presents a selection of tools that can be used to manipulate the cells in the program. There are tools to create new cell blocks, cut existing cell blocks, copy and paste cell blocks, move a cell up or down, run the cell block code individually, and stop the execution of a cell block (Renotte, 2022).

3.2.3 Packages for image collection



1. Import Dependencies

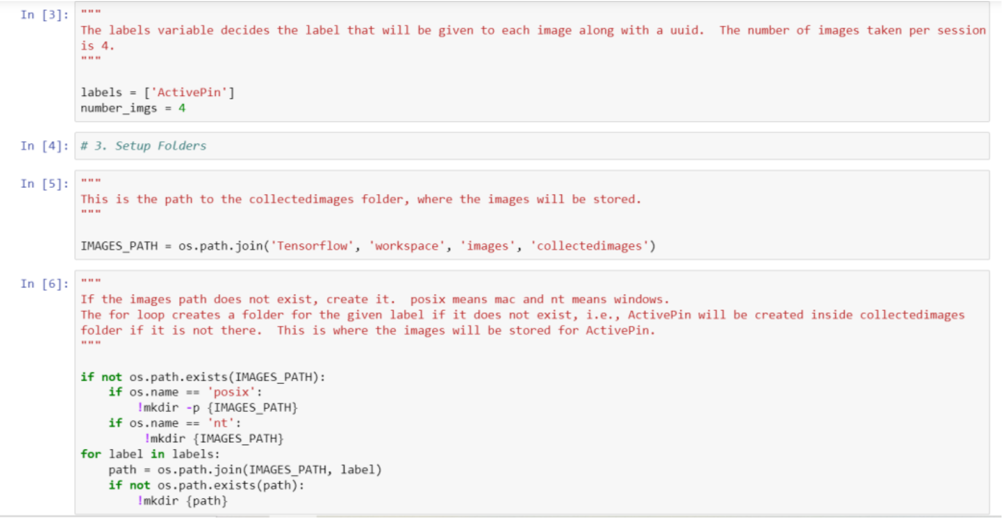
```
In [2]: """
        This is to install Opencv in the command prompt.
        """

        !pip install opencv-python

        Requirement already satisfied: opencv-python in c:\users\shane\downloads\tensorflow object detection\tfodcourse\tfod\lib\site-pa
        ckages (4.6.0.66)
        Requirement already satisfied: numpy>=1.13.3 in c:\users\shane\downloads\tensorflow object detection\tfodcourse\tfod\lib\site-pa
        ckages (from opencv-python) (1.19.5)
```

```
In [1]: """
        Import libraries.  OpenCV for reading images.  uuid for giving unique name ids to images, so that the names do not clash,
        os for navigating operating system files and folders, time for reading the time.
        """

        # Import opencv
        import cv2

        # Import uuid
        import uuid

        # Import Operating System
        import os

        # Import time
        import time
```

Figure 3: Library Imports for Image Collection

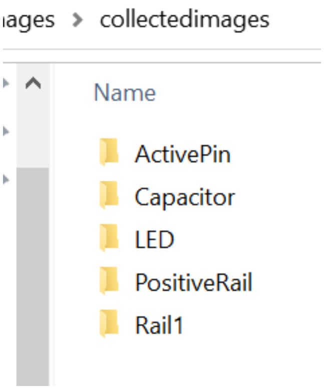
The first cell in Figure 3 is used to install OpenCV (OpenCV, 2019) for Python. OpenCV is an open-source computer vision library, created by Intel in 1999 (OpenCV, n.d.). In this project, OpenCV is used for machine learning, image processing and object detection. The second cell in Figure 3 is used for importing libraries to the program. Cv2 (OpenCV), uuid, os and time are the imported libraries. Uuid stands for universally unique identifier and is a package used for creating unique identifications for images. This is useful for machine learning because there can be a lot of images required for a machine learning model that need unique names to prevent the image names from overwriting each other (Leach, Mealling, & Salz, 2005). Os is a library that allows the developer to navigate the operating system. This library can be used for changing files or folder locations, creating new folders and change existing folders and files Python Software Foundation. (2021). Time is a module in Python that has time-related functions. This module can get the current time and freeze the program to create a delay for a set amount of time ((Python Software Foundation, 2021; (Renotte, 2022))).

3.2.4 Object Folder Creation



```
In [3]: """
        The labels variable decides the label that will be given to each image along with a uuid.  The number of images taken per session
        is 4.
        """

        labels = ['ActivePin']
        number_imgs = 4

In [4]: # 3. Setup Folders

In [5]: """
        This is the path to the collectedimages folder, where the images will be stored.
        """

        IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')

In [6]: """
        If the images path does not exist, create it.  posix means mac and nt means windows.
        The for loop creates a folder for the given label if it does not exist, i.e., ActivePin will be created inside collectedimages
        folder if it is not there.  This is where the images will be stored for ActivePin.
        """

        if not os.path.exists(IMAGES_PATH):
            if os.name == 'posix':
                !mkdir -p {IMAGES_PATH}
            if os.name == 'nt':
                 !mkdir {IMAGES_PATH}
        for label in labels:
            path = os.path.join(IMAGES_PATH, label)
            if not os.path.exists(path):
                !mkdir {path}
```

Figure 4: Creating folders for each object class.

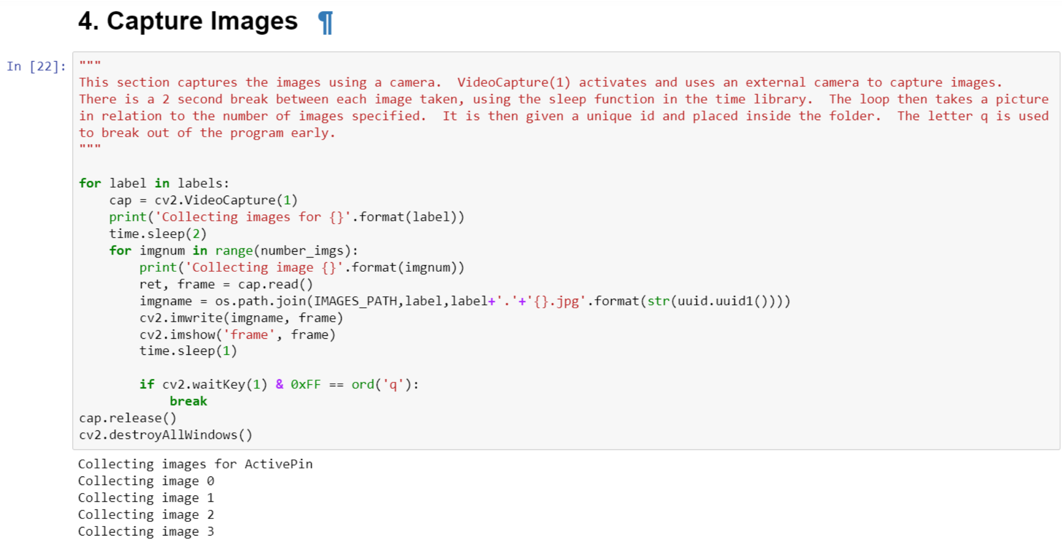


ActivePin
Capacitor
LED
PositiveRail
Rail1

Figure 5: Object folders

Figure 4 consists of the code for gathering images. A list ‘labels’ was used to store the object names that were detected in the images, i.e., ‘ActivePin’, ‘LED’, ‘Capacitor’ are objects that were detected in the images. This variable can be changed to add more objects to the project. The number of images taken were 4, but this integer value can be changed to any amount. In the next code block, OS is used to store the pathway location of the collected images folder. A conditional statement is used to check if the image pathway exists within the file manager on the operating system. If the pathway does not exist, the program makes the directory for the developer using the mkdir command. There is also a nested conditional statement that checks if the user is on mac or pc. This is necessary because the commands differ on mac and pc. There is a for loop to create a new folder for each object inside the collected images folder, i.e., an ActivePin folder was created inside the collected images. These folders were used to store each collected image and can be seen in Figure 5 (Renotte, 2022).

3.2.5 Object Image Collection



4. Capture Images ¶

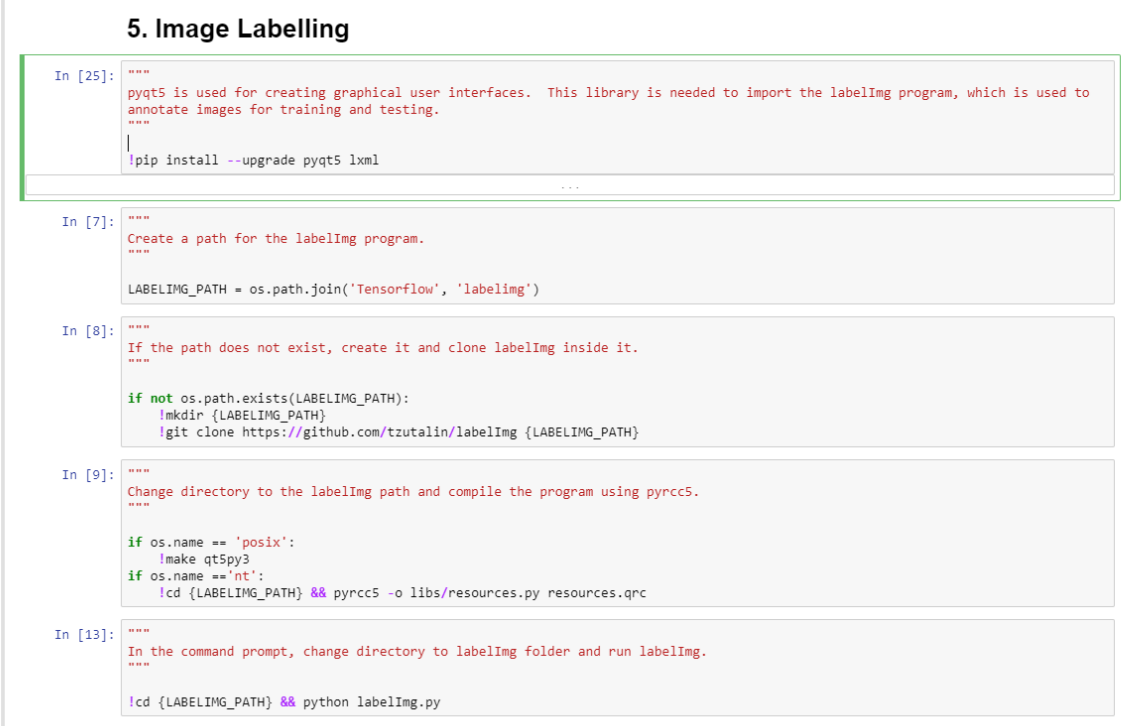
```
In [22]: """
This section captures the images using a camera.  VideoCapture(1) activates and uses an external camera to capture images.
There is a 2 second break between each image taken, using the sleep function in the time library.  The loop then takes a picture
in relation to the number of images specified.  It is then given a unique id and placed inside the folder.  The letter q is used
to break out of the program early.
"""

for label in labels:
    cap = cv2.VideoCapture(1)
    print('Collecting images for {}'.format(label))
    time.sleep(2)
    for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
        ret, frame = cap.read()
        imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(1)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

```
Collecting images for ActivePin
Collecting image 0
Collecting image 1
Collecting image 2
Collecting image 3
```

Figure 6: Capturing the Images.

In Figure 6, a loop is used to capture images for each object. OpenCV video capture is used to open the selected camera. In this case, 1 is used to specify an external camera. 0 is used for a web camera. There is a nested for loop that captures the images presented on the external camera, by turning the camera on using cap.read() function and saving them using imwrite() function. This repeats for the number of images desired to be taken. Each image is formatted with a title that includes the name of the object combined with the unique identification numbering. These images are then stored inside their related object folders using os. The time module is used in this loop to pause the program for a few seconds between each image captured, to give the developer time to re-adjust the camera if it is necessary. The developer must press 'q' on the keyboard to exit the program early if needed. The progress is printed as strings inside the terminal for the developer to keep track of the object being collected and the image number (Renotte, 2022).

3.2.6 Object Labelling Requirements



5. Image Labelling

```
In [25]: """
         pyqt5 is used for creating graphical user interfaces.  This library is needed to import the labelImg program, which is used to
         annotate images for training and testing.
         """

         !pip install --upgrade pyqt5 lxml
```

```
In [7]:  """
         Create a path for the labelImg program.
         """

         LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')
```

```
In [8]:  """
         If the path does not exist, create it and clone labelImg inside it.
         """

         if not os.path.exists(LABELIMG_PATH):
             !mkdir {LABELIMG_PATH}
             !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}
```

```
In [9]:  """
         Change directory to the labelImg path and compile the program using pyrcc5.
         """

         if os.name == 'posix':
             !make qt5py3
         if os.name =='nt':
             !cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc
```

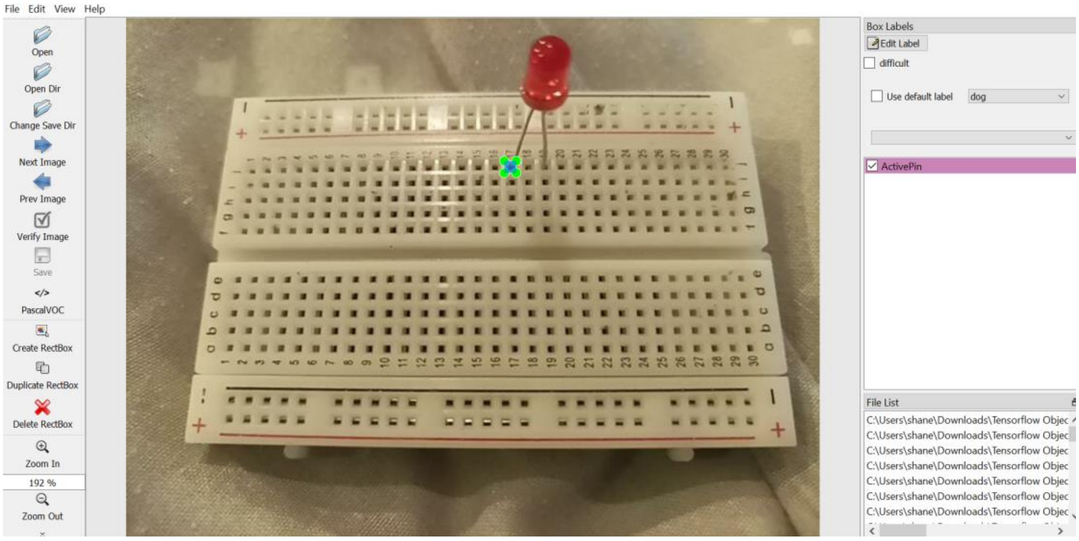
```
In [13]: """
         In the command prompt, change directory to labelImg folder and run labelImg.
         """

         !cd {LABELIMG_PATH} && python labelImg.py
```

Figure 7: Preparing images for labelling.

The first cell in Figure 7 is used to install Pyqt5 and LXML. Pyqt5 allows for the development of graphical user interfaces (GUI) in Python using the Qt framework. This framework has a toolkit for developers that provides widgets, toolbars and several other tools (Riverbank Computing, n.d.). LXML is for HTML and XML document processing. It is helpful for its ease of use, manipulating and creating XML and HTML documents (lxml, n.d.). These libraries are required to use LabelImg. LabelImg is a graphical image annotation tool that provides a toolkit for labelling objects in an image by surrounding the object with a bounding box (Lin, T. (2018)). Bounding boxes are rectangular boxes which surround the target object in an image. For this project, it is used for object detection purposes to identify and locate the targeted objects within an image (Ren et al., 2015; Redmon et al., 2016). In

Figure 7 in the second cell, a new folder 'Labelimg' is created inside the Tensorflow folder. If the folder does not exist, the program makes the directory and LabelImg is cloned inside it from Github, which is a platform that developers use to share and post source code (GitHub, n.d.). Pyrcc5 was then downloaded as output (-o), which compiles Qt resource files into Python files for use in Pyqt5 (Riverbank Computing Limited. (n.d.)). Finally, the directory was changed to the LabelImg path and executed, causing LabelImg to launch (Renotte, 2022).

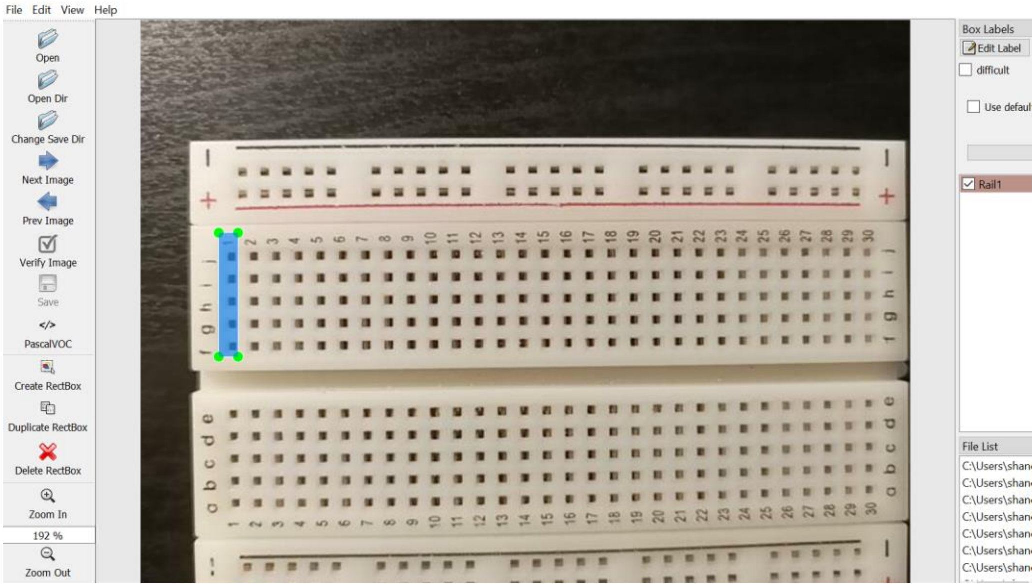
3.2.7 Object Labelling



☑ ActivePin

C:\Users\shane\Downloads\Tensorflow Objec
C:\Users\shane\Downloads\Tensorflow Objec
C:\Users\shane\Downloads\Tensorflow Objec
C:\Users\shane\Downloads\Tensorflow Objec
C:\Users\shane\Downloads\Tensorflow Objec
C:\Users\shane\Downloads\Tensorflow Objec
C:\Users\shane\Downloads\Tensorflow Objec

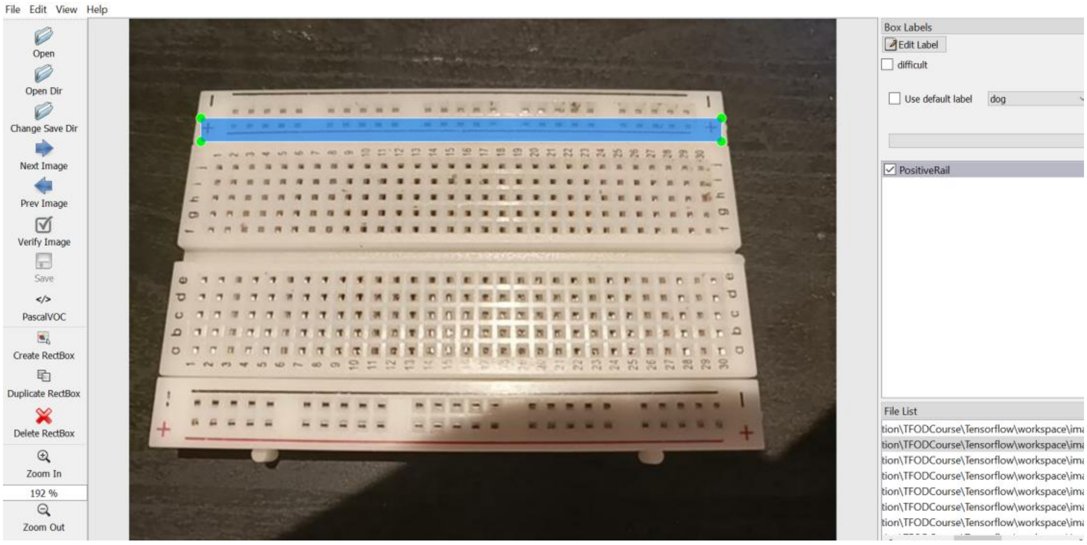
Figure 8: LabelImg ActivePin.



☑ Rail1

C:\Users\shan
C:\Users\shan
C:\Users\shan
C:\Users\shan
C:\Users\shan
C:\Users\shan
C:\Users\shan

Figure 9: LabelImg Rail1



□ difficult

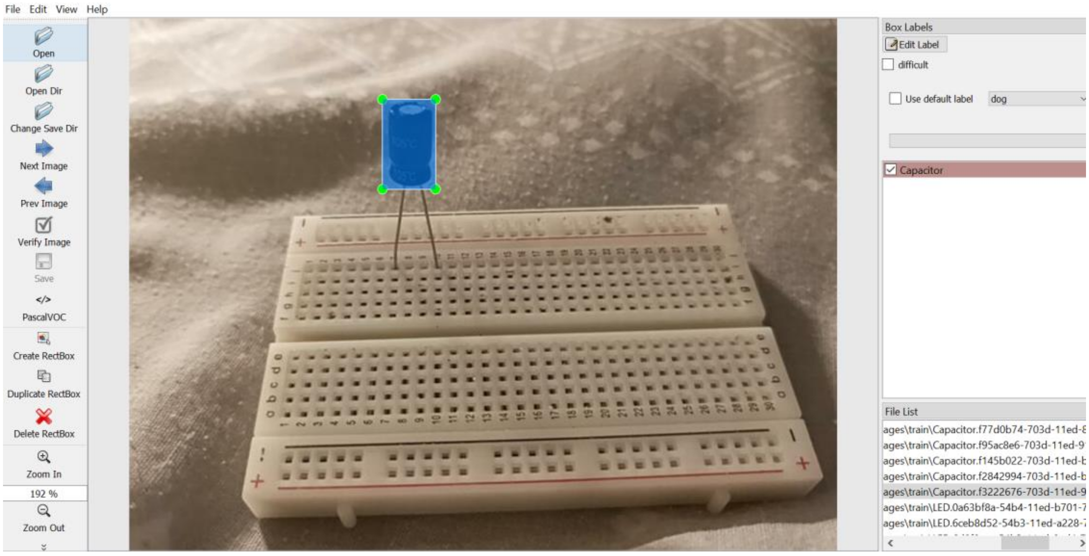
□ Use default label dog

☑ PositiveRail

File List

```
tion\TFODCourse\Tensorflow\workspace\ima
tion\TFODCourse\Tensorflow\workspace\ima
tion\TFODCourse\Tensorflow\workspace\ima
tion\TFODCourse\Tensorflow\workspace\ima
tion\TFODCourse\Tensorflow\workspace\ima
tion\TFODCourse\Tensorflow\workspace\ima
tion\TFODCourse\Tensorflow\workspace\ima
```

Figure 10: LabelImg PositiveRail



□ difficult

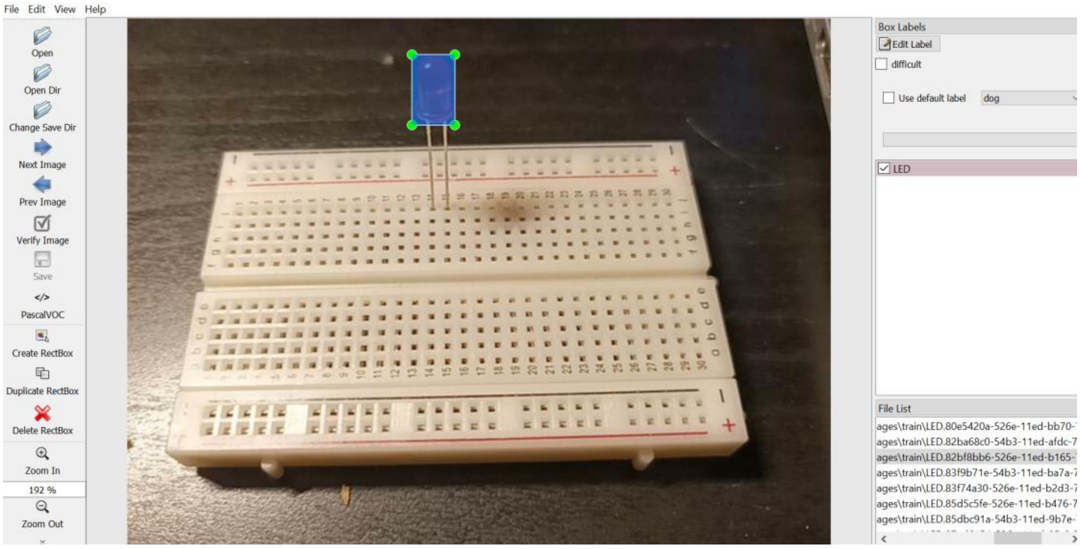
□ Use default label dog

☑ Capacitor

File List

```
ages\train\Capacitor.f77d0b74-703d-11ed-
ages\train\Capacitor.f95ac8e6-703d-11ed-9
ages\train\Capacitor.f145b022-703d-11ed-b
ages\train\Capacitor.f2842994-703d-11ed-b
ages\train\Capacitor.f3222676-703d-11ed-9
ages\train\LED.0a63bf8a-54b4-11ed-b701-7
ages\train\LED.6ceb8d52-54b3-11ed-a228-
```

Figure 11: LabelImg Capacitor



ages\train\LED.80e5420a-526e-11ed-bb70-
ages\train\LED.82ba68c0-54b3-11ed-afdc-7
ages\train\LED.82bf8bb6-526e-11ed-b165-
ages\train\LED.83f9b71e-54b3-11ed-ba7a-7
ages\train\LED.83f74a30-526e-11ed-b2d3-7
ages\train\LED.85d5c5fe-526e-11ed-b476-7
ages\train\LED.85dbc91a-54b3-11ed-9b7e-

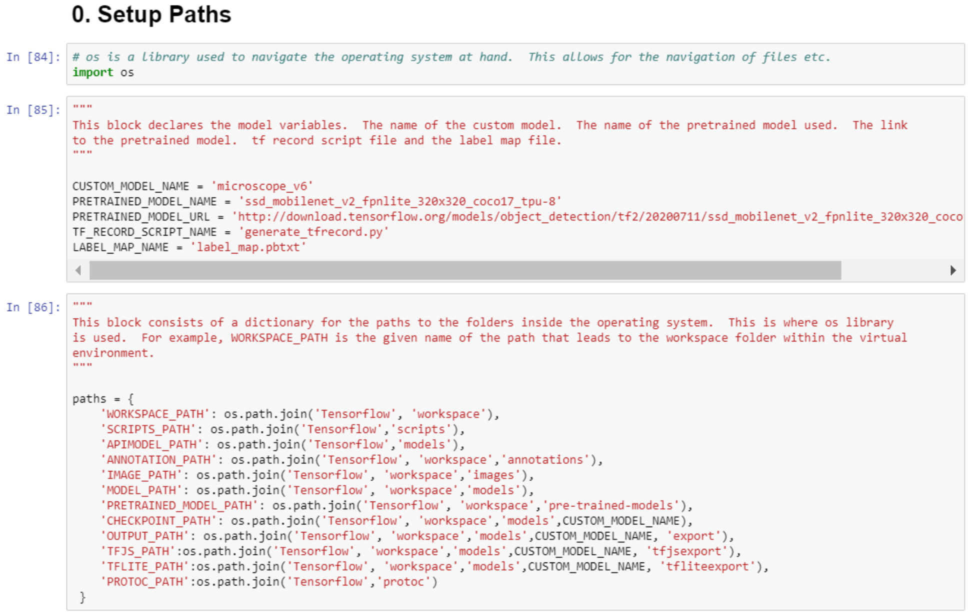
Figure 12: LabelImg LED

As shown in Figure 8, LabelImg is a graphical image annotation tool that allows developers to draw and label bounding boxes around desired objects within an image, to prepare them for training and testing in the later stages of a custom machine learning model. The left-most side of Figure 8 provides a set of tools for the developer. The developer can open folders from the operating system file manager, navigate images in the folder, create bounding boxes, duplicate bounding boxes, delete bounding boxes, and other tools such as zooming in and out of images or adjusting the brightness to better the production of the training. As can be seen in Figure 8, a rectangular bounding box was drawn around one of the pins of the light emitting diode (LED) and labelled ‘ActivePin’. This is to prove that one of the LED pins is occupied in a breadboard rail, which could indicate that an LED is active in a schematic diagram. Figure 9 shows the labelling of Rail1, which is used to locate the position of the first rail on the breadboard, its purpose is to solve whether components are inside this rail. Figure 10 shows a labelling of the positive rail on a breadboard. This is to find the location of the positive rail on the breadboard, to check if there are any components inside this rail that are receiving power. Figure 11 and Figure 12 shows the capacitor and LED labelling respectively. This is to determine the type of component that is detected in the image (Renotte, 2022).

3.3 Prototype 2

The goal of prototype 2 was to use the images collected from prototype 1 to train and test the custom model. After training and testing the model, the model is then used to make predictions and detections on the dataset, as well as drawing a schematic diagram from the detections. In this section, a custom model with custom dataset is created, trained, tested and ready to be deployed to a website. The procedure of this prototype is as follows: create the path directories for the custom model, download pretrained model and Tensorflow Object Detection from Tensorflow, create a label map, create TF record files, adjust model configurations, update the configurations, train and test the model.

3.3.1 Creating Custom Model Path



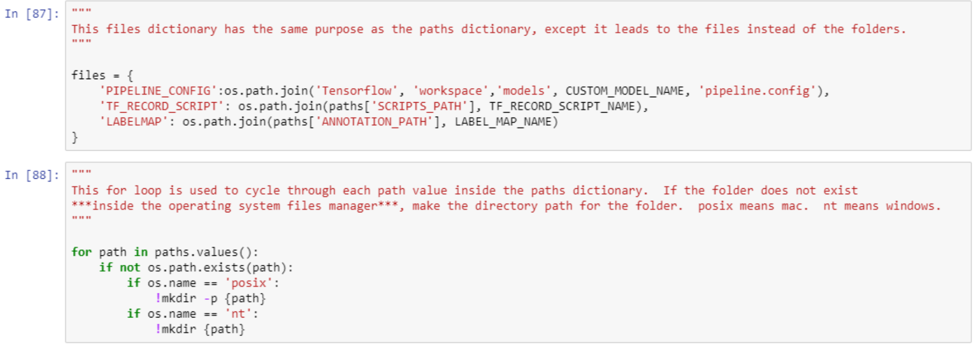
0. Setup Paths

```
In [84]: # os is a library used to navigate the operating system at hand.  This allows for the navigation of files etc.
         import os
```

```
In [85]: """
         This block declares the model variables.  The name of the custom model.  The name of the pretrained model used.  The link
         to the pretrained model.  tf record script file and the label map file.
         """

         CUSTOM_MODEL_NAME = 'microscope_v6'
         PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
         PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco
         TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
         LABEL_MAP_NAME = 'label_map.pbtxt'
```

```
In [86]: """
         This block consists of a dictionary for the paths to the folders inside the operating system.  This is where os library
         is used.  For example, WORKSPACE_PATH is the given name of the path that leads to the workspace folder within the virtual
         environment.
         """

         paths = {
             'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
             'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
             'APIMODEL_PATH': os.path.join('Tensorflow','models'),
             'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
             'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
             'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
             'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
             'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
             'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'export'),
             'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
             'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
             'PROTOC_PATH':os.path.join('Tensorflow','protoc')
          }
```

Figure 13: Setting up model folders.



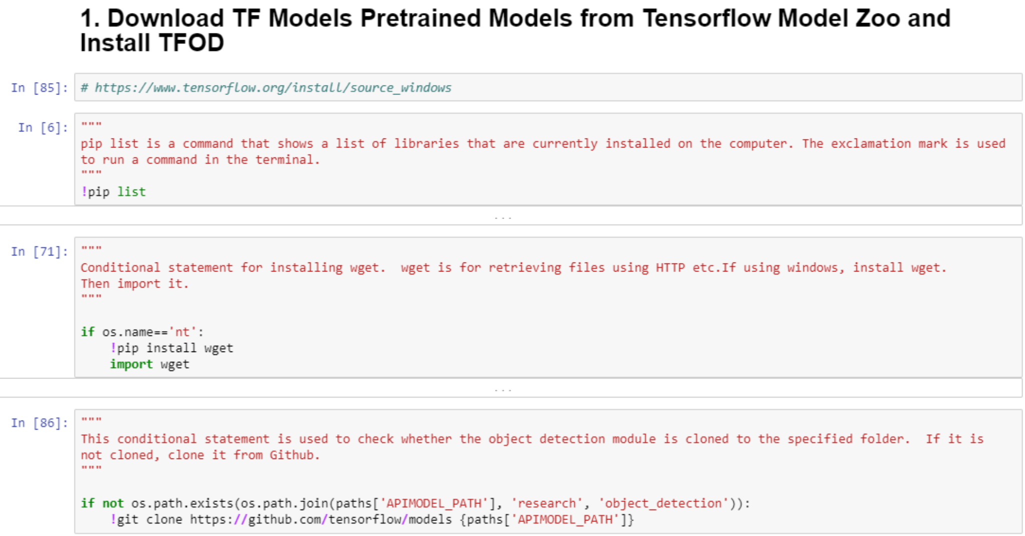
```
In [87]: """
         This files dictionary has the same purpose as the paths dictionary, except it leads to the files instead of the folders.
         """

         files = {
             'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
             'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
             'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
         }
```

```
In [88]: """
         This for loop is used to cycle through each path value inside the paths dictionary.  If the folder does not exist
         ***inside the operating system files manager***, make the directory path for the folder.  posix means mac.  nt means windows.
         """

         for path in paths.values():
             if not os.path.exists(path):
                 if os.name == 'posix':
                     !mkdir -p {path}
                 if os.name == 'nt':
                     !mkdir {path}
```

Figure 14: File directories

Figure 13 consists of several variables that are used for generating the custom model, label map and TF record files. To create these folders, the os module must first be imported. The custom model's name is 'microscope_v6'. A pretrained model is not required, but it is used in this project to speed up the training process by using datasets that already exist to enhance the new custom dataset. This saves time for the developer than starting over for each project (LeCun et al., 2015). The chosen pretrained model is the ssd_mobilenet_v2 coco model, as shown in Figure 13. The TF record file is stored in a variable, along with the label map text file. The next cell block is a dictionary of the paths for the custom model folder. This dictionary is to create a neat, readable work environment. This workspace consists of the model folders, scripts, annotation files, images, pre-trained model directory, and the TF export files to deploy to the website for prototype 3. Figure 14 shows the file directories for the pipeline configuration, TF record script and label map. If the folders do not exist, they are created using os (Renotte, 2022).

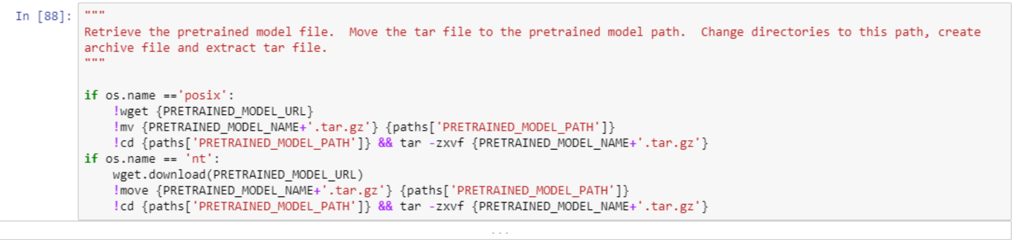
3.3.2 Pretrained model



1. Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD

```
In [85]: # https://www.tensorflow.org/install/source_windows

In [6]:  """
         pip list is a command that shows a list of libraries that are currently installed on the computer. The exclamation mark is used
         to run a command in the terminal.
         """
         !pip list

                                                        ...

In [71]: """
         Conditional statement for installing wget.  wget is for retrieving files using HTTP etc.If using windows, install wget.
         Then import it.
         """

         if os.name=='nt':
             !pip install wget
             import wget

                                                        ...

In [86]: """
         This conditional statement is used to check whether the object detection module is cloned to the specified folder.  If it is
         not cloned, clone it from Github.
         """

         if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
             !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

Figure 15: Downloading Pretrained Model and tfod api

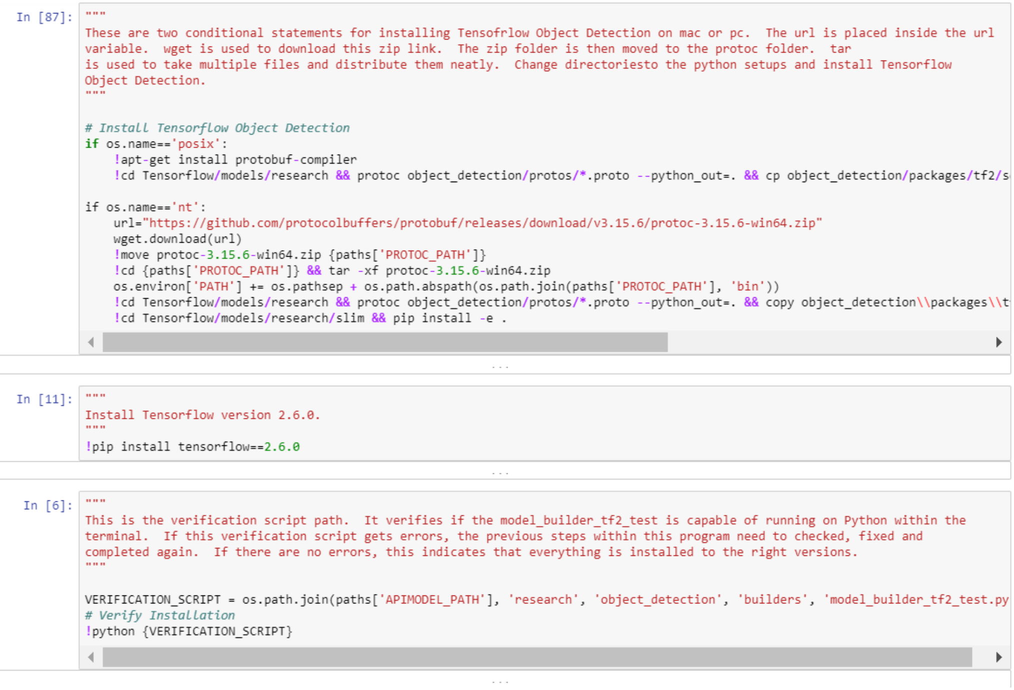


```
In [88]: """
         Retrieve the pretrained model file.  Move the tar file to the pretrained model path.  Change directories to this path, create
         archive file and extract tar file.
         """

         if os.name =='posix':
             !wget {PRETRAINED_MODEL_URL}
             !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
             !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
         if os.name == 'nt':
             wget.download(PRETRAINED_MODEL_URL)
             !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
             !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
```

Figure 16: Applying the pre-trained model.

Figure 15 and Figure 16 shows the next steps of creating the custom model. This step consists of downloading the Tensorflow pretrained models from Tensorflow Model Zoo, as well installing the Tensorflow Object Detection module. Pip list is used as a command to display a list of all packages that are installed on the computer. This gives the version of Tensorflow installed, as well as other package versions which are important to know for the model to function and be compatible with other packages. In the second code block, wget is downloaded and imported. Wget is used for downloading large files from the internet, such as HTTP, FTP. This library can be used for downloading files from the internet easier (GNU Wget. (n.d.)). In the third cell, if the object detection pathway does not exist, it is created, and the Tensorflow Object Detection module is cloned inside the api folder. In the last cell, the pre-trained model is move and applied to the pretrained-model folder (Renotte, 2022).

3.3.3 Installing Tensorflow Object Detection



```
In [87]: """
         These are two conditional statements for installing Tensofrlow Object Detection on mac or pc.  The url is placed inside the url
         variable.  wget is used to download this zip link.  The zip folder is then moved to the protoc folder.  tar
         is used to take multiple files and distribute them neatly.  Change directoriesto the python setups and install Tensorflow
         Object Detection.
         """

         # Install Tensorflow Object Detection
         if os.name=='posix':
             !apt-get install protobuf-compiler
             !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/s

         if os.name=='nt':
             url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
             wget.download(url)
             !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
             !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
             os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
             !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\t
             !cd Tensorflow/models/research/slim && pip install -e .
```

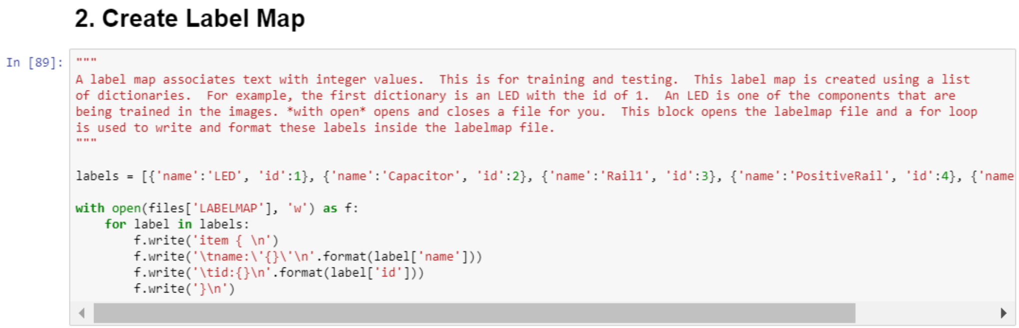
```
In [11]: """
         Install Tensorflow version 2.6.0.
         """
         !pip install tensorflow==2.6.0
```

```
In [6]: """
        This is the verification script path.  It verifies if the model_builder_tf2_test is capable of running on Python within the
        terminal.  If this verification script gets errors, the previous steps within this program need to checked, fixed and
        completed again.  If there are no errors, this indicates that everything is installed to the right versions.
        """

        VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py
        # Verify Installation
        !python {VERIFICATION_SCRIPT}
```

Figure 17: Tensorflow Object Detection and Verification Script

As shown in Figure 17, there are two conditional statements. One is for mac, and one is for pc commands. Wget is used to download the url link to Protobuf zip folder. Protobuf makes it easier for data to be transferred from one application to another (Google Developers. (n.d.)). This zip file is then moved to the protoc pathway using the move command. In the environmental variables on the operating system, a new path is added for the bin file and the Tensorflow Object Detection is applied to this path. In this prototype, Tensorflow version 2.6.0 is used for compatibility with other libraries. The verification script is an important test in the program that uses the model_builder_tf2_test file to verify that all requirements for the object detection pipeline are met and everything in the program, including packages, are functioning (TensorFlow Object Detection API. (2021)). This is where most of the errors show up and must be corrected in order to advance to the next step (Renotte, 2022).

3.3.4 Label Map

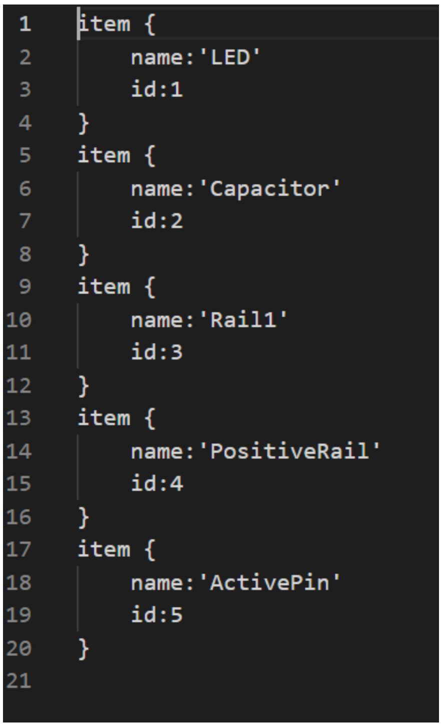


2. Create Label Map

In [89]:

```
"""
A label map associates text with integer values.  This is for training and testing.  This label map is created using a list
of dictionaries.  For example, the first dictionary is an LED with the id of 1.  An LED is one of the components that are
being trained in the images. *with open* opens and closes a file for you.  This block opens the labelmap file and a for loop
is used to write and format these labels inside the labelmap file.
"""

labels = [{'name':'LED', 'id':1}, {'name':'Capacitor', 'id':2}, {'name':'Rail1', 'id':3}, {'name':'PositiveRail', 'id':4}, {'name

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

Figure 18: Creating a Label Map

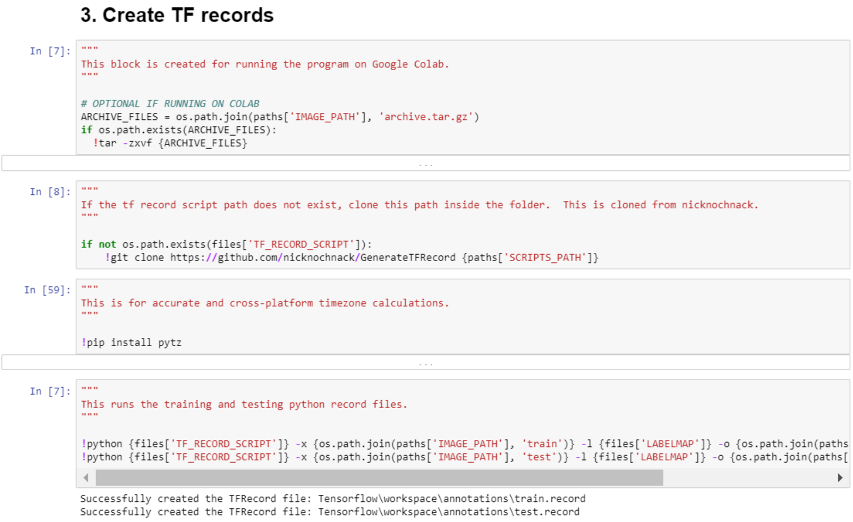


```
item {
    name:'LED'
    id:1
}
item {
    name:'Capacitor'
    id:2
}
item {
    name:'Rail1'
    id:3
}
item {
    name:'PositiveRail'
    id:4
}
item {
    name:'ActivePin'
    id:5
}
```

Figure 19: Breadboard2Schematic label map.

As shown in Figure 18, a list of dictionaries is created and stored inside a variable ‘labels’. There are five object labels that were annotated previously: an LED, Capacitor, Rail1, PositiveRail and ActivePin. These labels are stored inside this dictionary, and each label was given an integer identification (ID), i.e., LED is ID: 1. The syntax ‘with open’ is used to open and close files for the developer. In this block of code, the label map file is created, opened and the labels are written and formatted inside of this file, as shown in Figure 19. The labels are formatted inside dictionaries with the name first and then the identification. Label maps are important for machine learning models because they give consistent results for different annotation tools and datasets ((Renotte, 2022); (Russakovsky et al., 2015)).

3.3.5 TF Records



3. Create TF records

```
In [7]: """
        This block is created for running the program on Google Colab.
        """

        # OPTIONAL IF RUNNING ON COLAB
        ARCHIVE_FILES = os.path.join(paths['IMAGE_PATH'], 'archive.tar.gz')
        if os.path.exists(ARCHIVE_FILES):
          !tar -zxvf {ARCHIVE_FILES}
```

```
In [8]: """
        If the tf record script path does not exist, clone this path inside the folder.  This is cloned from nicknochnack.
        """

        if not os.path.exists(files['TF_RECORD_SCRIPT']):
            !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}
```

```
In [59]: """
         This is for accurate and cross-platform timezone calculations.
         """

         !pip install pytz
```

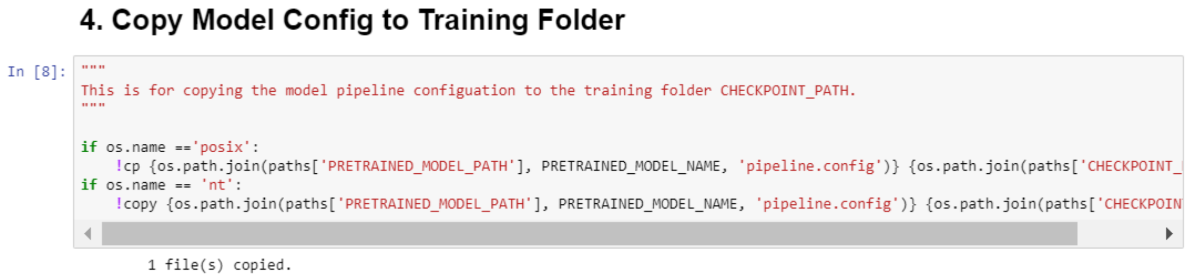
```
In [7]: """
        This runs the training and testing python record files.
        """

        !python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(paths
        !python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']} -o {os.path.join(paths[

        Successfully created the TFRecord file: Tensorflow\workspace\annotations\train.record
        Successfully created the TFRecord file: Tensorflow\workspace\annotations\test.record
```

Figure 20: Creating the TF Record Files.

Figure 20 shows the process of creating TF records for the model to use for training. TF records hold large amount of data that are used by Tensorflow for training. TF records can enhance the pre-processing for a model (Google, 2021). In Figure 20, if the developer is running the model on Google Colab (Google, 2019), the program creates an archive file for the annotated images to transfer easier. Google Colab is a cloud-based application that gives developers access to several machine learning libraries. Google Colab is useful because it offers virtual machines that allow the developer to train custom models on the machines with a graphics processing unit (GPU), instead of a central processing unit (CPU). This significantly speeds up the training of the custom model (Google. (2021)).

In the second block of code, the TF record script path is created, and the GenerateTFRecord file is cloned to the path from Github. As shown in the last cell of Figure 20, both the training tf record and testing tf record are executed in the Python compiler (Renotte, 2022).

3.3.6 Placing the Model Configuration in the Training Folder



4. Copy Model Config to Training Folder

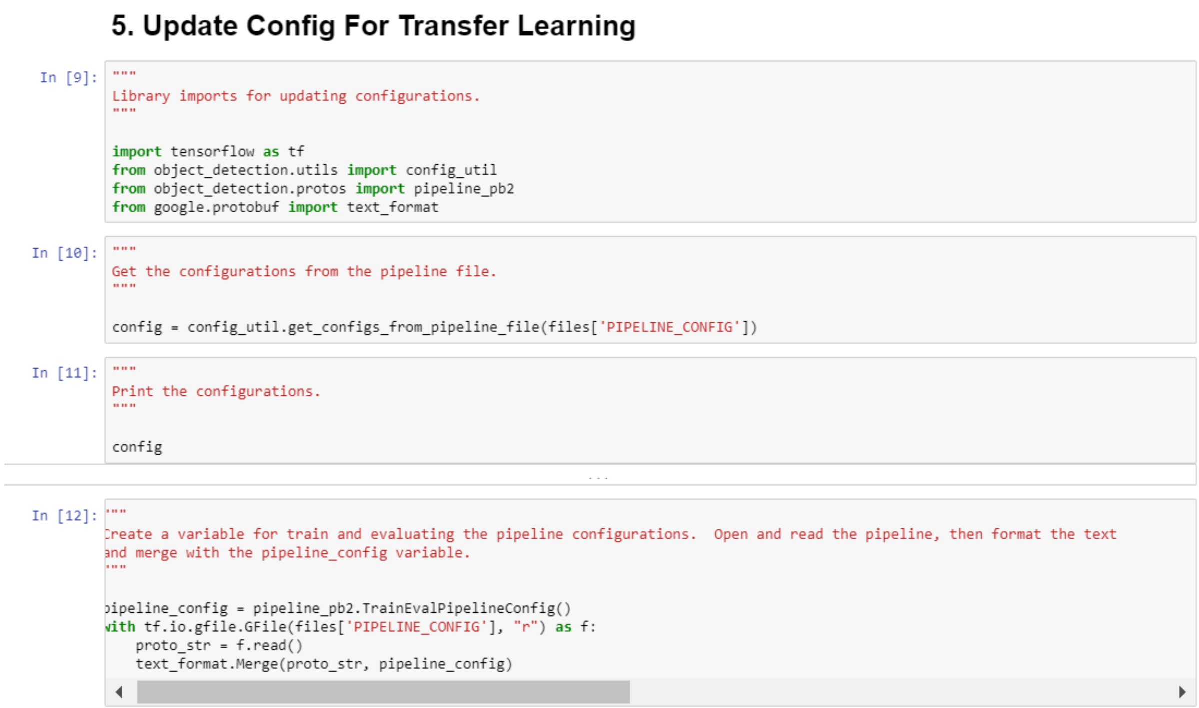
```
"""
This is for copying the model pipeline configuation to the training folder CHECKPOINT_PATH.
"""

if os.name =='posix':
    !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_
if os.name == 'nt':
    !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOIN
```

```
1 file(s) copied.
```

Figure 21: Copying the Model Configuration for Training

Figure 21 shows the model pipeline and configuration being copied to the checkpoint folder. A model configuration is a host of settings that create the architecture and behaviour of the model. A model configuration is important for the performance of a custom model (Gupta, 2020). A model pipeline is the workflow of the model created from data processing that follows a sequential pattern ((Geron, 2019); (Renotte, 2022)).

3.3.7 Updating Configuration



5. Update Config For Transfer Learning

In [9]:
```
"""
Library imports for updating configurations.
"""

import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

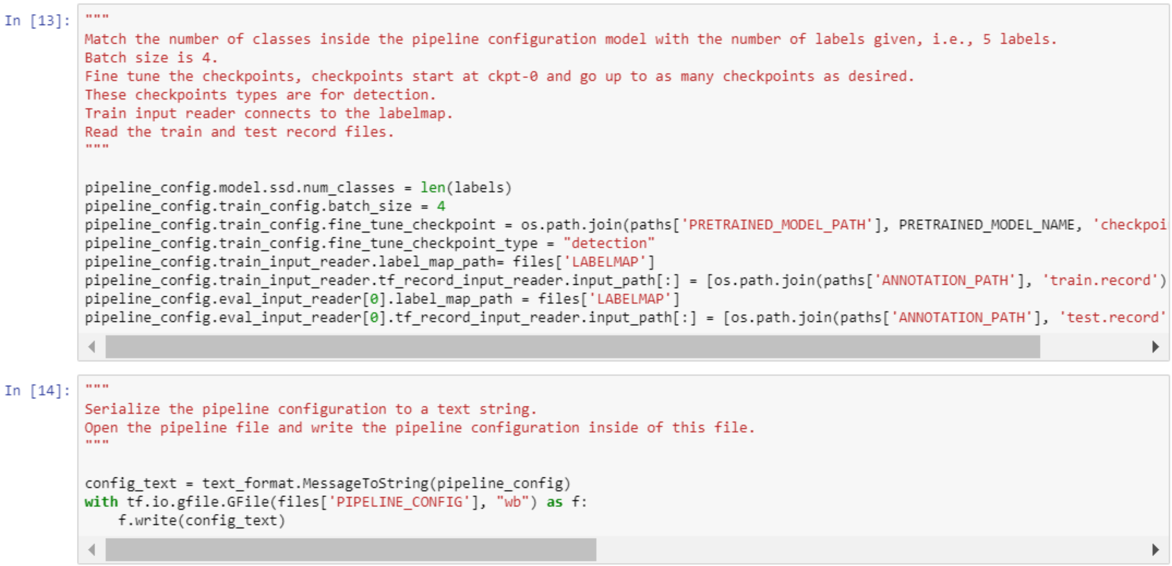
In [10]:
```
"""
Get the configurations from the pipeline file.
"""

config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

In [11]:
```
"""
Print the configurations.
"""

config
```

In [12]:
```
"""
Create a variable for train and evaluating the pipeline configurations.  Open and read the pipeline, then format the text
and merge with the pipeline_config variable.
"""

pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

Figure 22: Configuration Process.

As shown in the first code block of Figure 22, a set of libraries are imported to updating the configuration. At this stage, Tensorflow is imported for use. Object detection utilities such as the configuration utilities and pipeline utilities are imported. The text format library is also imported to format the data so that it is readable for humans (Google LLC. (n.d.)). The configurations are then retrieved from the pipeline file and can be printed to the terminal. Evaluations are create using the TrainEvalPipelineConfig function, the file is read, and the text format is changed to become readable (Renotte, 2022).



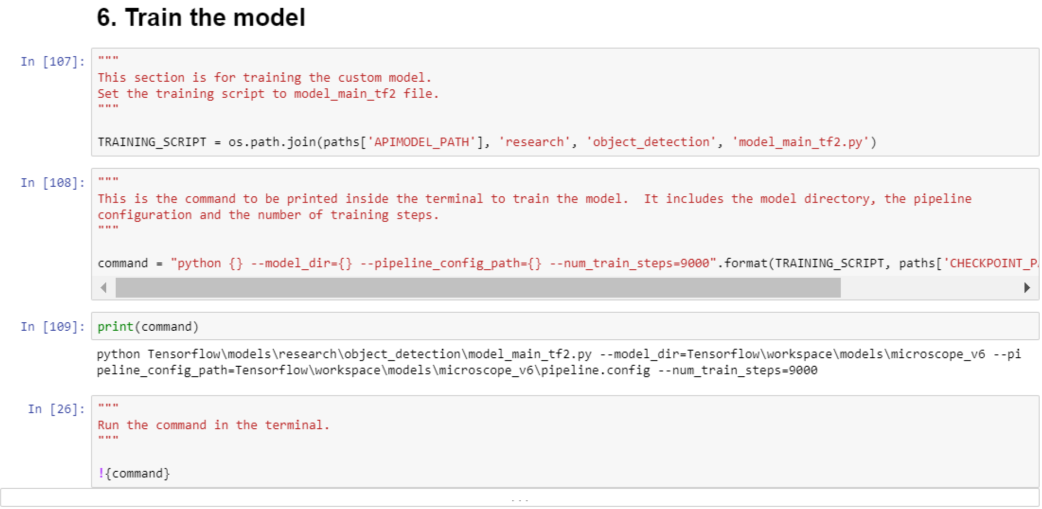
```
In [13]: """
Match the number of classes inside the pipeline configuration model with the number of labels given, i.e., 5 labels.
Batch size is 4.
Fine tune the checkpoints, checkpoints start at ckpt-0 and go up to as many checkpoints as desired.
These checkpoints types are for detection.
Train input reader connects to the labelmap.
Read the train and test record files.
"""

pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoi
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record'
```

```
In [14]: """
Serialize the pipeline configuration to a text string.
Open the pipeline file and write the pipeline configuration inside of this file.
"""

config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

Figure 23: Fitting the Data to the Pipeline Configuration

Figure 23 shows the fitting of the data in the custom model to the pipline onfiguration. There are several parameters fitted and determined. The length of the label dictionary in the label map is fitted to the pipeline. The batch size is given a value of 4. A batch size is the flow of the training of the custom model. It determines the batches of data that gets passed through to the model in particular intervals (Brownlee, J. (2017)). The next step involves determining the checkpoints of the model. The checkpoints are fitted to the pipeline as a detection type. Checkpoints can be thought of as saved points during training process of the model. Checkpoints save specific points in the training and developers can load back to these checkpoints if necessary (Brownlee, J. (2021)). The train and test records are also fitted to the pipeline configuration. The pipeline configuration is then made readable with the text format and is pasted inside the pipeline configuration folder (Renotte, 2022).

3.3.8 Model Training

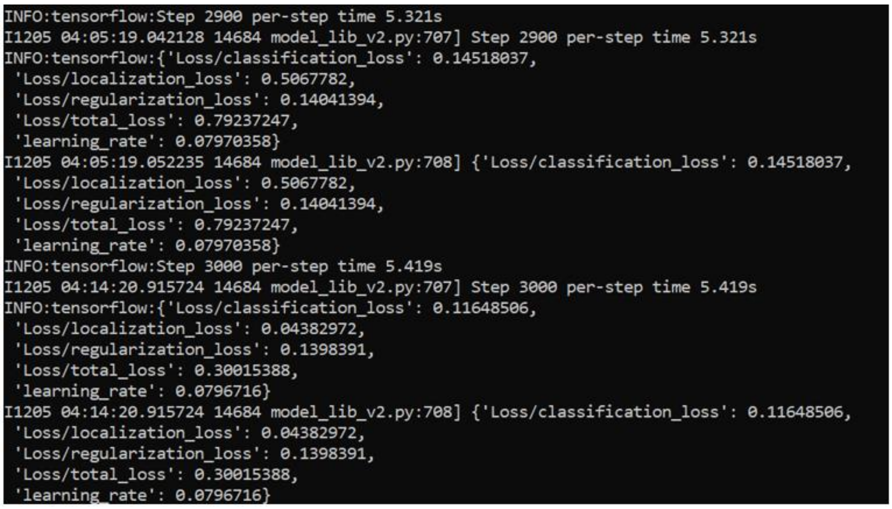


6. Train the model

```
In [107]: """
          This section is for training the custom model.
          Set the training script to model_main_tf2 file.
          """

          TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'model_main_tf2.py')

In [108]: """
          This is the command to be printed inside the terminal to train the model.  It includes the model directory, the pipeline
          configuration and the number of training steps.
          """

          command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps=9000".format(TRAINING_SCRIPT, paths['CHECKPOINT_P

In [109]: print(command)

          python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\microscope_v6 --pi
          peline_config_path=Tensorflow\workspace\models\microscope_v6\pipeline.config --num_train_steps=9000

In [26]:  """
          Run the command in the terminal.
          """

          !{command}
```

Figure 24: Preparing the Custom Model for Training

Figure 24 shows the training commands for beginning the training process of the custom model. The first line of code attached the model main tf2 file to the training script variable. This is used in the command to initialize the training for the model. The command is executed in the Python interpreter and requires the model directory, the pipeline configuration directory and the number of training steps that the model uses to determine the length of training the model takes to complete. The training steps are the number of updates the model takes during training (Goodfellow, I., Bengio, Y., & Courville, A. (2016)). This command gets executed in the terminal and the training begins (Renotte, 2022).

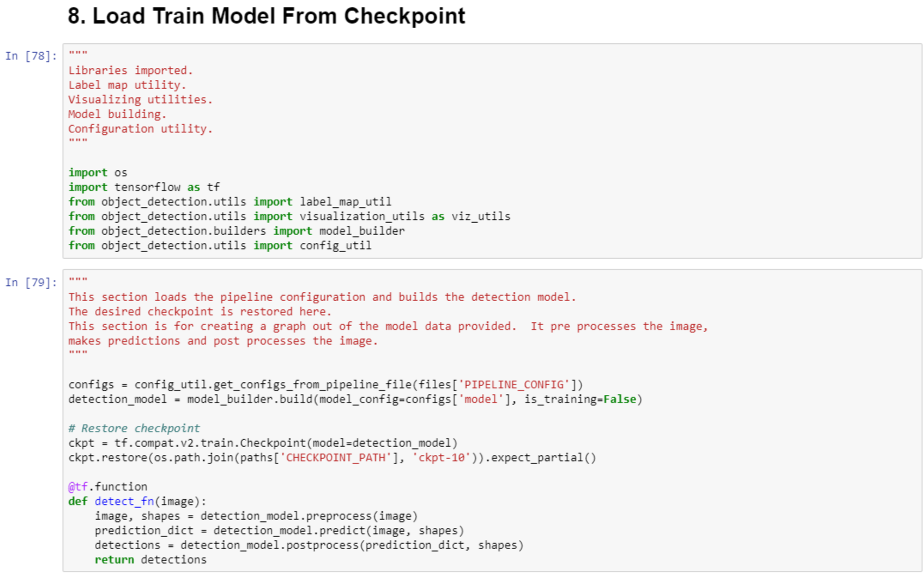


```
INFO:tensorflow:Step 2900 per-step time 5.321s
I1205 04:05:19.042128 14684 model_lib_v2.py:707] Step 2900 per-step time 5.321s
INFO:tensorflow:{'Loss/classification_loss': 0.14518037,
 'Loss/localization_loss': 0.5067782,
 'Loss/regularization_loss': 0.14041394,
 'Loss/total_loss': 0.79237247,
 'learning_rate': 0.07970358}
I1205 04:05:19.052235 14684 model_lib_v2.py:708] {'Loss/classification_loss': 0.14518037,
 'Loss/localization_loss': 0.5067782,
 'Loss/regularization_loss': 0.14041394,
 'Loss/total_loss': 0.79237247,
 'learning_rate': 0.07970358}
INFO:tensorflow:Step 3000 per-step time 5.419s
I1205 04:14:20.915724 14684 model_lib_v2.py:707] Step 3000 per-step time 5.419s
INFO:tensorflow:{'Loss/classification_loss': 0.11648506,
 'Loss/localization_loss': 0.04382972,
 'Loss/regularization_loss': 0.1398391,
 'Loss/total_loss': 0.30015388,
 'learning_rate': 0.0796716}
I1205 04:14:20.915724 14684 model_lib_v2.py:708] {'Loss/classification_loss': 0.11648506,
 'Loss/localization_loss': 0.04382972,
 'Loss/regularization_loss': 0.1398391,
 'Loss/total_loss': 0.30015388,
 'learning_rate': 0.0796716}
```

Figure 25: Training in the Command Prompt

Figure 25 shows an example of training in the command prompt. The training began when the command was entered. The training step increases in intervals of 100 training steps per batch. This is to cause less error in the training by slowly feeding it the data. A low loss means an effective detection. The training stops after reaching the specified training steps and after this training process finishes, detections can be made on the images (Renotte, 2022).

3.3.9 Checkpoints



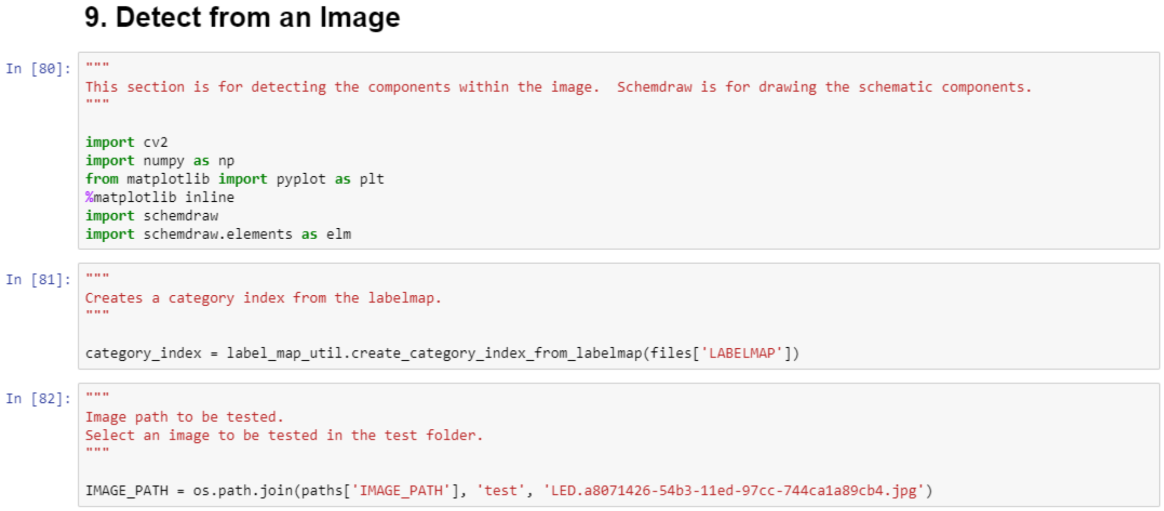
8. Load Train Model From Checkpoint

```
In [78]: """
         Libraries imported.
         Label map utility.
         Visualizing utilities.
         Model building.
         Configuration utility.
         """

         import os
         import tensorflow as tf
         from object_detection.utils import label_map_util
         from object_detection.utils import visualization_utils as viz_utils
         from object_detection.builders import model_builder
         from object_detection.utils import config_util

In [79]: """
         This section loads the pipeline configuration and builds the detection model.
         The desired checkpoint is restored here.
         This section is for creating a graph out of the model data provided.  It pre processes the image,
         makes predictions and post processes the image.
         """

         configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
         detection_model = model_builder.build(model_config=configs['model'], is_training=False)

         # Restore checkpoint
         ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
         ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-10')).expect_partial()

         @tf.function
         def detect_fn(image):
             image, shapes = detection_model.preprocess(image)
             prediction_dict = detection_model.predict(image, shapes)
             detections = detection_model.postprocess(prediction_dict, shapes)
             return detections
```

Figure 26: Loading the Custom Model

In Figure 26, several packages are imported from the object detection utilities library. This is used to get the configurations from the completed pipeline file. The model is then defined and the checkpoints during the training are restored and set to the desired checkpoint. The ‘detect_fn’ function pre-processes the image, the model predictions are made, and the detections are made. This function is used for the final detections of the model (Renotte, 2022).

3.3.10 Image Detections and Model Results



9. Detect from an Image

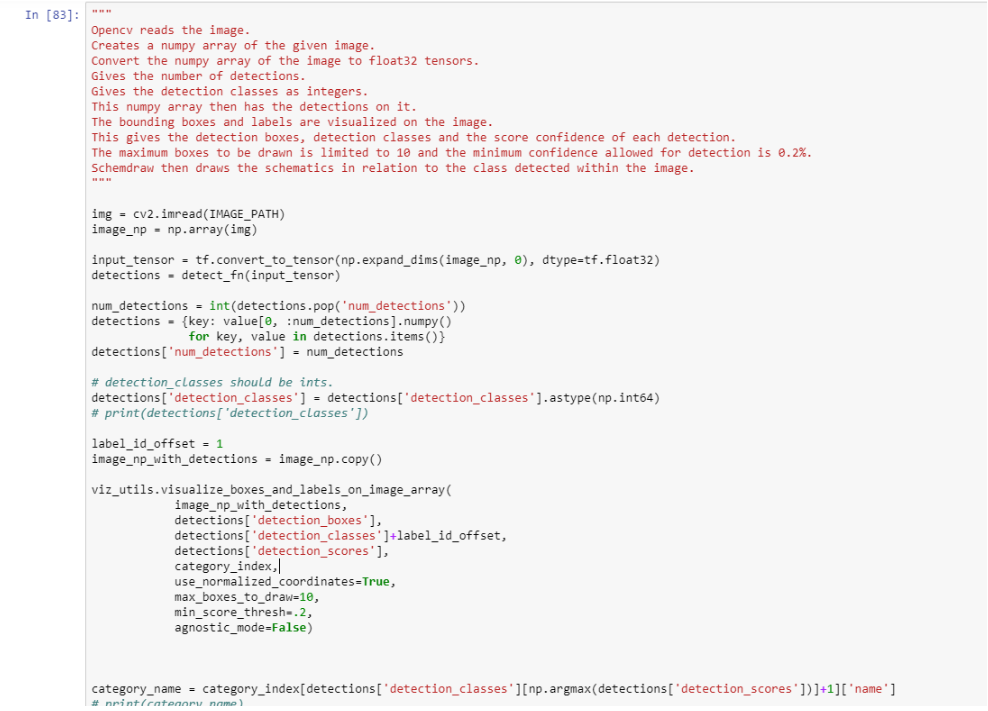
```
In [80]: """
         This section is for detecting the components within the image.  Schemdraw is for drawing the schematic components.
         """

         import cv2
         import numpy as np
         from matplotlib import pyplot as plt
         %matplotlib inline
         import schemdraw
         import schemdraw.elements as elm
```

```
In [81]: """
         Creates a category index from the labelmap.
         """

         category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
```

```
In [82]: """
         Image path to be tested.
         Select an image to be tested in the test folder.
         """

         IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'LED.a8071426-54b3-11ed-97cc-744ca1a89cb4.jpg')
```

Figure 27: Image Detections

The first cell in Figure 27 is used for importing the libraries for the post processing of prototype 2. OpenCV is required for the post processing of the image, and matplotlib (Matplotlib, 2012) is responsible to the visualization of this data. Numpy (Numpy, 2009) is needed to create a Numpy array of the processed image. Schemdraw (Schemdraw Documentation — Schemdraw 0.16 Documentation, n.d.) was used to draw a schematic diagram of the result in the image. A category index was created from the label map, which categorizes each object in the detection. The image path variable decides the image that the developer wants to test for the result (Renotte, 2022).

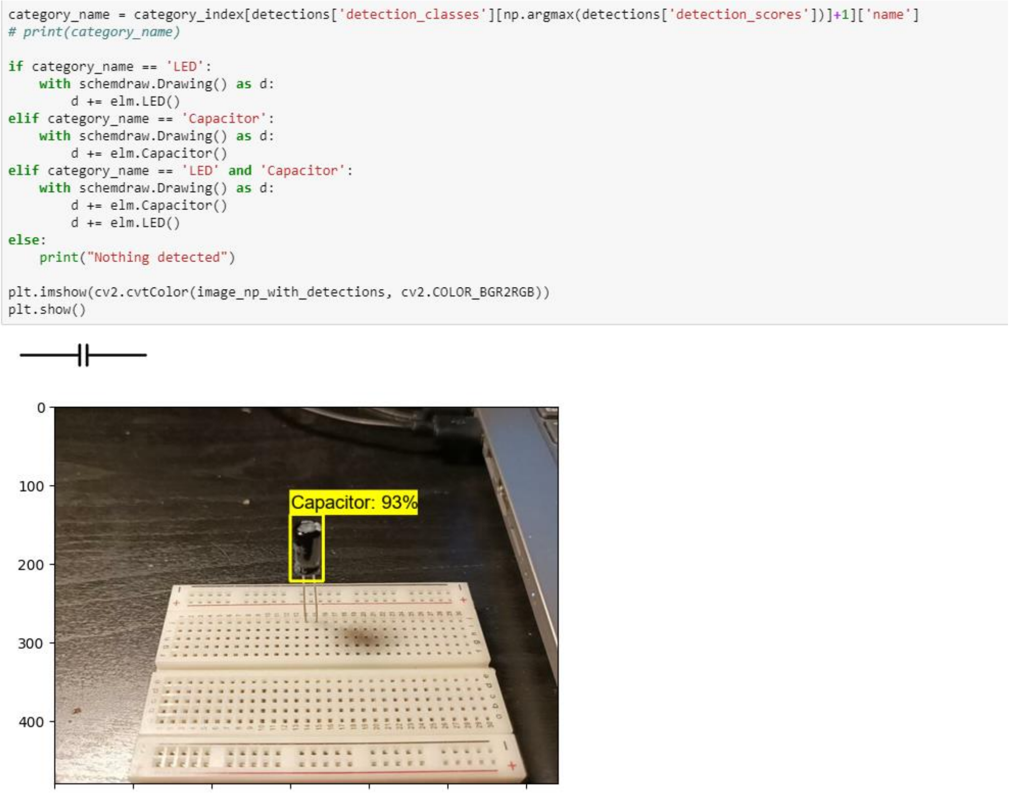


```
In [83]: """
         Opencv reads the image.
         Creates a numpy array of the given image.
         Convert the numpy array of the image to float32 tensors.
         Gives the number of detections.
         Gives the detection classes as integers.
         This numpy array then has the detections on it.
         The bounding boxes and labels are visualized on the image.
         This gives the detection boxes, detection classes and the score confidence of each detection.
         The maximum boxes to be drawn is limited to 10 and the minimum confidence allowed for detection is 0.2%.
         Schemdraw then draws the schematics in relation to the class detected within the image.
         """

         img = cv2.imread(IMAGE_PATH)
         image_np = np.array(img)

         input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
         detections = detect_fn(input_tensor)

         num_detections = int(detections.pop('num_detections'))
         detections = {key: value[0, :num_detections].numpy()
                       for key, value in detections.items()}
         detections['num_detections'] = num_detections

         # detection_classes should be ints.
         detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
         # print(detections['detection_classes'])

         label_id_offset = 1
         image_np_with_detections = image_np.copy()

         viz_utils.visualize_boxes_and_labels_on_image_array(
                     image_np_with_detections,
                     detections['detection_boxes'],
                     detections['detection_classes']+label_id_offset,
                     detections['detection_scores'],
                     category_index,
                     use_normalized_coordinates=True,
                     max_boxes_to_draw=10,
                     min_score_thresh=.2,
                     agnostic_mode=False)



         category_name = category_index[detections['detection_classes'][np.argmax(detections['detection_scores'])]+1]['name']
         # print(category_name)
```

Figure 28: Prototype 2 Final Detections

Figure 28 shows the post processing of the testing image that was previously selected. The image is opened using OpenCV and a numpy array copied version of the image is made, which is the image the post-processing takes place. The numpy image is converted to tensors and the detect_fn function is used to do post-processing. The number of detections is added to an array. The total detection in the image is determined in the for loop. The detection classes are recorded as integers. Finally, the bounding boxes, detection boxes, detection classes and detection scores are visualized on the image. There is also a selection of parameters: the maximum number of boxes to display on the image is limited to ten and the minimum score detection threshold is set to 0.2% confidence (Renotte, 2022).

3.3.11 Prototype 2 Results and Schematic Outcome



```
category_name = category_index[detections['detection_classes'][np.argmax(detections['detection_scores'])]+1]['name']
# print(category_name)

if category_name == 'LED':
    with schemdraw.Drawing() as d:
        d += elm.LED()
elif category_name == 'Capacitor':
    with schemdraw.Drawing() as d:
        d += elm.Capacitor()
elif category_name == 'LED' and 'Capacitor':
    with schemdraw.Drawing() as d:
        d += elm.Capacitor()
        d += elm.LED()
else:
    print("Nothing detected")

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

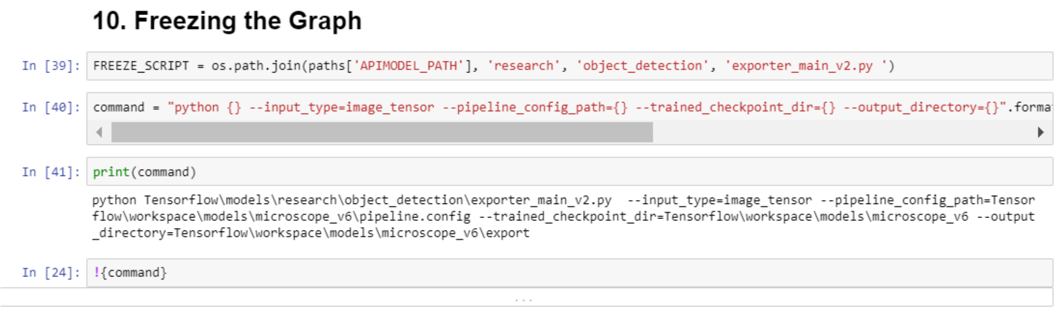
Figure 29: Results of Prototype 2.

As can be seen in Figure 29, the results of prototype 2 can be seen and is visualized on a Matplotlib graph. The confidence detection for a capacitor is high, and the detections are accurate at 93% confidence rate. The schematic symbol for capacitor is drawn above the post-processed image. To draw this schematic symbol, the category name detected in the image was placed inside a variable ‘category_name’. If the category name detected is LED, it draws an LED schematic symbol. If the category name is capacitor, it draws a Capacitor. These schematics are drawn using Schemdraw (Renotte, 2022).

3.4 Prototype 3

The aim of prototype 3 was to freeze the output graph from prototype 2, convert the model to Tensorflow JavaScript and post the model to the public using IBM Cloud to make it deployable and usable on a JavaScript website.

3.4.1 Graph Freezing



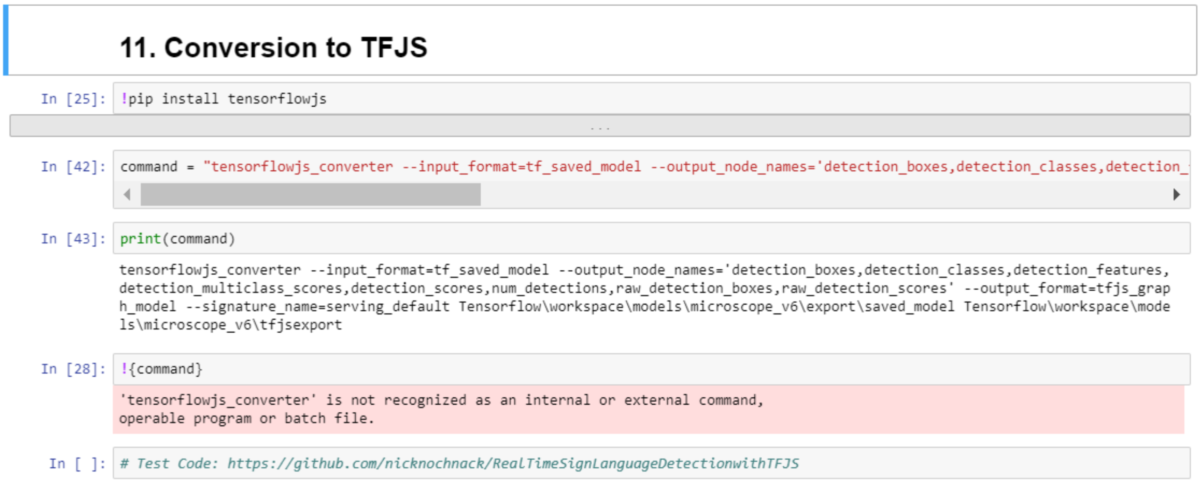
10. Freezing the Graph

```
In [39]: FREEZE_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'exporter_main_v2.py ')

In [40]: command = "python {} --input_type=image_tensor --pipeline_config_path={} --trained_checkpoint_dir={} --output_directory={}".forma

In [41]: print(command)

         python Tensorflow\models\research\object_detection\exporter_main_v2.py  --input_type=image_tensor --pipeline_config_path=Tensor
         flow\workspace\models\microscope_v6\pipeline.config --trained_checkpoint_dir=Tensorflow\workspace\models\microscope_v6 --output
         _directory=Tensorflow\workspace\models\microscope_v6\export

In [24]: !{command}
```

Figure 30: Freezing the Graph

The code snippet in Figure 30 uses the command prompt to freeze the graph of the model. Freezing a graph is useful for optimizing and deploying a custom model. Freezing the graph converts the variables inside the model into constants, which removes the problem of maintaining variables in inference, reducing the size of the custom model. This makes it easy to use for websites and better compatibility with other applications such as frameworks (T. Chen, "Introduction to TensorFlow for Deep Learning - Freezing a Model for Deployment," in Medium, (Nov 14, 2017)). In Figure 30, a FREEZE_SCRIPT constant variable is used to store a pathway to the ‘exporter_main_v2’ file. The Python interpreter executes the command with a parameter for the input type, which is an image tensor, the pipelines configuration path, the checkpoint directory and the output directory (Renotte, 2022).

3.4.2 Converting the Custom Model to Tensorflow JavaScript File



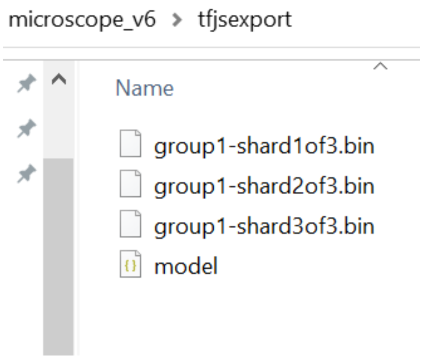
11. Conversion to TFJS

```
In [25]: !pip install tensorflowjs

In [42]: command = "tensorflowjs_converter --input_format=tf_saved_model --output_node_names='detection_boxes,detection_classes,detection_

In [43]: print(command)

         tensorflowjs_converter --input_format=tf_saved_model --output_node_names='detection_boxes,detection_classes,detection_features,
         detection_multiclass_scores,detection_scores,num_detections,raw_detection_boxes,raw_detection_scores' --output_format=tfjs_grap
         h_model --signature_name=serving_default Tensorflow\workspace\models\microscope_v6\export\saved_model Tensorflow\workspace\mode
         ls\microscope_v6\tfjsexport

In [28]: !{command}

         'tensorflowjs_converter' is not recognized as an internal or external command,
         operable program or batch file.

In [ ]:  # Test Code: https://github.com/nicknochnack/RealTimeSignLanguageDetectionwithTFJS
```

Figure 31: TFJS Conversion

Figure 31 shows the code snippet for converting the custom model to a Tensorflow JavaScript (TensorFlow.js, 2019) (TFJS) file. The first command parameter for converting the model requires the saved Tensorflow custom model. The output node names of the model are a list of detection output parameters for the website. Detection boxes are for drawing the bounding box. Detection classes are for identifying the label of the object. Detection features are used for recording the features of the object. Detection multiclass score is for recording the confidence score of the detection. The number of detections outlines the number of objects detected in the image. These were all used for debugging purposes of the terminal for the final website. After executing this command, the TFJS file was saved inside the custom model folder (Renotte, 2022).

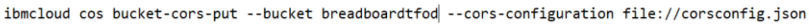
3.4.3 Posting the Custom Model to IBM Cloud


microscope_v6 > tfjsexport

Name

group1-shard1of3.bin
group1-shard2of3.bin
group1-shard3of3.bin
model

Figure 32: TFJS Export Files



```
ibmcloud cos bucket-cors-put --bucket breadboardtfod --cors-configuration file://corsconfig.json
```

Figure 33: JSON File

The model was posted on IBM Cloud storage for the model to run on the JavaScript website. As can be seen in Figure 32, the ‘model.json’ file and all three bin files within the TFJS export folder are stored into a ‘bucket’ on IBM Cloud (IBM Cloud, 2019) and posted publicly (Renotte, 2023). After this, the Cloud Object Store Plugin (IBM Cloud Docs, n.d.) and the command in Figure 33 was executed. This command allows for the model to be shared and used by the public (Renotte, 2023).

3.5 Conclusion

This chapter described the full process of the three prototypes that were constructed and tested before the final project development. The first prototype uses Jupyter Notebook to create a program that collects and labels images for the training and testing of the custom model. The second prototype uses another program that focuses on training and testing the custom model. In this procedure, new folders are created and moved. Git repositories are cloned into directories. A pre-trained model and object detection is downloaded from the Tensorflow website. A label map is created. TF records are created. Model configurations are set before training and testing. The model is then trained, loaded and tested. This gave the result of an application that detects electronic components within an image and then draws the electrical schematic symbol of that component above the image. The third prototype then freezes the graph, exports the model into a TFJS file and uses a public IBM Cloud bucket link to the model.json file for websites to have access to it.

4 The Final Project Development

4.1 Introduction

The main objective of this chapter is to prove the research of the final project development and display the final project design. The website purpose is discussed, which explains the approach and desired theme for the design of the website. The design of the website involved three main steps, a hyper-text markup language (HTML) document file, cascading style sheet (CSS) file and JavaScript file. These three components joined together form a website. The HTML document is the main structure and content of the website. The CSS file accounts for the styling of the HTML file. Finally, the JavaScript file is used to create interactions on the website, such as buttons that perform a task. Once the model is deployed on the website, the user can make detections on the image.

4.2 Website Purpose

The purpose of this website is to create a simple, user-friendly and approachable way for people to convert electrical circuitry into schematic diagrams. Machine learning websites commonly have two buttons, one for choosing the image file within the file manager of a computer. This opens the file manager of the computer and allows the user to select an image. The second button is commonly used to predict the inputted image and make detections from these predictions. This second button involves the post-processing of the image.

4.3 Website Design

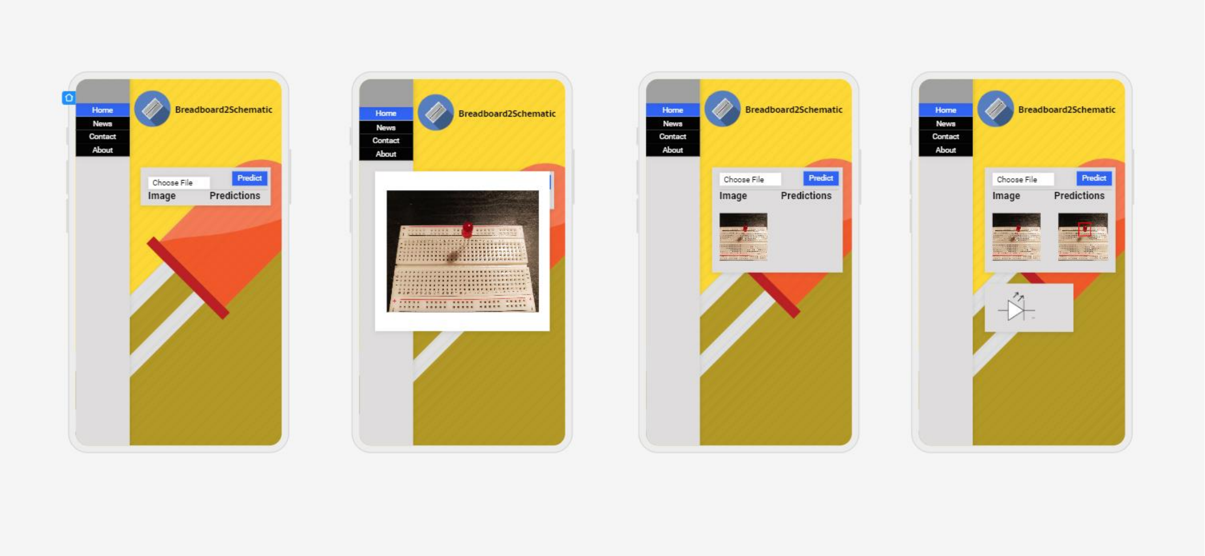


Figure 34: Website Design Ideas

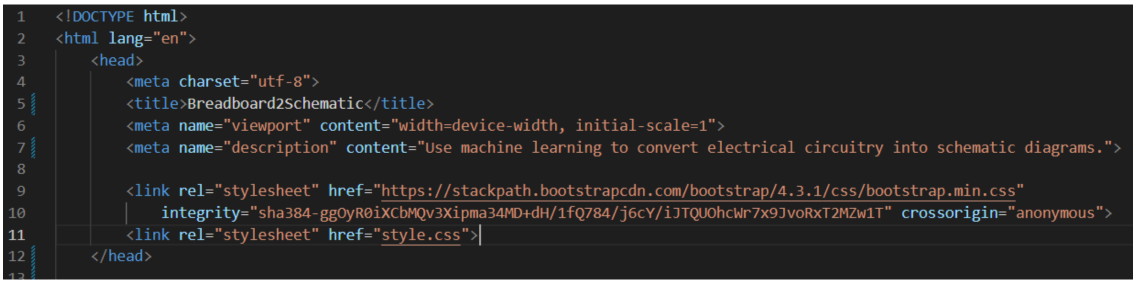
Figure 34 shows the website design layout and process, which was designed using Uizard. Uizard is a tool that can help developers visualize and design websites using wireframe diagrams and these designs can then be applied to real websites (Uizard, n.d.). On the left-hand side of the website is a navigation menu that the user can use to navigate between different pages. The top of the website has the logo and project name. In the centre of the website is where the user can make predictions with an image that is inputted through a button. There are two buttons, a ‘choose file’ button and ‘predict’ button. Figure 34 shows that the user can press the choose file button to choose a picture file to input. The image is then placed on the website when the user confirms an image choice. The user then presses the predict button to make predictions and detections on the image and this is where the post processing takes place. The result, as shown in the last wireframe in Figure 34, is a bounding box surrounding the component and the website draws the symbol for the schematics detected.

4.4 Website Architecture

In this section, the architecture of the Breadboard2Schematic is discussed, which is a machine learning website for converting electrical circuitry to schematic diagrams. The architecture follows the three main responsible files that make up a website: a HTML document, CSS file and JavaScript file.

4.4.1 HTML Document

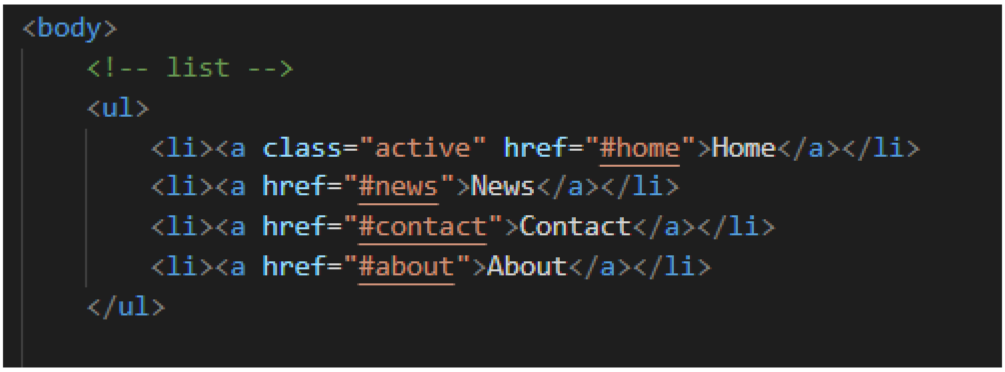
A hypertext markup language (HTML) document is a file that uses markup tags that are known as elements. These elements hold the main structure of the website, including images, links and the general writing of the website. A typical HTML document has a head element and body element. The body element is where the main text and content is placed, and the head element is where the website information is placed (W3C. (2021)). This website was created using Visual Studio Code. Visual Studio Code is a code editor environment for developers to program and debug (Microsoft. (2021)).



```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Breadboard2Schematic</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta name="description" content="Use machine learning to convert electrical circuitry into schematic diagrams.">

        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
            integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
        <link rel="stylesheet" href="style.css">
    </head>
```

Figure 35: HTML Document Head

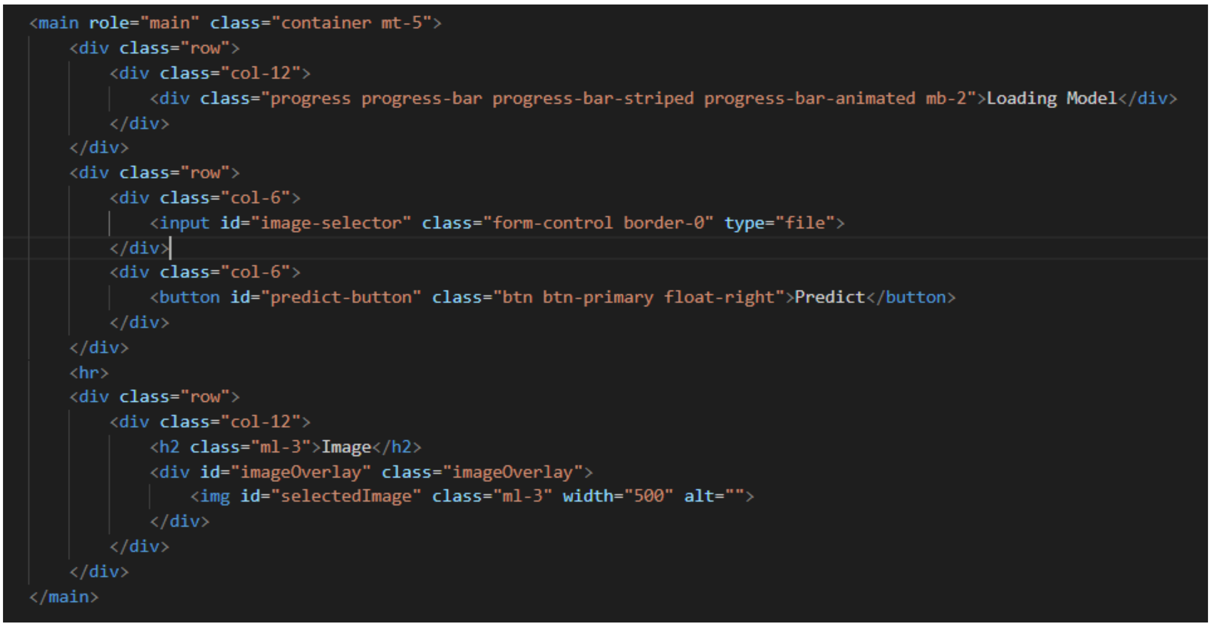
Figure 35 shows the head element structure of the HTML document. The first element (tag) in the document informs the code editor that this document type is a HTML file and the chosen language for the file is English. The title of the website is Breadboard2Schematic, which is placed in the title element. A device-width viewport is used to scale the website to fit the screen of the website on the device the user is on. A viewport decides the size of the website and how the website fits the screen for different devices such as computer or mobile phone (Mozilla. (n.d.)). The description meta element is the summary of the website for the user to read before they click on the website (W3Schools. (n.d.)). There are two stylesheet link attachments in the document. The first stylesheet is a bootstrap stylesheet. The bootstrap stylesheet has pre-built cascading style sheet (CSS) styles for the web page (Bootstrap. (n.d.)). This makes it easier for developers and saves time. The second stylesheet links to the custom CSS file (Dittmann, 2023).



```html
<body>
    <!-- list -->
    <ul>
        <li><a class="active" href="#home">Home</a></li>
        <li><a href="#news">News</a></li>
        <li><a href="#contact">Contact</a></li>
        <li><a href="#about">About</a></li>
    </ul>
```

Figure 36: HTML Document Navigation Bar

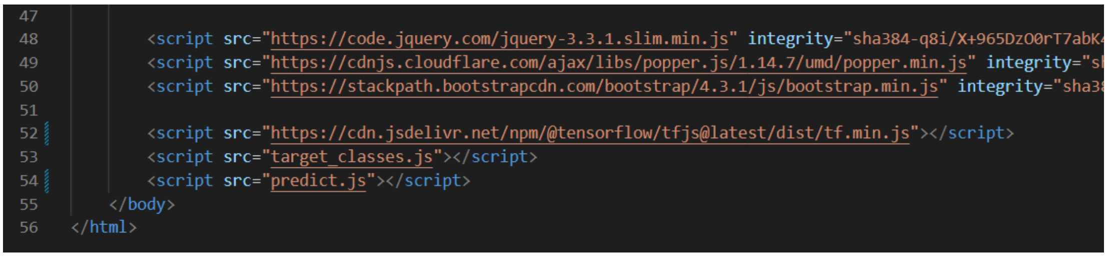
As shown in Figure 36, the body element is used to structure the main writing of the website. Inside the body tag is a list tag. This list creates a navigation bar on the website, with titles ‘home’, ‘news’, ‘contact’ and ‘about’ (Dittmann, 2023).



```
<main role="main" class="container mt-5">
    <div class="row">
        <div class="col-12">
            <div class="progress progress-bar progress-bar-striped progress-bar-animated mb-2">Loading Model</div>
        </div>
    </div>
    <div class="row">
        <div class="col-6">
            <input id="image-selector" class="form-control border-0" type="file">
        </div>
        <div class="col-6">
            <button id="predict-button" class="btn btn-primary float-right">Predict</button>
        </div>
    </div>
    <hr>
    <div class="row">
        <div class="col-12">
            <h2 class="ml-3">Image</h2>
            <div id="imageOverlay" class="imageOverlay">
                <img id="selectedImage" class="ml-3" width="500" alt="">
            </div>
        </div>
    </div>
</main>
```

Figure 37: HTML Document Main

The main tag in Figure 37 means the main content of the web page (Mozilla. (n.d.)). Figure 37 also shows a series of divided sections, ‘class’ and ‘id’ attributes. These attributes are used to name and categorize the sections of the web page. This is done to reference them in the custom stylesheet. The custom stylesheet can then refer to these attribute names and style them. Classes are for grouping elements. This is useful if the developer wants to group elements that are like one another, so that the elements can share the same styling and design. The id attribute is the opposite. It is for individual elements that do not share styles with any other element on the web page. To structure the web page, each content, such as a button, is placed inside a row or column. These row and column dividers can be referred to in the stylesheet and divide the web page into rows and columns by determining their position. An animated loading bar appears at the top of the page to show the user the progress of the custom model loading into the website. There are two buttons placed at the centre of the web page, inside the button tag. One button is for image selection and the other button is for image post-processing. A secondary heading ‘image’ is at the centre of the web page to inform the user where the image is inserted after selection. An image is then placed in the image overlay underneath the title (Dittmann, 2023).

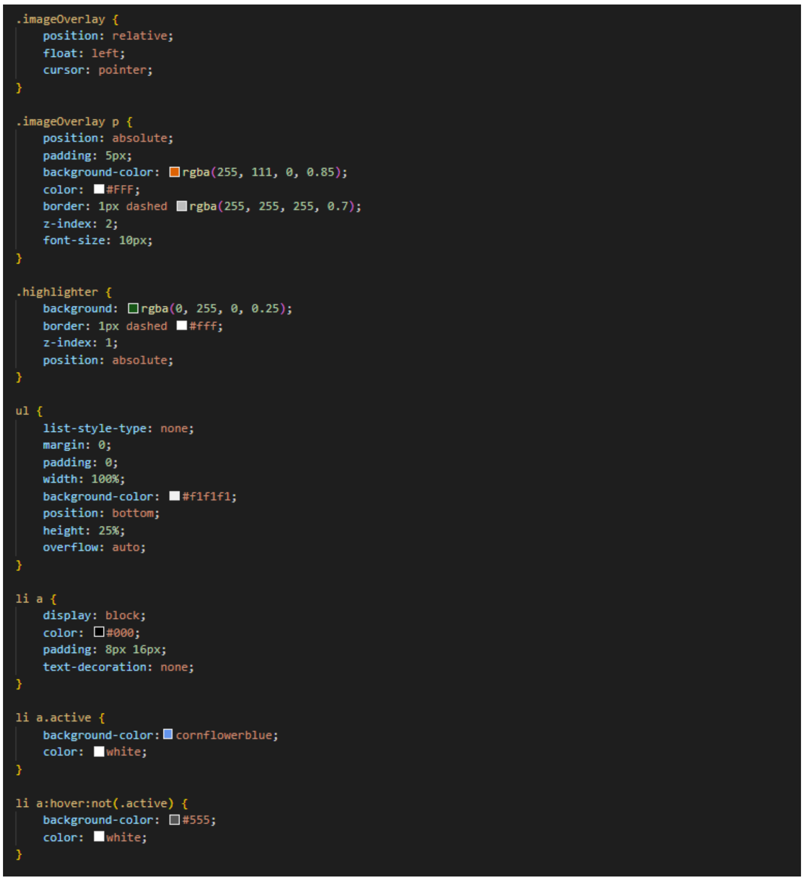


```html

        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="s
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha3

        <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
        <script src="target_classes.js"></script>
        <script src="predict.js"></script>
    </body>
</html>
```

Figure 38: HTML Document Scripts

There are six scripts included at the end of the body tag in Figure 38. The first script included is Jquery. Jquery is a JavaScript library that developers can use to make interactive web design easier. Jquery can help make animations for the website and handle user interactions with buttons and events such as HTML document events (Freeman, A., & Robson, B. (2011)). Cdnjs is a script that allows for faster loading times for the user and overall better performance. Cdnjs also provides a collection of popular libraries for the developer to use, such as Jquery (Cloudflare, Inc. (n.d.)). Stackpath bootstrap is the third imported script. Stackpath bootstrap allows for easier interactions for the website because it provides pre-built objects, such as pre-built buttons and navigation menus (StackPath, LLC. (n.d.)). Cdn Jsdelivr offers similar technologies to Cdnjs, however, Jsdelivr offers HTTPS for custom domains ((CDNJS. (n.d.)); ProspectOne. (n.d.)). The final two script imports are custom script files. The predict JavaScript file includes all the backend code, such as the button interactions and model deployment. The target classes JavaScript file includes a dictionary label map of the objects the web site detects (Dittmann, 2023).

4.4.2 CSS File

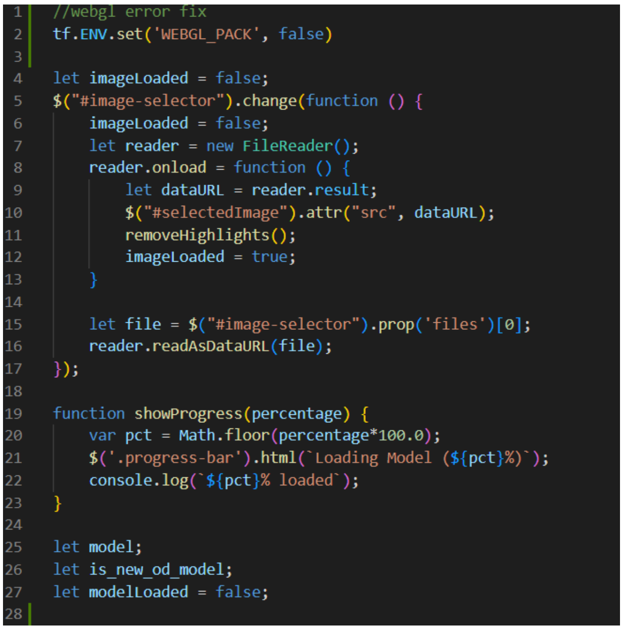


```css
.imageOverlay {
    position: relative;
    float: left;
    cursor: pointer;
}

.imageOverlay p {
    position: absolute;
    padding: 5px;
    background-color: rgba(255, 111, 0, 0.85);
    color: #FFF;
    border: 1px dashed rgba(255, 255, 255, 0.7);
    z-index: 2;
    font-size: 10px;
}

.highlighter {
    background: rgba(0, 255, 0, 0.25);
    border: 1px dashed #fff;
    z-index: 1;
    position: absolute;
}

ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 100%;
    background-color: #f1f1f1;
    position: bottom;
    height: 25%;
    overflow: auto;
}

li a {
    display: block;
    color: #000;
    padding: 8px 16px;
    text-decoration: none;
}

li a.active {
    background-color: cornflowerblue;
    color: white;
}

li a:hover:not(.active) {
    background-color: #555;
    color: white;
}
```

Figure 39: Document Styling.

Figure 39 shows the cascading style sheet for the website. Cascading style sheets are files that developers can use to adjust the style and design of a web page. In this file, elements, classes and ids are separated and can be given specific styles (World Wide Web Consortium. (2018)). In Figure 39, the ‘image overlay’, ‘image overlay p’ and ‘highlighter’ are used to style the bounding box on the image and is referred to in the JavaScript file. The box has an orange outline with a green highlighter border. The ‘ul’, ‘li a’, ‘li a.active’ and ‘li a:hover:not(.active)’ refers to the navigation menu list and the style changes when the user hovers the cursor over the boxes in the menu. The theme of the website design is made for simplicity (Dittmann, 2023).

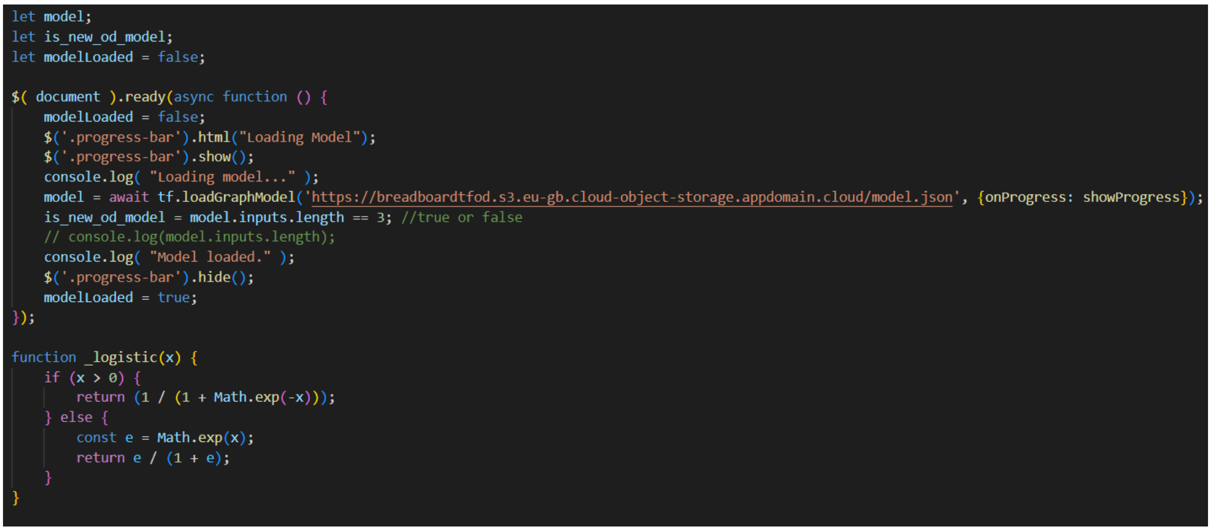
4.4.3 JavaScript Files



```javascript
//webgl error fix
tf.ENV.set('WEBGL_PACK', false)

let imageLoaded = false;
$("#image-selector").change(function () {
    imageLoaded = false;
    let reader = new FileReader();
    reader.onload = function () {
        let dataURL = reader.result;
        $("#selectedImage").attr("src", dataURL);
        removeHighlights();
        imageLoaded = true;
    }

    let file = $("#image-selector").prop('files')[0];
    reader.readAsDataURL(file);
});

function showProgress(percentage) {
    var pct = Math.floor(percentage*100.0);
    $('.progress-bar').html(`Loading Model (${pct}%)`);
    console.log(`${pct}% loaded`);
}

let model;
let is_new_od_model;
let modelLoaded = false;

```

Figure 40: Loading the Image.

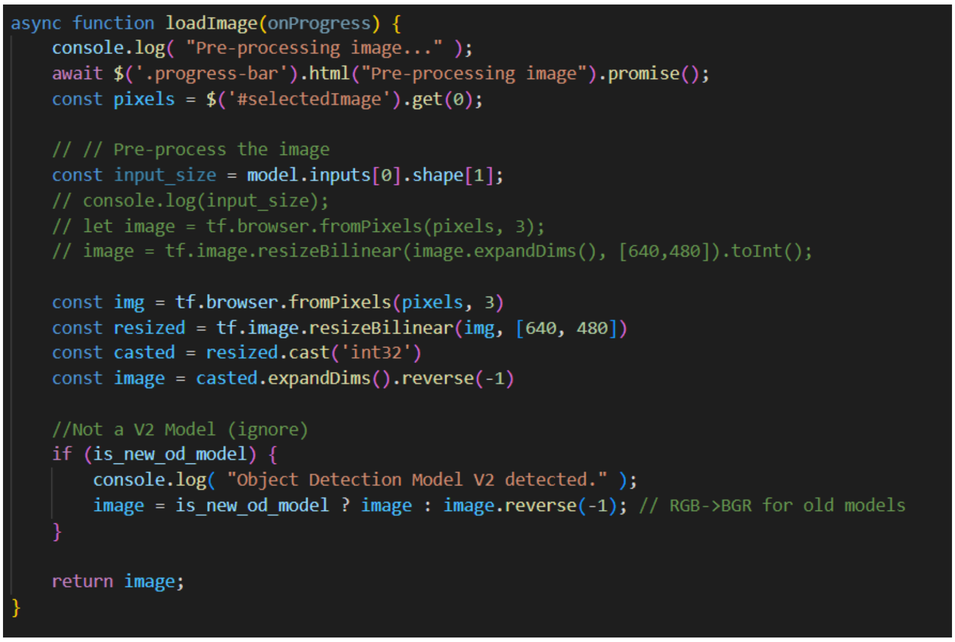
The JavaScript code for the website can be seen in Figure 40. A JavaScript file uses JavaScript as the programming language. JavaScript is a popular programming language for designing web pages. JavaScript has many uses, such as user interface and interpreting data (ECMA International. (2015)). The first block of code includes a boolean variable to check if the image was inserted on the web page. To do this, Jquery checks if the image selector ID is changed using the change function. The image file is created and read. When the file is opened, the url link of the image is retrieved and the url pasted the image on the selected image attribute. The image loaded variable then becomes true, proving that the image loaded. The file variable acts as a button and when the user presses the choose file button, it allows the user to select a file from the file manager and then the program reads the image as a url. Another function ‘showProgress’ is created to show the progress percentage of the progress bar that loads the model when the user opens the website. This percentage is also printed in the console for developer debugging purposes. There are three variables that represent the custom model. The first variable ‘model’ stores the custom model inside. The second variable ‘is new od model’ is used for checking if the model is a new object detection model or old one. The third variable checks if the model loaded successfully (Dittmann, 2023).



```
let model;
let is_new_od_model;
let modelLoaded = false;

$( document ).ready(async function () {
    modelLoaded = false;
    $('.progress-bar').html("Loading Model");
    $('.progress-bar').show();
    console.log( "Loading model..." );
    model = await tf.loadGraphModel('https://breadboardtfod.s3.eu-gb.cloud-object-storage.appdomain.cloud/model.json', {onProgress: showProgress});
    is_new_od_model = model.inputs.length == 3; //true or false
    // console.log(model.inputs.length);
    console.log( "Model loaded." );
    $('.progress-bar').hide();
    modelLoaded = true;
});

function _logistic(x) {
    if (x > 0) {
        return (1 / (1 + Math.exp(-x)));
    } else {
        const e = Math.exp(x);
        return e / (1 + e);
    }
}
```

Figure 41: Loading the model.

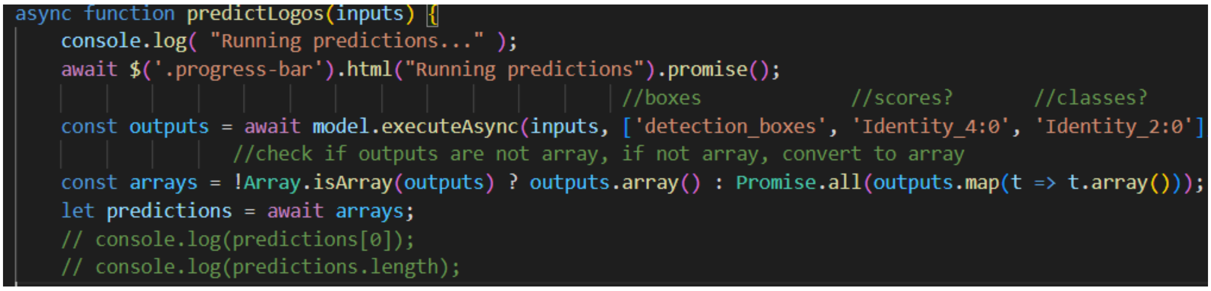
As shown in Figure 41, when the HTML document is ready, the model is loaded through a url link to the model.json custom model file on the IBM Cloud object storage. While this model is loading, a progress bar is animated at the top of the web site that says, “Loading Model”. Once the model has loaded, the progress bar disappears and the console prints, “Model Loaded.”. This is an asynchronous function because the model must load first before the rest of the code gets executed. This happens because of the await keyword, which pauses the async function to wait until model is loaded (ECMA International. (2021)). The logistic function is used in older models for post-processing the bounding box (Dittmann, 2023).



```
async function loadImage(onProgress) {
    console.log( "Pre-processing image..." );
    await $('.progress-bar').html("Pre-processing image").promise();
    const pixels = $('#selectedImage').get(0);

    // // Pre-process the image
    const input_size = model.inputs[0].shape[1];
    // console.log(input_size);
    // let image = tf.browser.fromPixels(pixels, 3);
    // image = tf.image.resizeBilinear(image.expandDims(), [640,480]).toInt();

    const img = tf.browser.fromPixels(pixels, 3)
    const resized = tf.image.resizeBilinear(img, [640, 480])
    const casted = resized.cast('int32')
    const image = casted.expandDims().reverse(-1)

    //Not a V2 Model (ignore)
    if (is_new_od_model) {
        console.log( "Object Detection Model V2 detected." );
        image = is_new_od_model ? image : image.reverse(-1); // RGB->BGR for old models
    }

    return image;
}
```

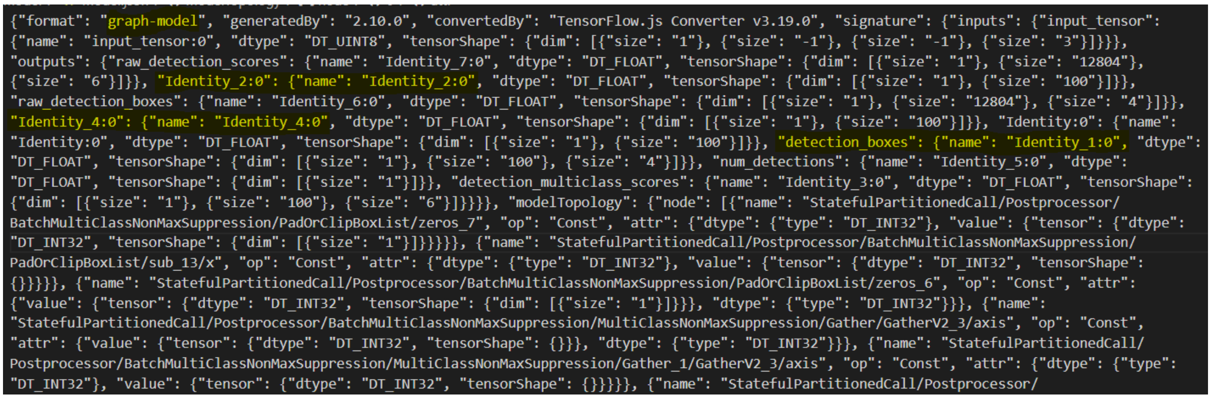
Figure 42: Pre-processing the Image.

Figure 42 shows the pre-processing of the selected image. The image is retrieved from the selected image attribute. For debugging purposes, the model input shape is stored in a variable ‘input size’. Once the image is retrieved, the image goes through pre-processing, which takes the original image, resizes it, changes the data type of the image to integers and the image array dimensions are expanded and ordered using the reverse function. The next conditional statement is created to check if the custom model is an object detection v2 model for extra pre-processing steps (Dittmann, 2023).



```
async function predictLogos(inputs) {
    console.log( "Running predictions..." );
    await $('.progress-bar').html("Running predictions").promise();
                                                    //boxes              //scores?       //classes?
    const outputs = await model.executeAsync(inputs, ['detection_boxes', 'Identity_4:0', 'Identity_2:0']
                //check if outputs are not array, if not array, convert to array
    const arrays = !Array.isArray(outputs) ? outputs.array() : Promise.all(outputs.map(t => t.array()));
    let predictions = await arrays;
    // console.log(predictions[0]);
    // console.log(predictions.length);
```

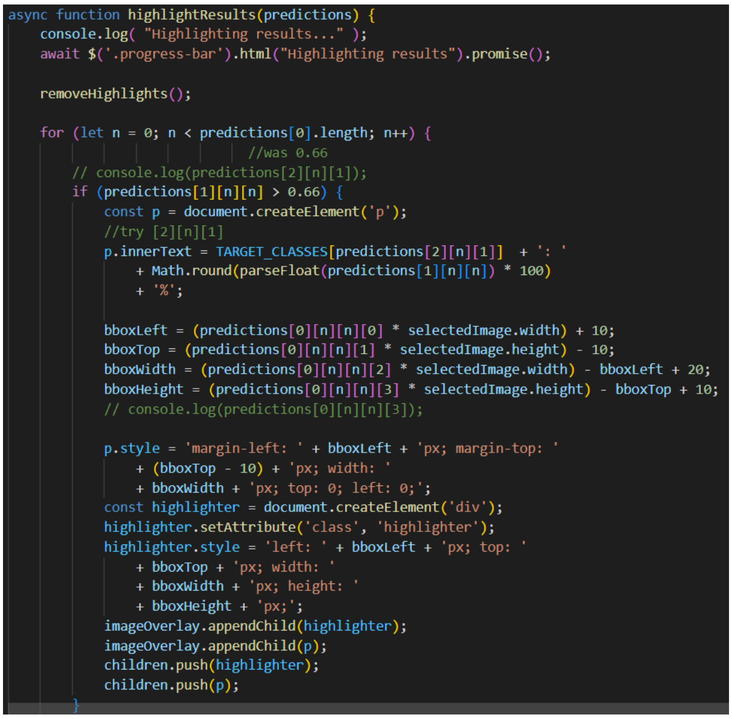
Figure 43: Running Predictions



```
{"format": "graph-model", "generatedBy": "2.10.0", "convertedBy": "TensorFlow.js Converter v3.19.0", "signature": {"inputs": {"input_tensor":
{"name": "input_tensor:0", "dtype": "DT_UINT8", "tensorShape": {"dim": [{"size": "1"}, {"size": "-1"}, {"size": "-1"}, {"size": "3"}]}}},
"outputs": {"raw_detection_scores": {"name": "Identity_7:0", "dtype": "DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}, {"size": "12804"},
{"size": "6"}]}}, "Identity_2:0": {"name": "Identity_2:0", "dtype": "DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}, {"size": "100"}]}},
"raw_detection_boxes": {"name": "Identity_6:0", "dtype": "DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}, {"size": "12804"}, {"size": "4"}]}},
"Identity_4:0": {"name": "Identity_4:0", "dtype": "DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}, {"size": "100"}]}}, "Identity:0": {"name":
"Identity:0", "dtype": "DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}, {"size": "100"}]}}, "detection_boxes": {"name": "Identity_1:0", "dtype":
"DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}, {"size": "100"}, {"size": "4"}]}}, "num_detections": {"name": "Identity_5:0", "dtype":
"DT_FLOAT", "tensorShape": {"dim": [{"size": "1"}]}}, "detection_multiclass_scores": {"name": "Identity_3:0", "dtype": "DT_FLOAT", "tensorShape":
{"dim": [{"size": "1"}, {"size": "100"}, {"size": "6"}]}}}}, "modelTopology": {"node": [{"name": "StatefulPartitionedCall/Postprocessor/
BatchMultiClassNonMaxSuppression/PadOrClipBoxList/zeros_7", "op": "Const", "attr": {"dtype": {"type": "DT_INT32"}, "value": {"tensor": {"dtype":
"DT_INT32", "tensorShape": {"dim": [{"size": "1"}]}}}}}, {"name": "StatefulPartitionedCall/Postprocessor/BatchMultiClassNonMaxSuppression/
PadOrClipBoxList/sub_13/x", "op": "Const", "attr": {"dtype": {"type": "DT_INT32"}, "value": {"tensor": {"dtype": "DT_INT32", "tensorShape":
{}}}}}, {"name": "StatefulPartitionedCall/Postprocessor/BatchMultiClassNonMaxSuppression/PadOrClipBoxList/zeros_6", "op": "Const", "attr":
{"value": {"tensor": {"dtype": "DT_INT32", "tensorShape": {"dim": [{"size": "1"}]}}}, "dtype": {"type": "DT_INT32"}}}, {"name":
"StatefulPartitionedCall/Postprocessor/BatchMultiClassNonMaxSuppression/MultiClassNonMaxSuppression/Gather/GatherV2_3/axis", "op": "Const",
"attr": {"value": {"tensor": {"dtype": "DT_INT32", "tensorShape": {}}}, "dtype": {"type": "DT_INT32"}}}, {"name": "StatefulPartitionedCall/
Postprocessor/BatchMultiClassNonMaxSuppression/MultiClassNonMaxSuppression/Gather_1/GatherV2_3/axis", "op": "Const", "attr": {"dtype": {"type":
"DT_INT32"}, "value": {"tensor": {"dtype": "DT_INT32", "tensorShape": {}}}}}, {"name": "StatefulPartitionedCall/Postprocessor/
```

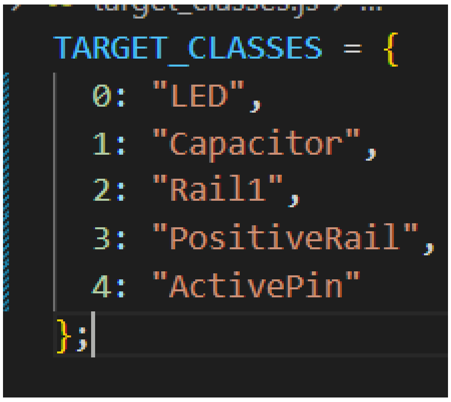
Figure 44: Model Json File

Figure 43 shows the function that executes the custom model and forms predictions. The outputs variable executes the model asynchronously. The input parameter of the function is the pre-processed image. The image needs to show detection boxes (bounding box), scores (confidence score) and classes (object label). These names of these three detections can be found in the model json file. As shown in Figure 44, the model file format was exported as a graph-model. This means that the program must use Tensorflow's loadGraphModel function. The 'detection_boxes' name is also known as 'Identity_1:0' and either name can be used in the executing of the model, as this draws the bounding box on the image. 'Identity_2:0' refers to the object class, which is the name of the object. 'Identity_4:0' refers to the object confidence score. These were found from debugging inside the website console. The arrays variable in Figure 43 checks if the outputs from the model are arrays using the '?' syntax. If they are not arrays, the outputs get converted to arrays. 'Promise.all' maps all the outputs and logs them in the console on the website in a readable manner. These output arrays are then stored inside the 'predictions' variable (Dittmann, 2023).



```
async function highlightResults(predictions) {
    console.log( "Highlighting results..." );
    await $('.progress-bar').html("Highlighting results").promise();

    removeHighlights();

    for (let n = 0; n < predictions[0].length; n++) {
                                    //was 0.66
        // console.log(predictions[2][n][1]);
        if (predictions[1][n][n] > 0.66) {
            const p = document.createElement('p');
            //try [2][n][1]
            p.innerText = TARGET_CLASSES[predictions[2][n][1]]  + ': '
                + Math.round(parseFloat(predictions[1][n][n]) * 100)
                + '%';

            bboxLeft = (predictions[0][n][n][0] * selectedImage.width) + 10;
            bboxTop = (predictions[0][n][n][1] * selectedImage.height) - 10;
            bboxWidth = (predictions[0][n][n][2] * selectedImage.width) - bboxLeft + 20;
            bboxHeight = (predictions[0][n][n][3] * selectedImage.height) - bboxTop + 10;
            // console.log(predictions[0][n][n][3]);

            p.style = 'margin-left: ' + bboxLeft + 'px; margin-top: '
                + (bboxTop - 10) + 'px; width: '
                + bboxWidth + 'px; top: 0; left: 0;';
            const highlighter = document.createElement('div');
            highlighter.setAttribute('class', 'highlighter');
            highlighter.style = 'left: ' + bboxLeft + 'px; top: '
                + bboxTop + 'px; width: '
                + bboxWidth + 'px; height: '
                + bboxHeight + 'px;';
            imageOverlay.appendChild(highlighter);
            imageOverlay.appendChild(p);
            children.push(highlighter);
            children.push(p);
        }
```

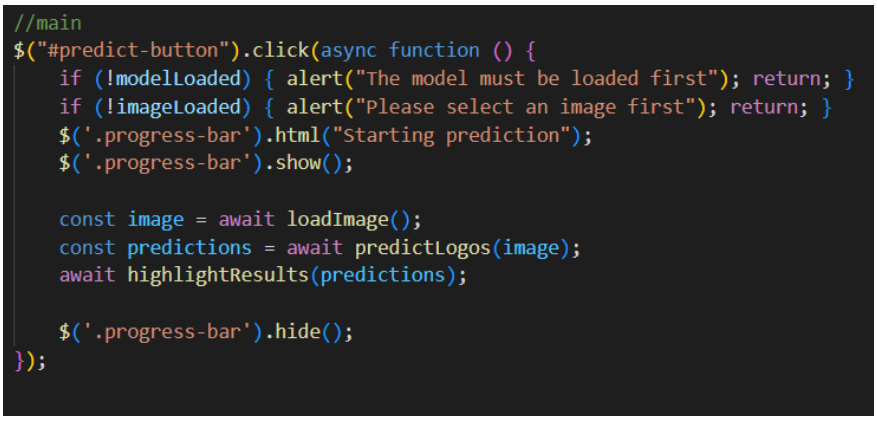
Figure 45: Highlighting the Results.



```
TARGET_CLASSES = {
  0: "LED",
  1: "Capacitor",
  2: "Rail1",
  3: "PositiveRail",
  4: "ActivePin"
};
```

Figure 46: Object Classes.

After the predictions are returned from the previous function, this ‘highlightResults’ function highlights the predictions on the image. To summarize this for-loop, the predictions array is cycled through and checks if any of the confidence score predictions inside the array are greater than 66%. If they are, a new element on the HTML document is created with the name ‘p’. This is a placeholder for the class name to be stored inside the element. The element text is then changed to the object class name and concatenated with the confidence score as a percentage. After this, the bounding box left, and top sides are calculated along with the width and height of the bounding box. These bounding box calculations allow for the drawing of the bounding box on the image, which can be at the bottom of Figure 46. The class name is styled. A new document element ‘div’ is stored in a variable ‘highlighter’ and given the class property. The highlighter is then styles and highlights the bounding box. The bounding box and object class label styles is then added to the image overlay (Dittmann, 2023).



```
//main
$("#predict-button").click(async function () {
    if (!modelLoaded) { alert("The model must be loaded first"); return; }
    if (!imageLoaded) { alert("Please select an image first"); return; }
    $('.progress-bar').html("Starting prediction");
    $('.progress-bar').show();

    const image = await loadImage();
    const predictions = await predictLogos(image);
    await highlightResults(predictions);

    $('.progress-bar').hide();
});
```

Figure 47: Main Function

The main function can be seen in Figure 47. This is where everything gets executed once the user clicks the predict button on the web site. If the model is not loaded when the predict button is clicked, the user is prompted a message for the model to be loaded first. If the image is not loaded when the predict button is clicked, the user is prompted to select an image first. The image is loaded, the predictions are made, and the results are highlighted on the image (Dittmann, 2023).

4.5 Results and Website Design

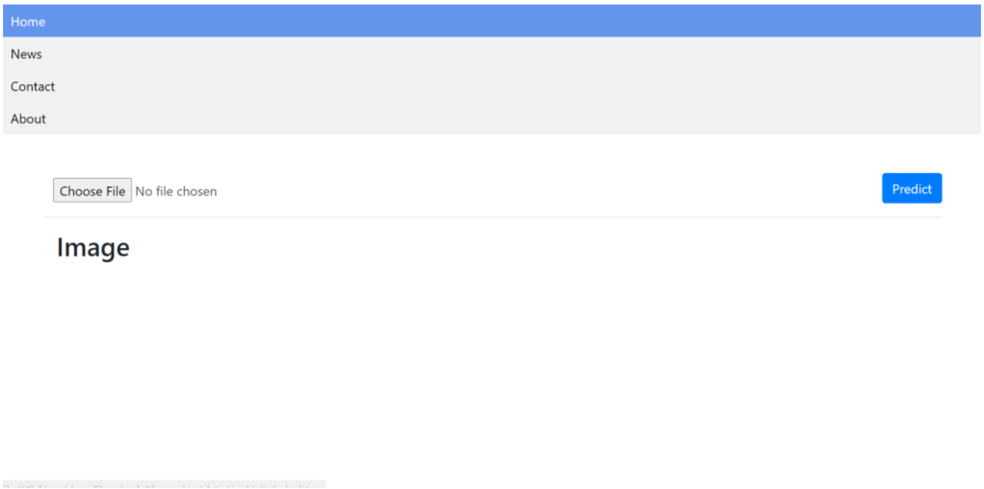
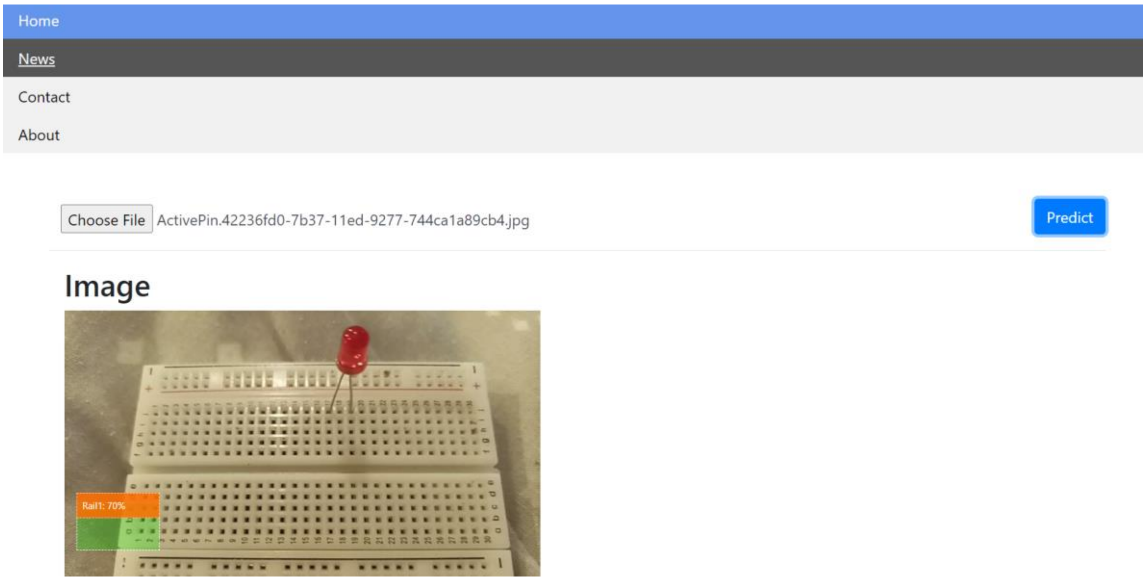
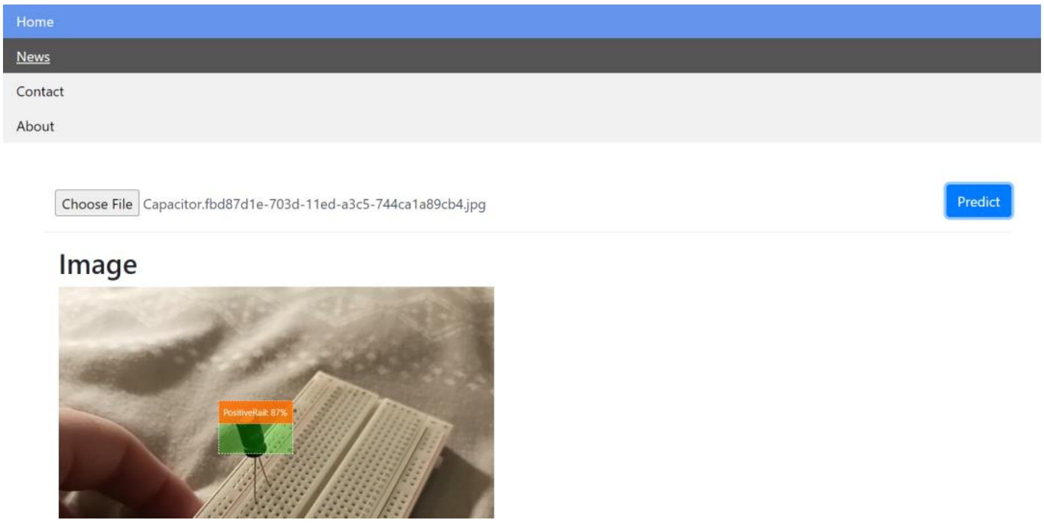


Figure 48: The Main Website Page



ActivePin.42236fd0-7b37-11ed-9277-744ca1a89cb4.jpg

Figure 49: Rail1 Results



Capacitor.fbd87d1e-703d-11ed-a3c5-744ca1a89cb4.jpg

Figure 50: Capacitor Results

Figure 48 shows the website working after opening the index.html file from the file manager. The website is simplistic but easy to understand. The top of the website includes the navigation bar for the user to navigate between pages on the website. The choose file button is in the centre of the screen. The user can select an image using this button and make predictions using the predict button. Figure 49 shows the model successfully located the first rail of the breadboard. Figure 50 shows the results for the capacitor. The model located the capacitor but instead detected a positive rail on the breadboard. This shows that the model failed to identify the object class label and needed more training and optimization such as switching the pretrained model. There could also be an issue with the drawing of the bounding box or there may be incorrect debugging data on the source code JavaScript file (Dittmann, 2023).

4.6 Conclusion

This chapter focused on the final project development. The final project development required three creating a HTML file, CSS file and JavaScript file. These three files form a website. From the results, the user can use this website to input an image and make predictions on it. The output draws the detections around the electrical components in the image. The outputs from the image are not consistent, as the confidence rate is not high enough, so the model predicts the wrong objects occasionally. The schematic is not drawn on the website, but it can be drawn in the Jupyter Notebook detection file. For future, the next step should involve producing schematics from detections, optimizing the model and adding extra object classes to be detected.

5 User Testing

5.1 Introduction

The purpose of this user testing section is to receive feedback on the project from users that have used and tested it. Feedback for the project is essential to improving the design and architecture of the website because users can give insight to what looks appealing to them. The users can also test the functionality of the website, deciding whether it is suitable, easy to understand, and that the website is working. The user can then make a conclusion, deciding whether the website is helpful and effective.

5.2 Participants

The participants of this project include three users. The first user is a 27-year-old Irish male citizen that has a professional background in engineering for 2 years and has relatively small professional experience in machine learning. The second user is a 21-year-old Irish female citizen that has an educational background in medicine for 4 years. The third user is a 50-year-old female that has a professional background in childcare for 15 years.

5.3 Procedures

This section explores the procedures taken during the user testing to accomplish valuable data for future improvement of the project. Before the user testing took place, each user was asked a set of questions in relation to demographic, along with questions about their background and professional experience. After the questionnaire, each user was asked to complete a set of tasks and scenarios during the testing of the project. This list of scenarios was created to test the user's thought process while they were using the website. These scenarios are used to specify the user's decision-making on the website, along with how they visually navigated it, with or without ease.

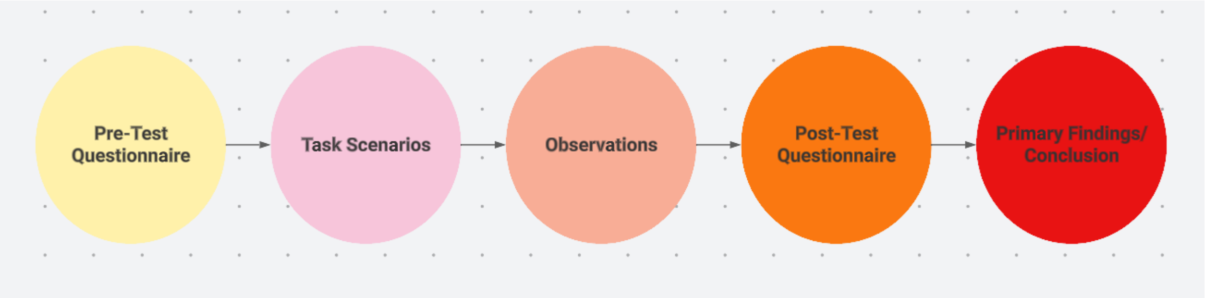
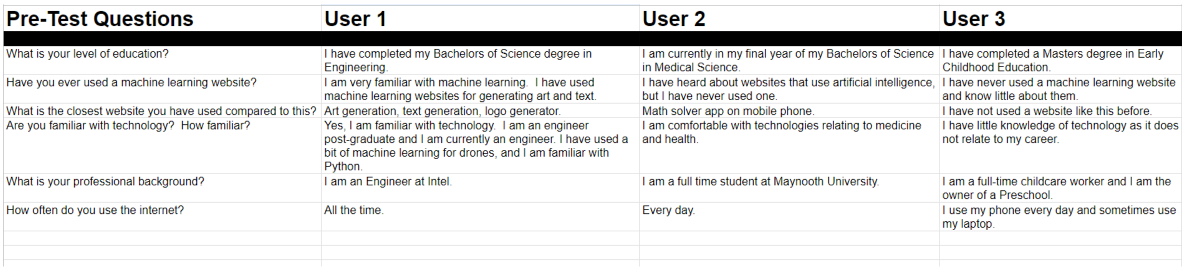


Figure 51: User Test Procedure.

Each user was observed, and the interactions were recorded, with the user's consent, for future improvements. The user's behavior and comments were also noted. After the user completed the task scenarios, another questionnaire took place to ask the user for feedback in relation to the website, such as usability, practicality and effectiveness. Finally, a conclusion was summarized from the user testing results. This procedure can be viewed in Figure 51.

5.3.1 Pre-Test Questions



Pre-Test Questions	User 1	User 2	User 3
What is your level of education?	I have completed my Bachelors of Science degree in Engineering.	I am currently in my final year of my Bachelors of Science in Medical Science.	I have completed a Masters degree in Early Childhood Education.
Have you ever used a machine learning website?	I am very familiar with machine learning. I have used machine learning websites for generating art and text.	I have heard about websites that use artificial intelligence, but I have never used one.	I have never used a machine learning website and know little about them.
What is the closest website you have used compared to this?	Art generation, text generation, logo generator.	Math solver app on mobile phone.	I have not used a website like this before.
Are you familiar with technology? How familiar?	Yes, I am familiar with technology. I am an engineer post-graduate and I am currently an engineer. I have used a bit of machine learning for drones, and I am familiar with Python.	I am comfortable with technologies relating to medicine and health.	I have little knowledge of technology as it does not relate to my career.
What is your professional background?	I am an Engineer at Intel.	I am a full time student at Maynooth University.	I am a full-time childcare worker and I am the owner of a Preschool.
How often do you use the internet?	All the time.	Every day.	I use my phone every day and sometimes use my laptop.

Figure 52: Pre-Test Questionnaire.

Figure 52 shows a questionnaire that was used before the user task scenarios and user testing took place. The first question asked in the questionnaire is to understand the level of education the user has received. In this question, it is notable that user 3 has completed a master's degree and therefore has received the highest education, after user 1 and user 2 respectively. The second question asked the users if they have ever used a machine learning website. The first user has used several machine learning websites, the second user said they have heard of machine learning websites, but they have never used one, the third user said that they know little about machine learning websites and have never used one. The third question asks the user for the closest website that they have used compared to Breadboard2Schematic. The first user used several machine learning websites that are like this website, such as a text generator. The second user said they have used a machine learning app on mobile phone, but not a website. The third user mentions that they have never used a website like Breadboard2Schematic. The fourth question asked the users for their professional backgrounds. The first user is an engineer, the second is a full-time student and the third is an owner of a preschool. The last question discusses how often the users use the internet. For this question, the first and second user use it often and the third user uses their mobile phone every day and sometimes uses their laptop.

5.3.2 Task Scenarios

In this section, a scenario was given to each user. Each user received a different scenario.

User 1 scenario: “You are studying schematics and you do not know how to draw the circuit that is in front of you, as a schematic drawing. Please find a way to do this using the website”.

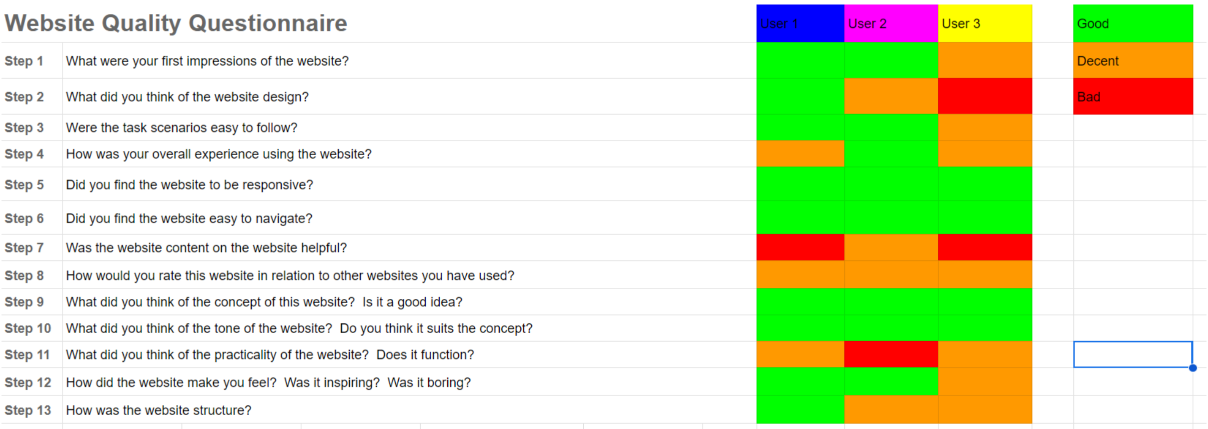
User 2 scenario: “You have taken interest in electronics and started by using breadboards. You want to learn how to draw a schematic diagram from the circuit you made on the breadboard, but you are unsure how to. Please use the website to take a photo of the breadboard to create a schematic”.

User 3 scenario: “You are completely new to electronics and have just bought an electronics kit, but you do not know the name of anything inside the kit and heard that it can be used to educate people on electronics. Please try uploading a photo of something in your kit to the website.”.

5.3.3 Observations

This section explores the observations that were made during the user testing phase. In relation to completing the tasks given, user 1 had no trouble completing the task and was familiar with the concept of the website. User 2 did not have much trouble but was confused at first of what to do. User 3 had much trouble navigating the website and completing the task. The user also struggled to understand the concept of the website at first. User 1 commented that they like the design of the website and that it reminded them of other machine learning websites. User 2 claimed that the website reminded them of the machine learning app they had previously used for mathematics. User 3 commented that the website looks good but that they hoped for a more vibrant and colorful design.

5.3.4 Post-Test Questions



Website Quality Questionnaire		User 1	User 2	User 3		Good
Step 1	What were your first impressions of the website?	🟩	🟩	🟧		Decent
Step 2	What did you think of the website design?	🟩	🟧	🟥		Bad
Step 3	Were the task scenarios easy to follow?	🟩	🟩	🟧		
Step 4	How was your overall experience using the website?	🟧	🟩	🟧		
Step 5	Did you find the website to be responsive?	🟩	🟩	🟩		
Step 6	Did you find the website easy to navigate?	🟩	🟩	🟩		
Step 7	Was the website content on the website helpful?	🟥	🟧	🟥		
Step 8	How would you rate this website in relation to other websites you have used?	🟧	🟧	🟧		
Step 9	What did you think of the concept of this website? Is it a good idea?	🟩	🟩	🟩		
Step 10	What did you think of the tone of the website? Do you think it suits the concept?	🟩	🟩	🟩		
Step 11	What did you think of the practicality of the website? Does it function?	🟧	🟥	🟧		
Step 12	How did the website make you feel? Was it inspiring? Was it boring?	🟩	🟩	🟧		
Step 13	How was the website structure?	🟩	🟧	🟧		

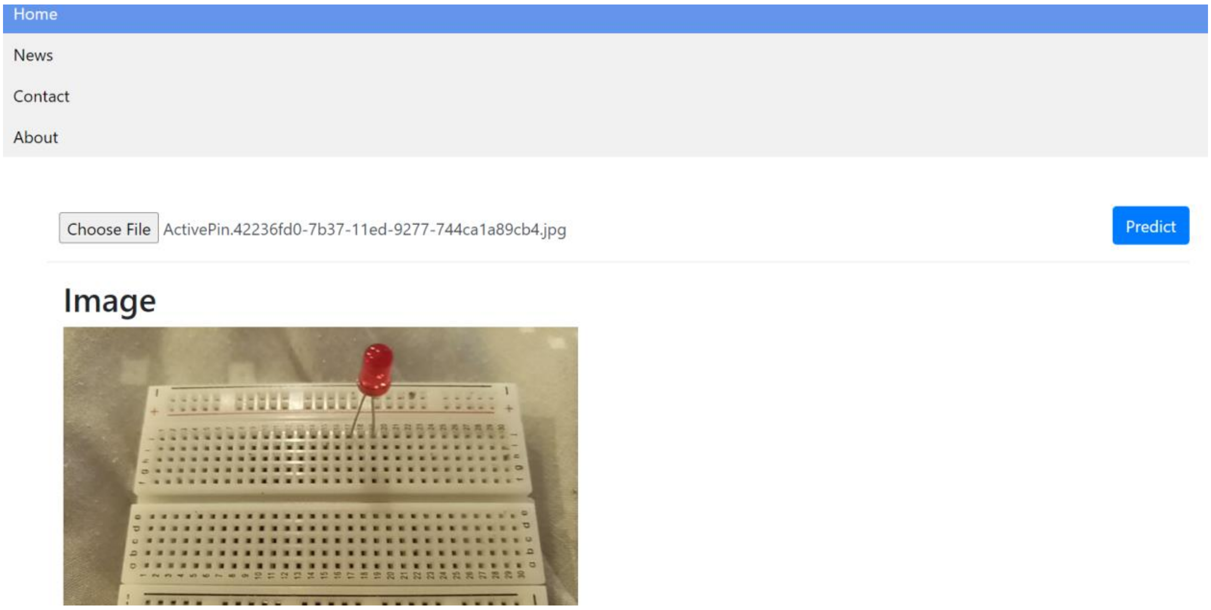
Figure 53: Post-Test Qualitative Survey

Figure 53 shows the results of the post-test qualitative survey that was given to each user. Generally, the first user found the website quality to be good. The second user found the website to be mostly good. The third user struggled with the use of the website and did not like the website from their experience with it.

5.4 Conclusion

To summarize this chapter, the user testing procedure consists of five steps. The first step is to give the user a pre-test questionnaire. This questionnaire asks demographic questions and questions that relate to their use of machine learning websites. The second step involved asking the users to complete a task and giving the users scenarios to visualize a case that they could use this project. The third step involved writing observations and visually checking how the user completed the task during the user test, as well as their behavior towards the website and writing the comments they made. The next step is to give the user a post-test qualitative survey, where the users can answer questions on their experience with the website and test. The final stage is summarizing the user's experiences and drawing conclusions from them.

6 Conclusion



ActivePin.42236fd0-7b37-11ed-9277-744ca1a89cb4.jpg

Figure 54: Final Project Conclusions

Figure 54 shows a final image of Breadboard2Schematic. In conclusion, the achievements produced included developing an end-to-end website and machine learning model that identifies components in an image. The prototype version of this project also achieved a schematic diagram output from the images. The future improvements of the project include adding a schematic output element to the website after post-processing the image. More future improvements include further optimization of the model and extra training and testing. In general, Breadboard2Schematic was mostly a successful project that achieved the concept effectively. The feedback received from the user testing phase can also be used to further develop and improve this project. The experience received in completing this project includes developing a machine learning model fully and deploying it to a website. Another experience this project has brought includes building and designing a website end-to-end, as well as gathering user feedback for further enhancing the project in the future. The results of Breadboard2Schematic have proven the speed, accuracy and effectiveness of machine learning in electrical circuitry and has proven the novelty and potential of this project for future research.

7 References

Brownlee, J. (2017). A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

Brownlee, J. (2021). Save and Load Machine Learning Models in Python with TensorFlow 2. Machine Learning Mastery. https://machinelearningmastery.com/save-load-machine-learning-models-tensorflow-2/

Bootstrap. (n.d.). Getting started. Retrieved from https://getbootstrap.com/docs/5.0/getting-started/introduction/

CDNJS. (n.d.). The #1 free and open source CDN built to make life easier for developers. Retrieved May 7, 2023, from https://cdnjs.com/

Chen, Y. (2020). Introduction to virtual environments in Python. Real Python. Retrieved from https://realpython.com/python-virtual-environments-a-primer/
(2023). Intellipaat.com. https://intellipaat.com/blog/wp-content/uploads/2018/07/ALMLDL-01.jpg

Cloudflare, Inc. (n.d.). cdnjs. Retrieved May 7, 2023, from https://cdnjs.com/

Dittmann, S. (2023, April 19). *TensorFlow.js Example: Using an Azure Custom Vision Object Detection model to detect Logos in a web browser*. GitHub. https://github.com/SaschaDittmann/tfjs-cv-objectdetection

ECMA International. (2021). ECMAScript® Language Specification, 12th edition. Retrieved May 7, 2023, from https://www.ecma-international.org/ecma-262/12.0/

ECMA International. (2015). ECMAScript® 2015 Language Specification. Retrieved May 7, 2023, from https://www.ecma-international.org/ecma-262/6.0/

Freeman, A., & Robson, B. (2011). jQuery Cookbook: Solutions & Examples for jQuery Developers. O'Reilly Media.

Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media, Inc.

GitHub. (n.d.). *GitHub*. GitHub. https://github.com/

GNU Wget. (n.d.). GNU Wget 1.21.1 Manual. GNU Project. Retrieved May 6, 2023, from https://www.gnu.org/software/wget/manual/wget.html

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Google Developers. (n.d.). Protocol buffers. Retrieved from https://developers.google.com/protocol-buffers

Google. (2019). *Google Colaboratory*. Google.com. https://colab.research.google.com/

Google LLC. (n.d.). Protocol Buffer Basics: Python. Retrieved from https://developers.google.com/protocol-buffers/docs/pythontutorial

Google. (2021). TFRecord and tf.train.Example. TensorFlow. Retrieved from https://www.tensorflow.org/tutorials/load_data/tfrecord

Google. (2021). Welcome to Colaboratory. https://research.google.com/colaboratory/faq.html.

Gupta, A. (2020). Model configuration in deep learning. Retrieved from https://towardsdatascience.com/model-configuration-in-deep-learning-666526b0a17b

IBM Cloud Docs. (n.d.). Cloud.ibm.com. Retrieved May 6, 2023, from https://cloud.ibm.com/docs/containers?topic=containers-cos_plugin_changelog

IBM Cloud. (2019). Ibm.com. https://www.ibm.com/cloud

Jupyter. (2019). *Project Jupyter*. Jupyter.org. https://jupyter.org/

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & Development Team, J. (n.d.). *Jupyter Notebooks-a publishing format for reproducible computational workflows*. https://doi.org/10.3233/978-1-61499-649-1-87

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., & Ivanov, P. (n.d.). Jupyter kernels. Project Jupyter. https://jupyter-client.readthedocs.io/en/stable/kernels.html

Leach, P., Mealling, M., & Salz, R. (2005). A Universally Unique IDentifier (UUID) URN Namespace. IETF. RFC 4122. https://tools.ietf.org/html/rfc4122

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. doi: 10.1038/nature14539

Lin, T. (2018). LabelImg. GitHub. https://github.com/tzutalin/labelImg

LucidChart. (2019). Lucidchart.com. https://www.lucidchart.com

lxml. (n.d.). lxml: XML and HTML with Python. Retrieved May 6, 2023, from https://lxml.de/

Matplotlib. (2012). *Matplotlib: Python plotting — Matplotlib 3.1.1 documentation*. Matplotlib.org. https://matplotlib.org/

Mozilla. (n.d.). <main>: The Main Content Element. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main

Mozilla. (n.d.). The viewport meta tag. MDN web docs. https://developer.mozilla.org/en-US/docs/Web/HTML/Viewport_meta_tag.

Microsoft. (2021). Visual Studio Code. https://code.visualstudio.com/

Numpy. (2009). *NumPy*. Numpy.org. https://numpy.org/

National Instruments. (n.d.). Schematic Diagrams. Retrieved from https://www.ni.com/en-us/support/documentation/supplemental/08/schematic-diagrams.html

Nicholas Renotte - YouTube. (n.d.). Www.youtube.com. https://www.youtube.com/@NicholasRenotte

OpenCV. (n.d.). About OpenCV. Retrieved from https://opencv.org/about/

OpenCV. (2019). *OpenCV library*. Opencv.org. https://opencv.org/

Piatetsky-Shapiro, G. (2021). Limitations of machine learning: From evaluation to the broader social context. Communications of the ACM, 64(2), 22-25. https://doi.org/10.1145/3447279

pip. (2019, February 20). PyPI. https://pypi.org/project/pip/

Project Jupyter. (n.d.). ipykernel. GitHub. https://github.com/ipython/ipykernel

ProspectOne. (n.d.). jsDelivr. Retrieved May 7, 2023, from https://www.jsdelivr.com/

Python Packaging Authority. (n.d.). Pip user guide. Retrieved from https://pip.pypa.io/en/stable/user_guide/

Python Packaging Authority. (2022). Installing packages. Python Packaging Authority. Retrieved May 9, 2023, from https://packaging.python.org/tutorials/installing-packages/

Python Software Foundation. (n.d.). Command line and environment. https://docs.python.org/3/using/cmdline.html

Python Software Foundation. (2021). os — Miscellaneous operating system interfaces. Python 3.10.0 documentation. https://docs.python.org/3/library/os.html

Python Software Foundation. (2021). 8.1. time — Time access and conversions — Python 3.10.0 documentation. https://docs.python.org/3/library/time.html

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Renotte, N. (2023, April 11). *Tensorflow Object Detection React Application*. GitHub. https://github.com/nicknochnack/TFODApp

Renotte, N. (2022, March 29). *Tensorflow Object Detection Walkthrough*. GitHub. https://github.com/nicknochnack/TFODCourse

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

Riverbank Computing Limited. (n.d.). pyrcc5 - PyQt 5.15.4 Reference Guide. Retrieved May 9, 2023, from https://www.riverbankcomputing.com/static/Docs/PyQt5/pyrcc5.html

Riverbank Computing. (n.d.). PyQt. Retrieved from https://www.riverbankcomputing.com/software/pyqt/

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y

Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach. Pearson Education.

Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210-229.

Sascha Dittmann - YouTube. (n.d.). Www.youtube.com. Retrieved May 7, 2023, from https://www.youtube.com/channel/UCbX8EdsCb-jFDfr05hzMjjw

Schemdraw documentation — Schemdraw 0.16 documentation. (n.d.). Schemdraw.readthedocs.io. Retrieved May 6, 2023, from https://schemdraw.readthedocs.io/en/latest/

Simple Gantt Chart. (n.d.). Vertex42.com. https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html?utm_source=ms&utm_medium=file&utm_campaign=office&utm_content=url

StackPath, LLC. (n.d.). StackPath Bootstrap. Retrieved May 7, 2023, from https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css.

Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media.

T. Chen, "Introduction to TensorFlow for Deep Learning - Freezing a Model for Deployment," in Medium, Nov 14, 2017. [Online]. Available: https://towardsdatascience.com/introduction-to-tensorflow-for-deep-learning-freezing-a-model-for-deployment-badab007b987. [Accessed May 06, 2023].

Team, Ip. D. (n.d.). *ipykernel: IPython Kernel for Jupyter*. PyPI. Retrieved May 6, 2023, from https://pypi.org/project/ipykernel/

TensorFlow.js. (2019). TensorFlow. https://www.tensorflow.org/js

TensorFlow. (n.d.). TensorFlow. Retrieved November 24, 2022, from https://www.tensorflow.org/?gclid=CjwKCAiAyfybBhBKEiwAgtB7fkacVjtmrNnWIOaJ2IUWDNxk0goLvEJyqANhk_H_wFmPw_RrXRvTkRoClsQQAvD_BwE

TensorFlow Object Detection API. (2021). Model builder TF2 test. Retrieved from https://github.com/tensorflow/models/blob/master/research/object_detection/builders/model_builder_tf2_test.py

(2023). Shutterstock.com. https://www.shutterstock.com/shutterstock/videos/1064002759/thumb/1.jpg?ip=x480

Uizard. (n.d.). Uizard.io. https://uizard.io/

W3C. (2021). HTML. Retrieved from https://www.w3.org/html/

World Wide Web Consortium. (2018). Cascading Style Sheets (CSS) Snapshot 2018. Retrieved May 7, 2023, from https://www.w3.org/TR/CSS/

W3Schools. (n.d.). HTML Meta Tags. Retrieved from https://www.w3schools.com/tags/tag_meta.asp

CMT (DL835) Student Projects

‘Low Risk’

Standard Risk Assessment Template

For activities carried out in the School of Creative Technologies facilities

and at the student’s home.

These are projects where all Hazards are Ranked as a 3rd Rating.

See Risk Rating Matrices on Pages 10 and 11.

Project Risk Assessments and the methodology are needed to comply with the Safety, Health and welfare at work act 2005 and all other relevant Legislation. This document is based on the ‘Joint Risk Assessments’ procedure – IADT – December 2010.

The document has been updated to now include:

- Home working risk identification and control
- Covid-19 risk identification and control
- The Low Voltage Directive, LDV (previously in a separate document

PROGRAMME/YEAR:

STUDENT NAME: SHANE SMYTH	SUPERVISOR: Dr. Paul Comiskey
DIGITAL SIGNATURE: Shane Smyth	DIGITAL SIGNATURE:
DATE: 22/11/2022	DATE:

By signing this assessment, it is agreed by all parties that:

- the full facts relating to the health and safety aspects of the project have been declared by the student
- All parties are fully aware of the safety risks
- All parties will implement the control measures detailed, in order to reduce the contribution of the hazards to the level of the risks detailed.

Location of Work:	Home

Brief RELEVANT Details of project: 20 words	Full programming project that converts breadboard circuitry to schematics diagrams.

Step 1: Initial Hazards Identification

Risk Assessment No.	INITIAL HAZARD
1	Electrocution
2	Fire
3	Cutting injuries
4	Drilling injuries
5	Heavy equipment
6	Burn Injury
7	Fumes

Step 2: Risk Assessment Forms

(Start Overleaf)

Significant Hazard and consequences:	**1. Electrocution**

Who might be exposed to the hazards:	Students and staff

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	Circuit design must include features that will minimise likelihood of electrocution of anybody, when in an unsafe mode, eg use of fuses and circuit breakers. Short circuits should be identified and removed before the testing stage. Cable and insulation should be checked before testing stage. Power supply equipment should be PAT tested on a regular basis. Water sources should be kept away from the project, when in operation. Components, whether connected to power supplies or not, should be fully discharged before inspections – isolated from power supplies, are commenced. Eg discharge capacitors greater than 50μF via a 100Ω resistor.

Significant Hazard and consequences:	**2. Fire**

Who might be exposed to the hazards:	Students and staff

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	Maintain tidy work practices on benches and the laboratory environment. Keep combustible materials, eg paper, plastic, away from heat sources, such as soldering irons. Stow heat sources away safely when not in use, eg use a sturdy soldering iron stand. Circuit design must include features that will minimise likelihood of a fire, when in an unsafe mode, eg use of fuses and circuit breakers. Short circuits should be identified and removed before the testing stage. Suitable cable and insulation should be used, with a safety margin on the rating and size. Water sources should be kept away from the project, when in operation. Be aware of the locations of first aid kit and fire extinguishers, and use these items if suitably competent/trained.

	In the event of a fire, leave the laboratory and building in an orderly manner, and sound the fire alarm if it has not already automatically activated.

Significant Hazard and consequences:	**3. Cutting injuries**

Who might be exposed to the hazards:	Students

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	Maintain tidy work practices on benches and the laboratory environment. Develop projects with the minimum requirement for cutting any jagged edged in the final manufactured item. Use of quality, maintained tools and clamps if necessary. Use of a cutting board and goggles. Clear a space around the cutting area before commencing work. Tie back hair and loose clothing from the cutting area. Remove jewellery. Be aware of the location of the first aid kit, and use the kit if suitably competent/trained.

Significant Hazard and consequences:	**4. Drilling injuries**

Who might be exposed to the hazards:	Students and staff

Significant Hazard and consequences:	**5. Heavy equipment**

Who might be exposed to the hazards:	Students and staff

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	Maintain tidy work practices on benches and the laboratory environment. Undertake heavy lifting only if suitable advised and/or trained. Correct posture and lifting procedures. Use mechanical lifting aids where possible and appropriate. One or more persons to be involved in lifting or supervising the lifting of heavy equipment. Clear a space around the lifting area before commencing work. Tie back hair and loose clothing from the cutting area. Remove jewellery. Use protective footwear, and also headwear if necessary.

Significant Hazard and consequences:	**6. Burns**

Who might be exposed to the hazards:	Students and staff.

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	Maintain tidy work practices on benches and the laboratory environm Only use soldering irons and other hot-works appliances if training, given by staff, has been undergone. Keep combustible materials, eg paper, plastic, away from heat sourc such as soldering irons. Stow heat sources away safely when not in u eg use a sturdy soldering iron stand. Cable and insulation should be checked before using soldering irons electrically powered hot-works appliances. Use gloves, goggles and other personal protection equipment where necessary. Use cooling equipment, such as wet sponges for solderin irons. Do not allow water from any source to penetrate electrical cabl and wires. Tie back hair and loose clothing from the cutting area. Remove jewellery. Be aware of the locations of first aid kit and fire extinguishers, and us these items if suitably competent/trained.

	In the event of a fire, leave the laboratory and building in an orderly manner, and sound the fire alarm if it has not already automatically activated. Electrically powered hot-works equipment, such as soldering irons, should be checked and tested on a regular basis.

Significant Hazard and consequences:	**6. Fumes**

Who might be exposed to the hazards:	Students and Staff.

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	Use fume extraction equipment, eg for solder fumes. Keep laboratories well ventilated. Take frequent breaks from activities generating fumes. Employ a higher level of control measures when an individual suffers from a respiratory condition, such as asthma, taking advice from a GP. Solder fume extraction equipment and other similar items, should be maintained checked and tested on a regular basis.

Significant Hazard and consequences:	

Who might be exposed to the hazards:	

Proposed Control Measures – to be written in conjunction with project supervisor and revised at key project milestone dates	

Risk Rating Matrices

By looking at the hazard and asking how many people will be exposed to it, decide on the probability of an incident/accident occurring.

Example: Take an extension lead trailing along the floor up against the wall. The extension lead is a hazard and if it is in an office with one person working in it the probability and likelihood is “improbable” (see table No.1) because the lead is along the wall. However, if the lead is in a corridor with 200 people walking by there is a chance that someone could kick it out from the wall accidentally and create a greater probability/likelihood of a loss occurring thus upping its rating to “remote”.

When this is done you must decide on the seriousness of the loss, using the four columns on the left side of the Table No.2 below.

Example: Firstly taking the one person office example from above the possibility/likelihood is “improbable “but the result might be a “minor injury” e.g. scrape or a bruise. This gives us an “acceptable risk no action required” If we were to put the lead on a building site across an unguarded stairwell with 50 people using it the result is now possibly “fatal”. This gives us a “1ST rank action”.

Table No. 1:

PROBABILITY/LIKELIHOOD	DESCRIPTION
Likely/frequent	Occurs
Probable	Not Surprised. Will occur several times.
Possible	Could occur sometimes.
Remote	Unlikely, though conceivable.
Improbable	So unlikely that probability is close to zero.

Table No. 2:

	LIKELY	PROBABLE	POSSIBLE	REMOTE	IMPROBABLE
Fatal	1st	2nd	2nd	3rd	🟨
Major Injury/ permanent disability	2nd	2nd	3rd	🟨	🟨
Minor Injury	3rd	3rd	🟨	🟨	🟨
No Injury	🟨	🟨	🟨	🟨	🟨

By using the matrices above we now have an action needed ranking system. This means we can prioritize the hazards depending on their ranking.

<u>Example:</u> A 1st rank action requires immediate remedy

A 2nd rank action requires a less immediate remedy but must be dealt with quickly.

Table No. 3:

🟥	1st rank actions
🟧	2nd rank actions
🟩	3rd rank actions
🟨	Acceptable risk – no action

Taking all this information and pooling it in the Initial Hazard Identification log we can now prioritise the hazards in the left hand column "Risk Assessment No." This number will appear on the top left of the risk assessment forms for easy referencing.

Low Risk Project – Risk Rating Summary

Risk Assessment No.	INITIAL HAZARD	Probability	Ranking
1	Electrocution	Remote	3rd
2	Fire	Remote	3rd
3	Cutting injuries	Possible	Acceptable
4	Drilling injuries	Possible	3rd
5	Heavy equipment	Remote	3rd
6	Burns	Possible	3rd
7	Fumes	Probable	3rd

Additional Information supplied (if appropriate)

CMT Projects

‘Low Voltage Directive’ Compliance Statement.

DIRECTIVE 2006/95/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 12 December 2006 on the harmonisation of the laws of Member States relating to electrical equipment designed for use within certain voltage limits (codified version) (Text with EEA relevance)

Otherwise known as the Low Voltage Directive (LVD).

If the electrical aspects of your project have a voltage between 50 and 1000 V for alternating current and between 75 and 1500 V for direct current, for voltages at the **electrical input or output** (not internally), then you are signing to **confirm that it complies with all the safety requirements of the LVD**. You must read the LVD and analyse your project before signing, taking advice where necessary.

Student name: Shane Smyth

Student signature: Shane Smyth

Date: 22/11/2022

Project supervisor or delegated representative who is an engineer by discipline:

Name: Dr. Sivakumar Ramachandran

Signature:

Date: