Project Notebook

Win Barua // 100277378

Note: extension given until the June 8th

Initial Gantt Chart:

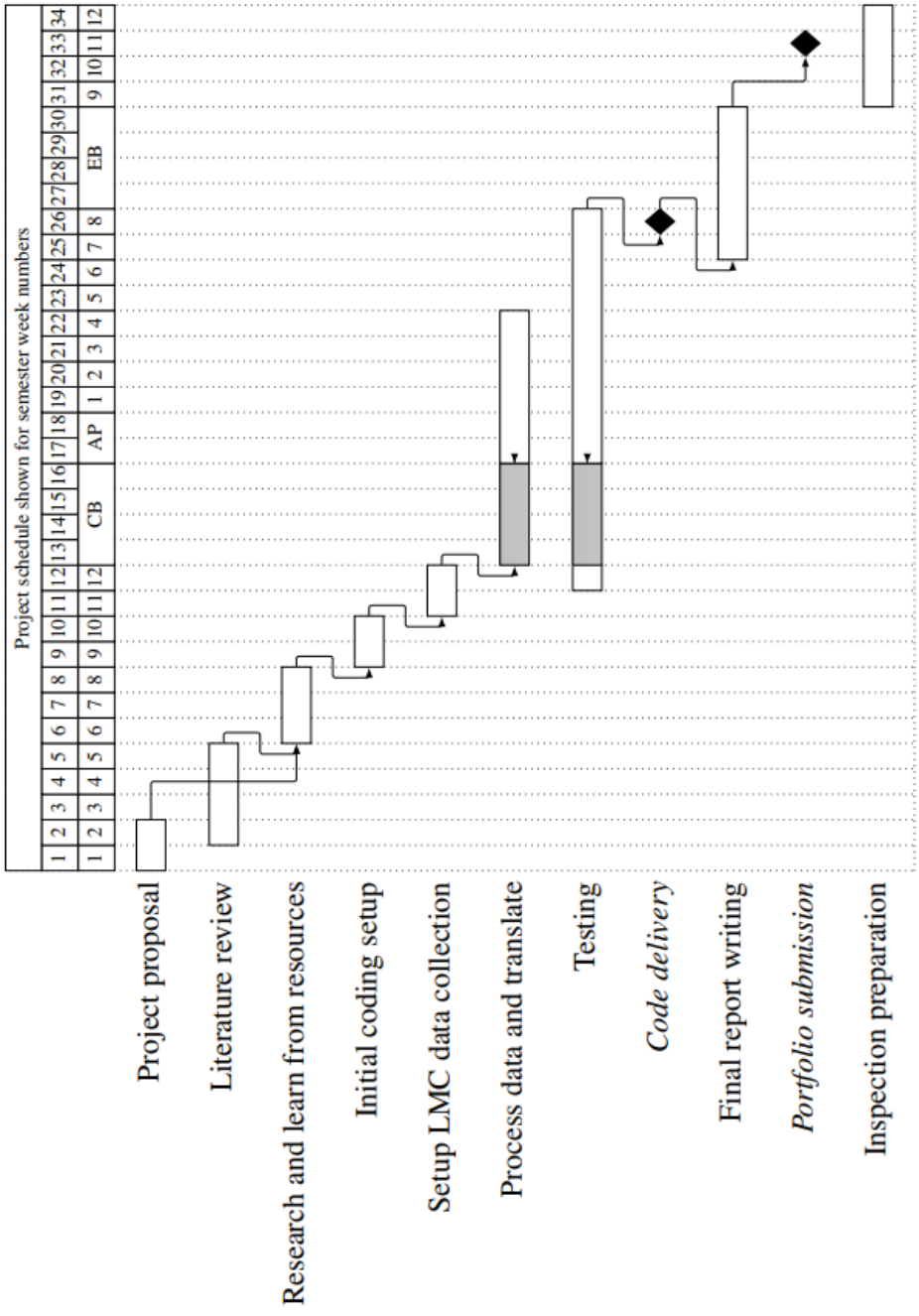


Figure 1: Project Gantt chart

Week 1

Started and completed project proposal slightly earlier than expected. Also started doing some initial research for the literature review.

Some useful sources I've found so far:

(PDF) American Sign Language Recognition Using Leap Motion Sensor (researchgate.net)

(PDF) Deep Learning Based Indian Sign Language Words Identification System (researchgate.net)

Some things I've learned:

- There are some attempts at a similar system using Leap motion.
- There are similar systems using computer vision.
- The average accuracy rate with the leap motion systems is around 75%.

Week 2

Continued similar systems research and started the Literature Review write up talking about my current findings. Completed the Introduction section and part of the Analysis of related work section

Notes:

- Similar systems have been done using flex sensors and 2D modelling.
- Free-hand interaction with leap motion controller for stroke rehabilitation | CHI '14 Extended Abstracts on Human Factors in Computing Systems (acm.org)

Week 3

Completed the Analysis of Related Works section, discussing in depth some of the sources I have found and how they function with the results they have yielded.

Week 4

No work on project

Week 5

This week I did some research and studying on how to apply the Leap Motion API to a Java program and use it to collect user hand data.

Week 6

Did some research and testing on how to create a basic machine learning model in order to learn PyTorch, however I did not like the features and libraries provided by it so I switched to TensorFlow Keras and did some basic projects.

Neural Network Multiclass Classification Model using TensorFlow | by Pasindu Ukwatta | Python in Plain English

Hands-on TensorFlow 2.0: Multi-Class Classifications with MLP | by Caner | Medium

Week 7

I had some problems with figuring out how to use TensorFlow so I had to do more tests to see how it works, especially when it came to multiclass classification using non-image data, in this case it is the sensor data given by the Leap Motion Controller.

Multi-Class Classification Tutorial with the Keras Deep Learning Library (machinelearningmastery.com)

Week 8

Switched the Data collection/Leap Motion data collection module to use Python instead of Java as it allowed for easier integration with my machine learning module.

Managed to set up a basic version of the Data collection module to collect some basic data from the user's hands, not all of the required features are collected yet, I'm only testing if it is working smoothly with the API and is writing to file properly.

Week 9

Completed the Data collection module, it will be used to collect all the data (features + labels) to train the machine learning module, and it will be later updated to be used as the interface for the user to activate the application.

Week 10, 11

No work on project

Week 12

Collected my first set of data that I will train the model with, I anticipate having to use multiple sets of data, maybe some with longer collection times or more trials to see which gives the model with the highest accuracy. Collected the data with 2 trials of 20 seconds for each sign.

Also created a simple model to train on this data, however, although the accuracy was about 95%, when I tested it using new data it would not be performing well at all, rarely getting the correct sign.

```
1 model.fit(train_x, train_y, epochs = 10, batch_size = 2)
```

```
Epoch 1/10
7588/7588 [==============================] - 10s 1ms/step - loss: 3.2913 - accuracy: 0.6716
Epoch 2/10
7588/7588 [==============================] - 9s 1ms/step - loss: 0.7013 - accuracy: 0.8947
Epoch 3/10
7588/7588 [==============================] - 8s 1ms/step - loss: 0.5517 - accuracy: 0.9133
Epoch 4/10
7588/7588 [==============================] - 8s 994us/step - loss: 0.2947 - accuracy: 0.9220
Epoch 5/10
7588/7588 [==============================] - 8s 1ms/step - loss: 0.2497 - accuracy: 0.9276
Epoch 6/10
7588/7588 [==============================] - 8s 1ms/step - loss: 0.2496 - accuracy: 0.9300: 0s - loss: 0.2498 - accuracy: 0.
Epoch 7/10
7588/7588 [==============================] - 8s 1ms/step - loss: 0.2230 - accuracy: 0.9378
Epoch 8/10
7588/7588 [==============================] - 8s 1ms/step - loss: 0.2032 - accuracy: 0.9383
Epoch 9/10
7588/7588 [==============================] - 8s 1ms/step - loss: 0.1978 - accuracy: 0.9425
Epoch 10/10
7588/7588 [==============================] - 8s 998us/step - loss: 0.2050 - accuracy: 0.9441

<keras.callbacks.History at 0x177de41fcd0>
```

```
1 scores = model.evaluate(test_x, test_y)
2 print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
204/204 [==============================] - 0s 810us/step - loss: 0.1463 - accuracy: 0.9565

accuracy: 95.65%
```

Week 13

Developed an alternate version of the machine learning model which balances the data, the results from the previous model were heavily biased towards one or two signs which had more data associated with them. Balancing data strips all the sets of data for the different signs so they all have the same amount of data, making the model more accurate and consistent. The accuracy was still about 95% which shows some overfitting however manual tests I have done give much better results.

Week 14

Learned more about pre-processing and decided to explore some different options, three more models will be made and in total I will have four different models:

- no processing, balanced data.
- balanced data with normalisation.
- balanced data with standardisation.
- Balanced data using data sampling

Week 15

Created the normalisation and standardisation models, which yielded very disappointing results, both had almost 99% accuracy when fitting the model but neither could recognise signs when given new data.

<u>Normalisation</u>

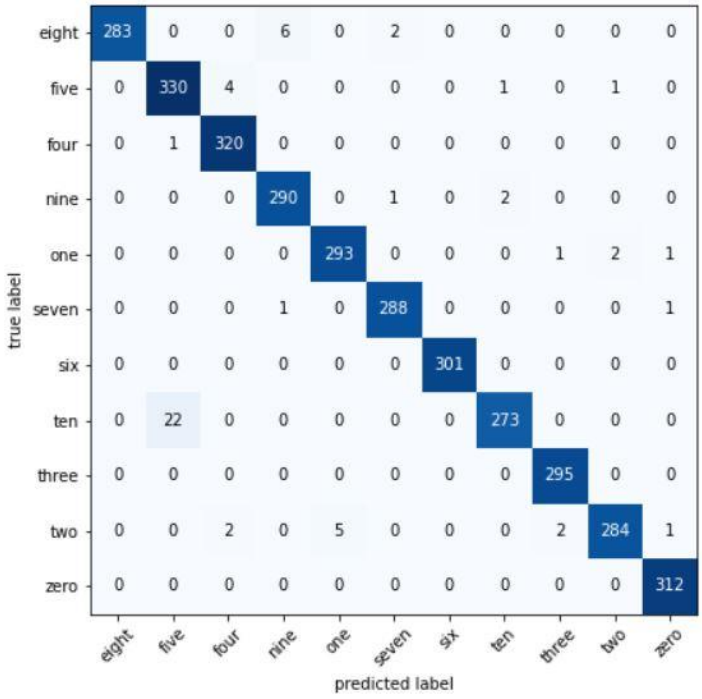
```
1 model.fit(train_x, train_y, epochs = 10, batch_size = 2)
```

```
Epoch 1/10
6648/6648 [==============================] - 6s 878us/step - loss: 0.6378 - accuracy: 0.8147
Epoch 2/10
6648/6648 [==============================] - 6s 888us/step - loss: 0.2072 - accuracy: 0.9533
Epoch 3/10
6648/6648 [==============================] - 6s 903us/step - loss: 0.1518 - accuracy: 0.9661
Epoch 4/10
6648/6648 [==============================] - 6s 912us/step - loss: 0.1310 - accuracy: 0.9717
Epoch 5/10
6648/6648 [==============================] - 6s 946us/step - loss: 0.1156 - accuracy: 0.9745
Epoch 6/10
6648/6648 [==============================] - 6s 973us/step - loss: 0.1032 - accuracy: 0.9773
Epoch 7/10
6648/6648 [==============================] - 6s 923us/step - loss: 0.1049 - accuracy: 0.9752
Epoch 8/10
6648/6648 [==============================] - 6s 910us/step - loss: 0.0979 - accuracy: 0.9773
Epoch 9/10
6648/6648 [==============================] - 6s 873us/step - loss: 0.0969 - accuracy: 0.9765
Epoch 10/10
6648/6648 [==============================] - 6s 933us/step - loss: 0.0880 - accuracy: 0.9783

<keras.callbacks.History at 0x177ee658730>
```

```
1 scores = model.evaluate(test_x, test_y)
2 print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
104/104 [==============================] - 0s 951us/step - loss: 0.0861 - accuracy: 0.9795

accuracy: 97.95%
```

```
(<Figure size 504x504 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



Standardisation

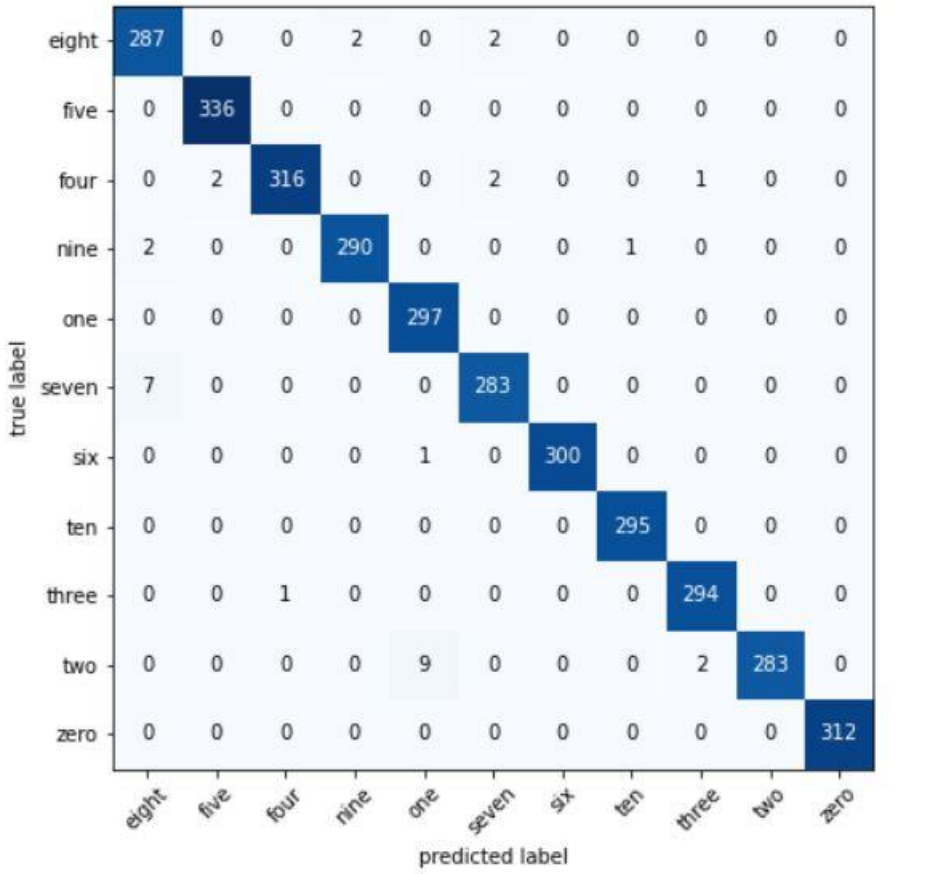
```
model.fit(train_x, train_y, epochs = 10, batch_size = 2)
```

```
Epoch 1/10
6648/6648 [==============================] - 7s 917us/step - loss: 0.2568 - accuracy: 0.9240
Epoch 2/10
6648/6648 [==============================] - 6s 881us/step - loss: 0.0428 - accuracy: 0.9893
Epoch 3/10
6648/6648 [==============================] - 6s 928us/step - loss: 0.0347 - accuracy: 0.9906
Epoch 4/10
6648/6648 [==============================] - 6s 910us/step - loss: 0.0257 - accuracy: 0.9924
Epoch 5/10
6648/6648 [==============================] - 6s 919us/step - loss: 0.0231 - accuracy: 0.9926
Epoch 6/10
6648/6648 [==============================] - 6s 879us/step - loss: 0.0204 - accuracy: 0.9935
Epoch 7/10
6648/6648 [==============================] - 6s 863us/step - loss: 0.0197 - accuracy: 0.9944
Epoch 8/10
6648/6648 [==============================] - 6s 934us/step - loss: 0.0161 - accuracy: 0.9948
Epoch 9/10
6648/6648 [==============================] - 6s 900us/step - loss: 0.0206 - accuracy: 0.9943
Epoch 10/10
6648/6648 [==============================] - 6s 899us/step - loss: 0.0153 - accuracy: 0.9944

<keras.callbacks.History at 0x177d9bef700>
```

```
scores = model.evaluate(test_x, test_y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
104/104 [==============================] - 0s 1ms/step - loss: 0.0423 - accuracy: 0.9907

accuracy: 99.07%
```

```
(<Figure size 504x504 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



Week 16

No Work on project

Week 17

Created Sampling model which yielded similar results to the balancing only model both in the fitting accuracy and the manual tests I have conducted.

Week 18

Decided to collect a new set of data which now has 5 trials of 20 seconds each, making the dataset much bigger and might allow me to get better real-world accuracy.

Re-created all the models with the new data set, from manual testing it is giving me slightly better results, with the standardisation and normalisation models still being unusable. I will create an automated testing script which will write a file for me and give me an overview of each system's accuracy and which classes/signs they struggle with.

Week 19, 20, 21, 22, 24

No work on project as I had some problems with other modules and was focussing on them, I also received an extension until 8th of June.

```
================================================
                MODEL TESTING
================================================


Overall model accuracy
==============
correct: 75.45 %
incorrect: 24.55 %

Predicted labels
==============
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: zero - True class: zero
Predicted class: one - True class: one
Predicted class: three - True class: one
Predicted class: three - True class: one
Predicted class: one - True class: one
Predicted class: one - True class: one
Predicted class: one - True class: one
Predicted class: one - True class: one
Predicted class: one - True class: one
Predicted class: three - True class: one
Predicted class: one - True class: one
```

Week 25

Created the testing script which will look through a folder and run the specified model on it to make a prediction then write details of it to a file, also calculating the overall model accuracy and the model's accuracy in predicting specific classes, ranking them

This example was done with new data testing on the balancing only model.

```
Per class rankings
==============
Class: six - correct: 0.00 % - incorrect: 100.00 %
Class: seven - correct: 30.00 % - incorrect: 70.00 %
Class: one - correct: 70.00 % - incorrect: 30.00 %
Class: two - correct: 70.00 % - incorrect: 30.00 %
Class: ten - correct: 80.00 % - incorrect: 20.00 %
Class: eight - correct: 90.00 % - incorrect: 10.00 %
Class: nine - correct: 90.00 % - incorrect: 10.00 %
Class: zero - correct: 100.00 % - incorrect: 0.00 %
Class: three - correct: 100.00 % - incorrect: 0.00 %
Class: four - correct: 100.00 % - incorrect: 0.00 %
Class: five - correct: 100.00 % - incorrect: 0.00 %
```

Week 26

Using my tests I decide to opt for the balancing only model. I updated my data collection module to now capture a sign from the user, save the sensor file and automatically run the model on the data to output a translation.

Week 27

Polished my application and models, organising my project in a suitable manner, and completing the project.

Week 28

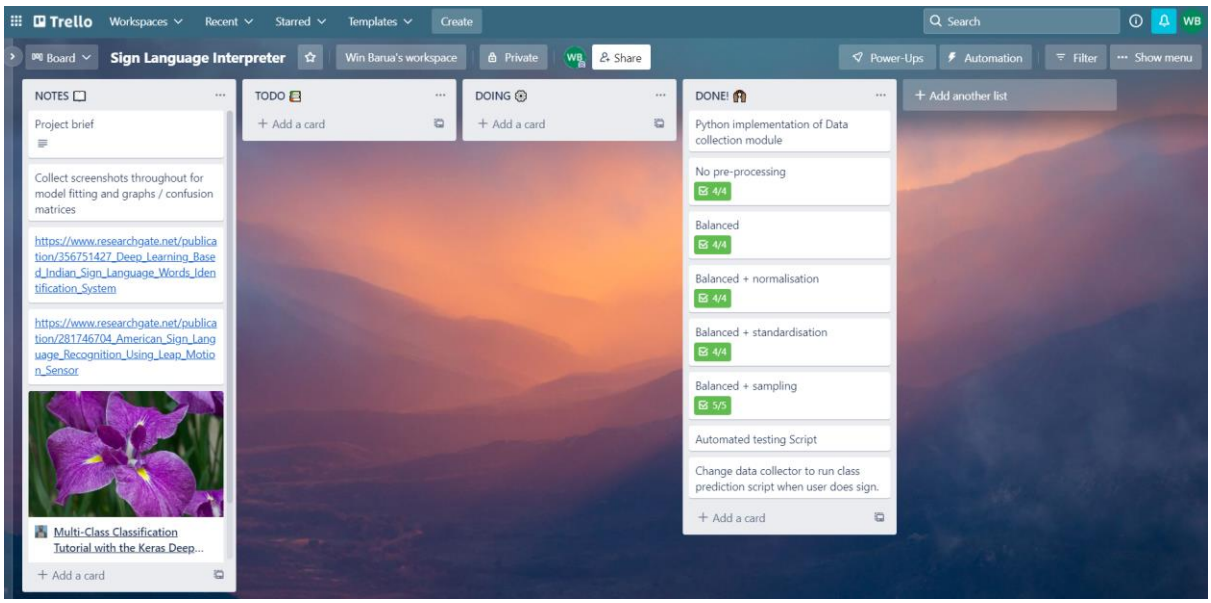
Start of my Final Report

Week 30

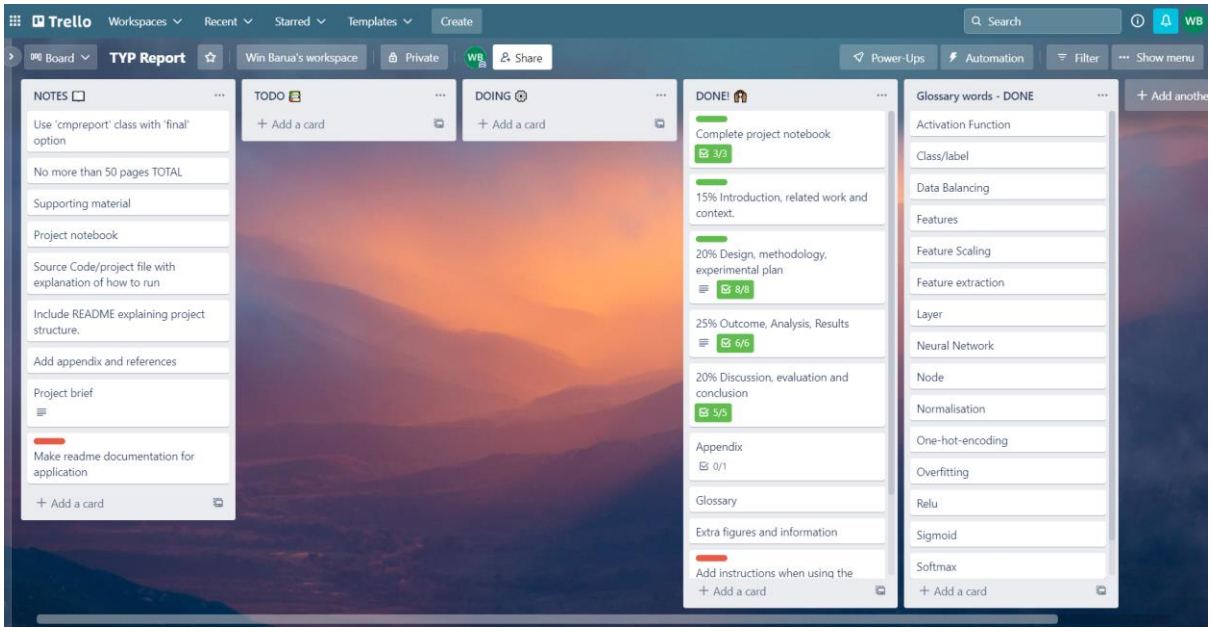
Finished report and made a short presentation for the start of the demo, also made a readme file with more detailed information about using the application, similar to a user documentation.

Extra information

Trello used for keeping track of my tasks

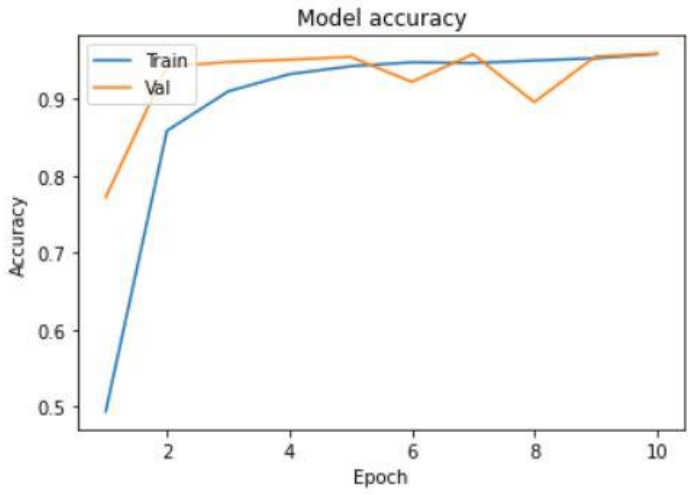


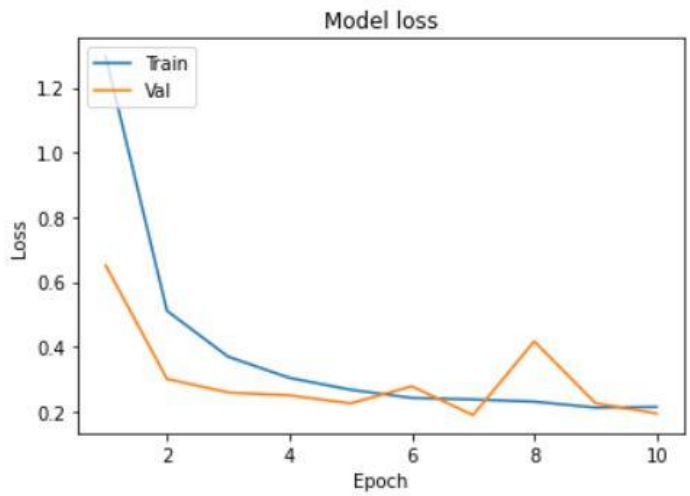
Trello used to keep track of final report task



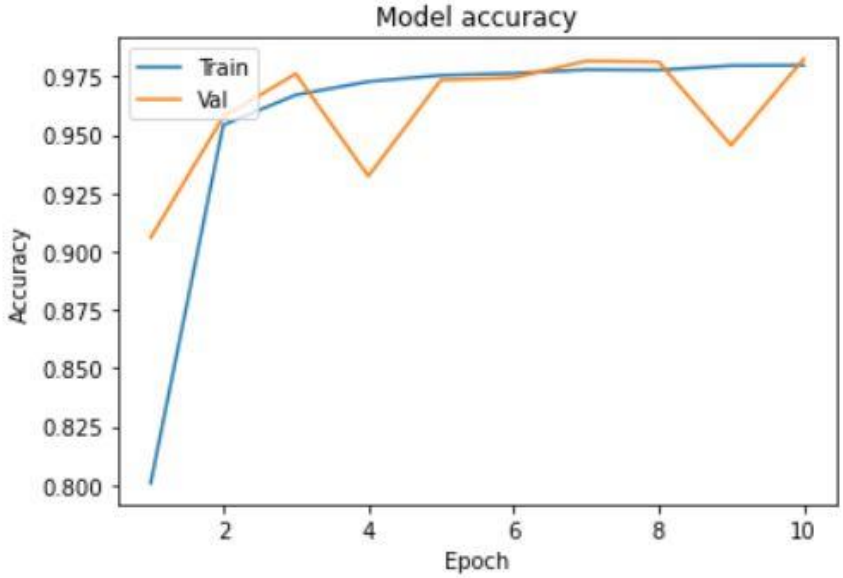
Model accuracy and loss graphs

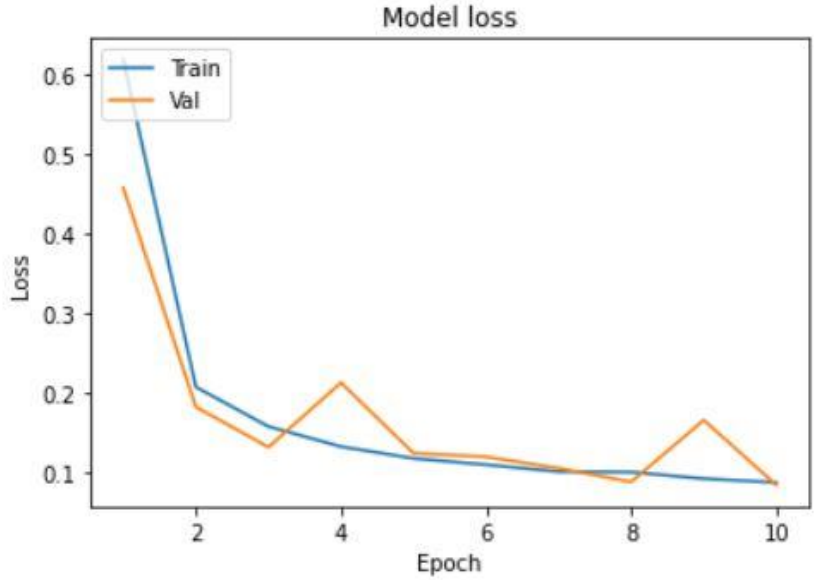
Balancing only



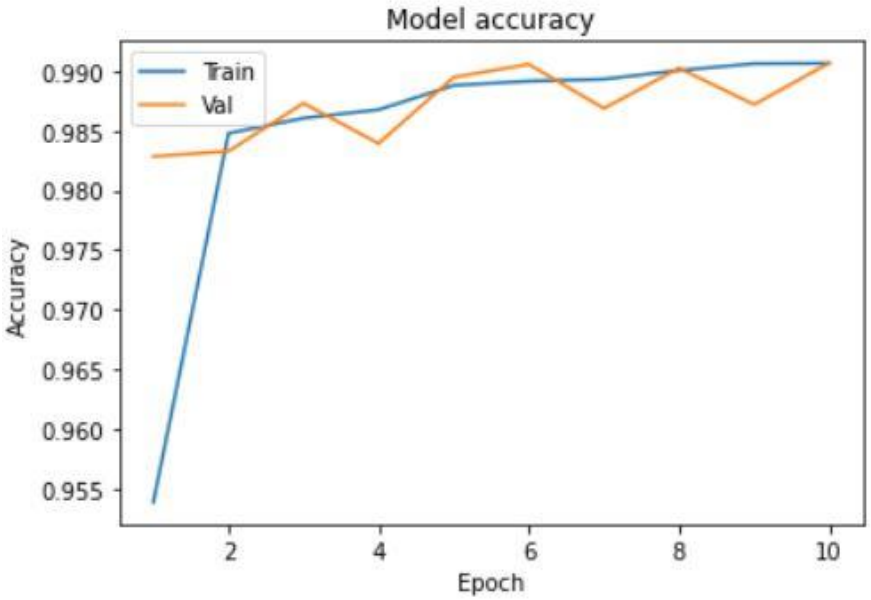


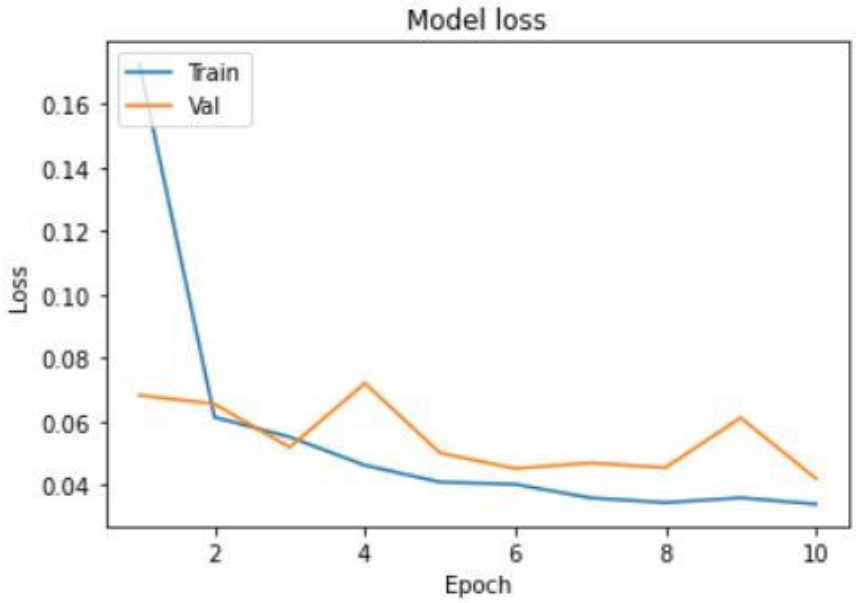
Normalisation



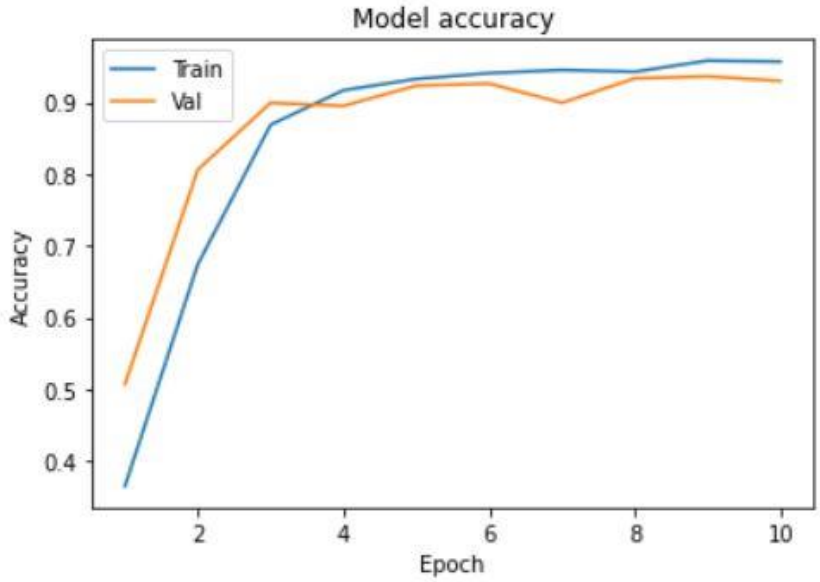


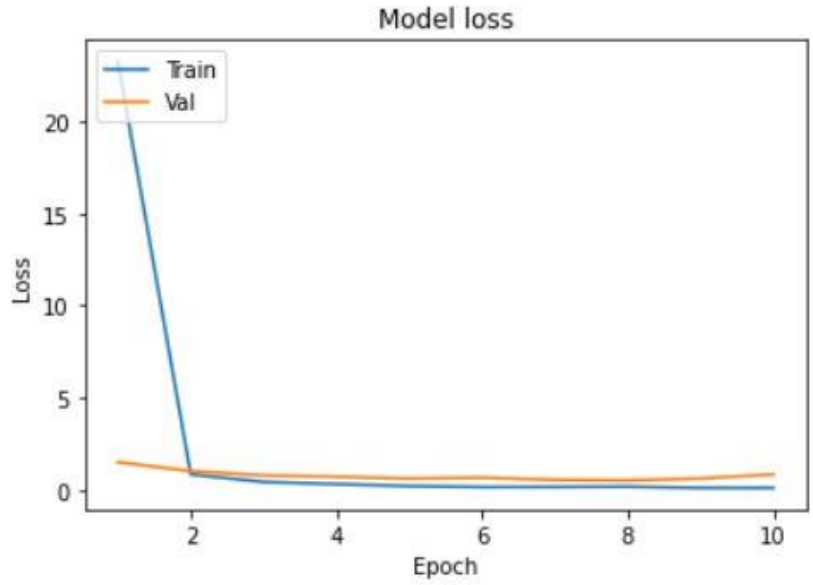
Standardisation





Sampling


Model accuracy
Train
Val
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
2
4
6
8
10
Epoch


Model loss
Train
Val
Loss
20
15
10
5
0
2
4
6
8
10
Epoch