Deep Learning
Assignment 1 Part B

Ronik Jayakumar
24680264

Objective:

A dataset is worked on that is composed of 70000 images of handwritten Japanese Hiragana characters. The overall target variables, which in this case consist of image labels, are composed of 10 different classes. The objective of the experiment is to create a custom neural network that can identify the characters and assign them to their corresponding labels. To achieve efficient results, 3 experiments are to be performed to train neural networks with varying architectures. The goal is to achieve an accuracy of at least 80% while avoiding overfitting of the data.

Methodology:

Data Loading:

- The data was provided from the Japanese MNIST dataset because of which conventional data cleaning steps such as checking for null values, outliers, etc. have been avoided.
- The data has been made available as NPZ files which are zipped archives or NumPy arrays.
- The image width and height have been defined at 28 each. There are 10 overall target variables.

Data Preprocessing:

The steps taken to prepare the data for training and testing are as follows.

- The training and testing data is **reshaped** to the dimensions (row_number, height, width, channel), followed by conversion to float decimals. To be specific, float32 is the choice of data type to increase processing speed.
- Using the image width and height, the input dimensions are defined as 28*28 = **784**
- Since each image consists of 255 pixels, standardisation is performed by dividing the training and testing data by 255.
- **One hot encoding is performed on the target variables** so as to be able to associate them with the training and testing data.
- The batch sizes for data training have been set at 128 and the number of epochs for training the neural network has been set to 500.

Now our data is ready for analysis.

Experiment 1 (SGD Optimiser):

Hypothesis:

Test the efficiency of a 4 layer Perceptron using the ReLU activation function and the SGD optimiser.
The code implements a simple Multi-Layer Perceptron (MLP) neural network for image classification using PyTorch. The architecture consists of the following specifications:

Architecture:

- Input Layer: A linear layer with input dimension input_dim (784 in this case, corresponding to 28x28 image dimensions).
- Hidden Layer 1: A linear layer with 350 neurons followed by the ReLU activation function.
- Hidden Layer 2: A linear layer with 100 neurons followed by the ReLU activation function.
- Output Layer: A linear layer with num_classes neurons (10 in this case, corresponding to the number of classes in the dataset).

Activation function:

```
CustomMLP(
  (layer1): Linear(in_features=784, out_features=350, bias=True)
  (layer2): Linear(in_features=350, out_features=100, bias=True)
  (layer3): Linear(in_features=100, out_features=10, bias=True)
)
```

The **rectified linear unit (ReLU)** or rectifier activation function introduces the property of nonlinearity to a deep learning model and solves the vanishing gradients issue. It interprets the positive part of its argument. It is one of the most popular activation functions in deep learning. Due to this reason, this is the activation function that has been used in all the three experiments. [1]

Cost Function and Optimiser:

The **SGD, or the stochastic gradient descent** optimiser has been selected. In SGD, instead of processing the entire dataset during each iteration, we randomly select batches of data. This implies that only a few samples from the dataset are considered at a time, allowing for more efficient and computationally feasible optimisation in deep learning models. To increase the speed of processing, a **momentum algorithm** is implemented into the SGD optimiser which helps in faster convergence of the loss function. [2]

Training and Testing:

The training and testing losses and accuracies are calculated separately in Experiment 1.

The training loop consists of the forward and the backward passes. This has been iterated over the defined 500 epochs.
The testing loop for experiment 1 is directly a calculation of accuracy and has not been done over the defined 500 epochs.

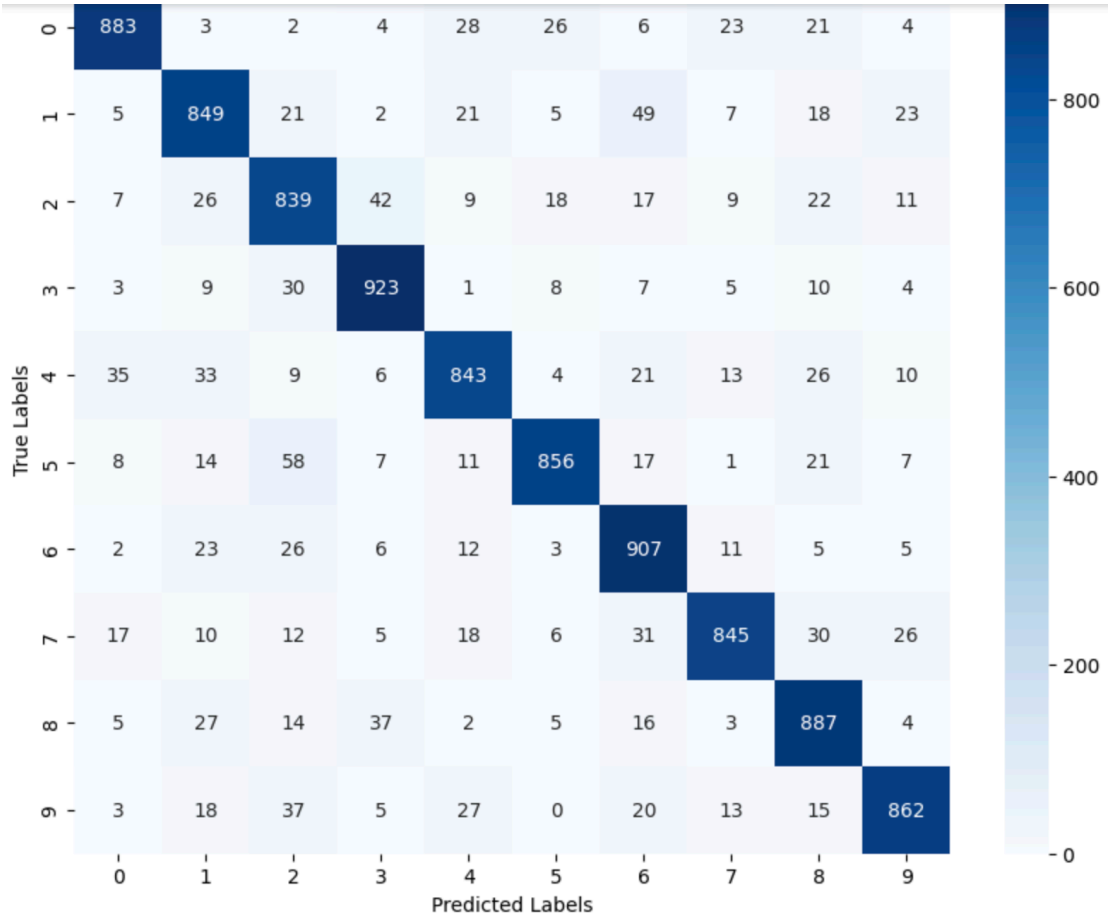
The final figures were as follows:
Training accuracy - 96.74%
Testing Accuracy - 86.94%

Training loss- 0.010
Testing Loss - 0.0035

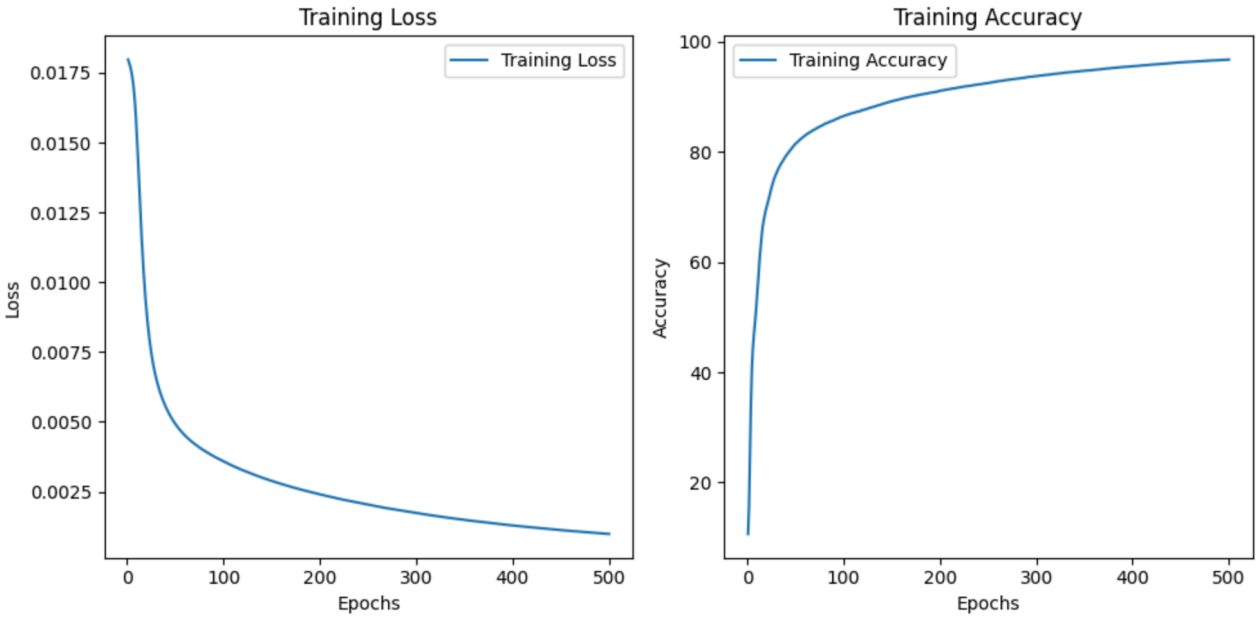
Conclusion:

The graph and the confusion matrix give us pictorial evidence on the model's performance where we can see the labels were predicted accurately in a large number of cases. The high accuracy rates tell us that the deep learning model worked well in identifying a majority of the pictures and assigning it to the correct target label.
The confusion matrix shows us that label 3 was the best predicted label with 923 correct predictions while label 2 was the worst, still at a high number of 839. This shows us a small imbalance that exists within the class which is something that can be worked on in the future.

Confusion Matrix:



Training Loss and Accuracy Plot:



Experiment 2 (AdaGrad Optimiser):

Hypothesis:

Experiment 2 is used to check if a change in optimiser proves variations in losses and efficiency. The optimiser is changed from SGD to AdaGrad.

Architecture:

The model architecture for experiment 2 has been kept the same as experiment 1 with the input layer, 2 hidden layers, and the output layer, all using the **Rectified Linear Unit (ReLU)** Activation function

```
CustomMLP2(
  (layer1): Linear(in_features=784, out_features=350, bias=True)
  (layer2): Linear(in_features=350, out_features=100, bias=True)
  (layer3): Linear(in_features=100, out_features=10, bias=True)
)
```

Cost Function and Optimiser:

With the Cross Entropy Loss function as the cost function, the SGD optimiser has been replaced with the AdaGrad optimiser. The reason for exploration into the AdaGrad optimiser is because of its feature where it abolishes the need to modify the learning rate manually. It is more reliable than gradient descent algorithms and their variants, and it reaches convergence at a higher speed. [2]

Training and Testing:

The training and testing process for experiment 2 has been carried out together with all 500 epochs running on both the training and the testing code together. A linear increase in accuracies and decrease in losses has been observed in using the AdaGrad optimiser. The final features were as follows:

Training accuracy - 97.90%
Testing Accuracy - 87.30%

Training Loss - 0.0007
Testing Loss - 0.0032

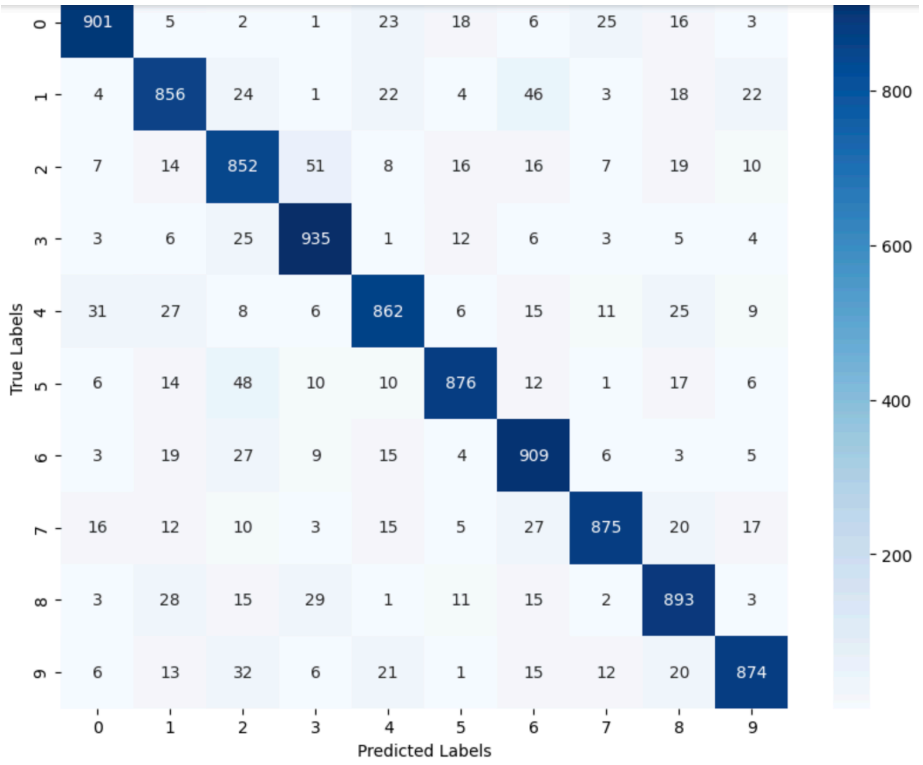
Conclusion:

In conclusion, the AdaGrad Optimiser did a slightly better job in predicting data. The accuracy percentage remained high throughout the evaluation process across the epochs and the final testing value was 0.4% higher than SGD, which in large datasets proves to be a massive increase.

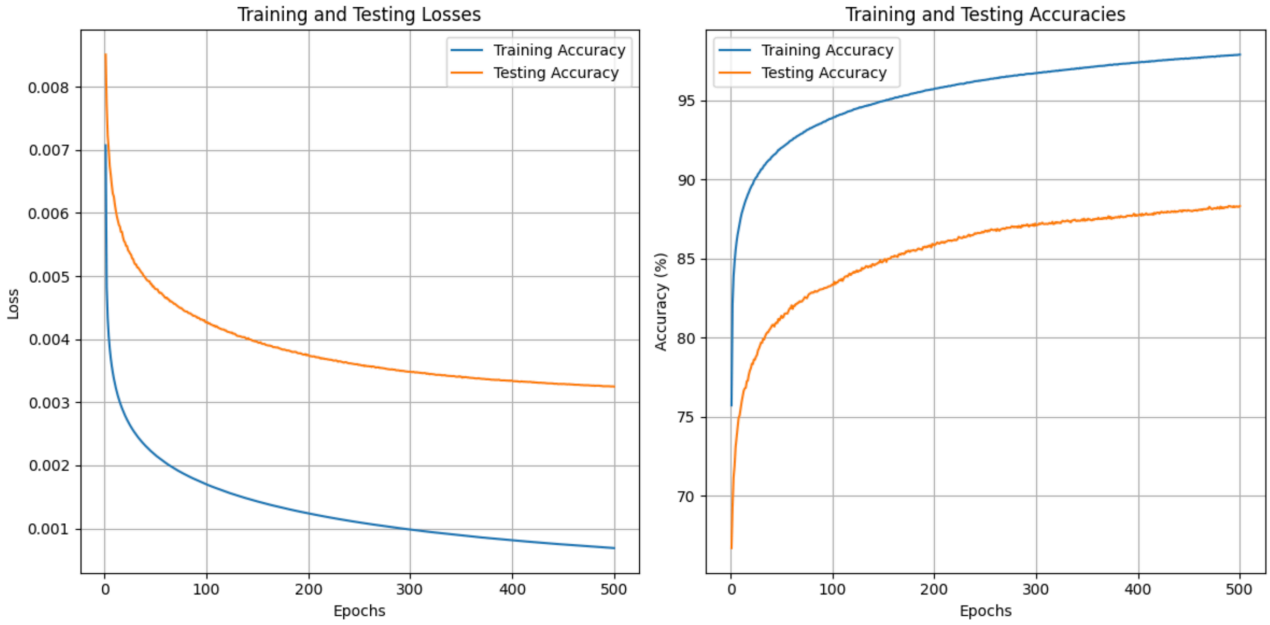
Again, with the confusion matrix we can see that the best predicted Label was label 3 and the worst continued to be label 2 which gives weight to the class imbalance theory.

A graph of the training and testing losses and the training and testing accuracies were plotted to get a pictorial representation of the same.

Confusion matrix:



Training and testing Loss vs Accuracy:



Experiment 3 (SGD with 5 layers + Dropout layer)

Hypothesis:

Experiment 3 goes back to the SGD optimiser to check if increasing the number of layers and adding a dropout layer adds to the accuracy of the model.

Architecture:

For the 3rd and final experiment, an architecture of 1 input layer, 5 hidden layers, a dropout layer, and an output layer has been defined. The same Rectified Linear Unit has been used as the activation function.

The overall architecture is as given below:

```
CustomMLP3(
  (layer1): Linear(in_features=784, out_features=512, bias=True)
  (layer2): Linear(in_features=512, out_features=350, bias=True)
  (layer3): Linear(in_features=350, out_features=100, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (layer4): Linear(in_features=100, out_features=10, bias=True)
)
```

Cost Function and Optimiser:

The Cross Entropy Loss Cost function has once again been associated with the SGD optimiser function with extra hidden layers in a bid to see if the extra layers along with the dropout layer increases its accuracy beyond the AdaGrad optimiser.
As done inn experiment 1, the momentum has been set to 0.9 to help **accelerate gradients vectors in the right directions in turn facilitating faster converging. [3]**

Training and Testing:

The training and testing architecture has been kept the same as experiment 2 with the training loop consisting of a Forward and Backward pass and the testing loop being the evaluating loop. Again, both training and testing data has been run over the entirety of the 500 defined epochs with a batch size of 128.
A key point to note here is the increase in processing time due to the increase in the number of networks.

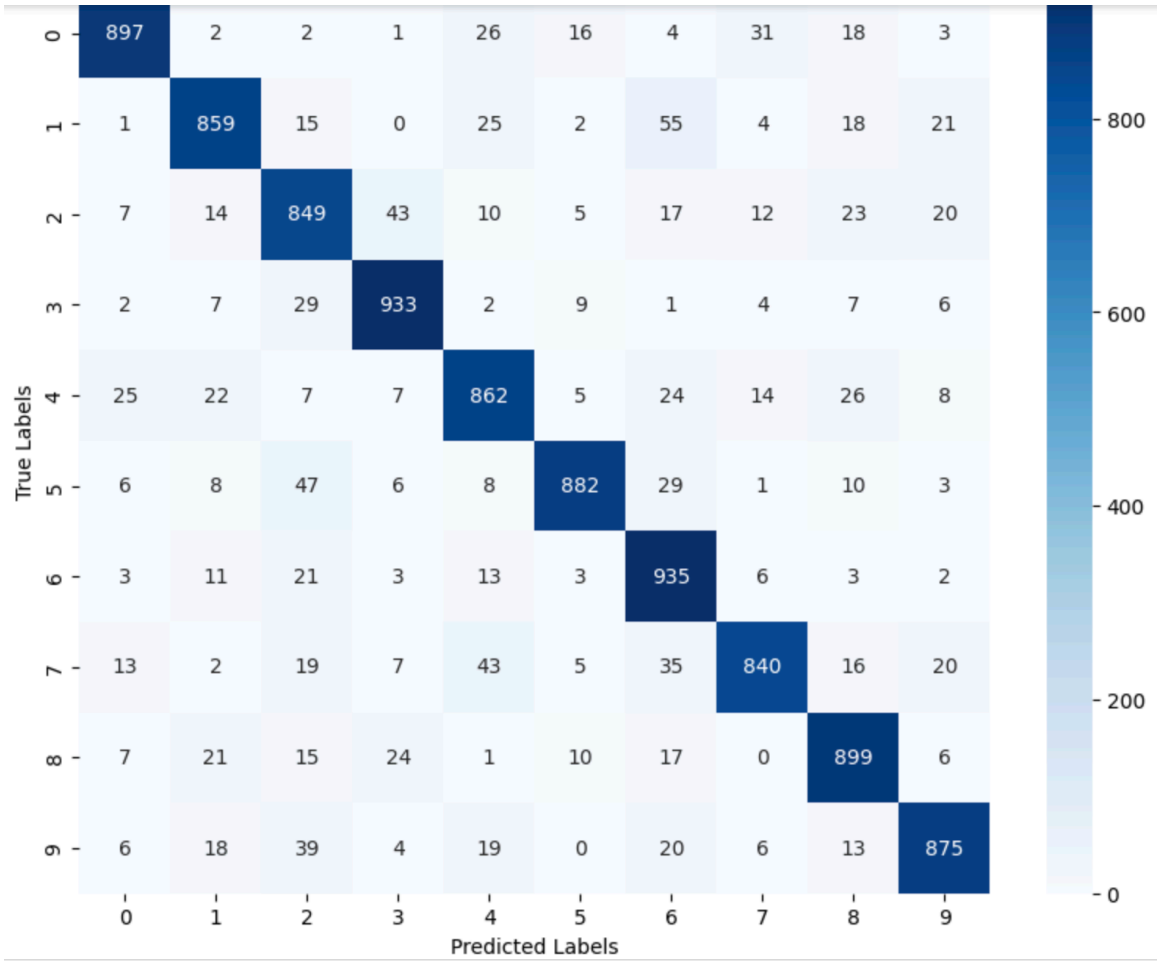
The final results from training and testing were as follows:
Training Loss - 0.0007
Testing Loss - 0.0033

Training Accuracy - 97.44%
Testing Accuracy - 88.31%

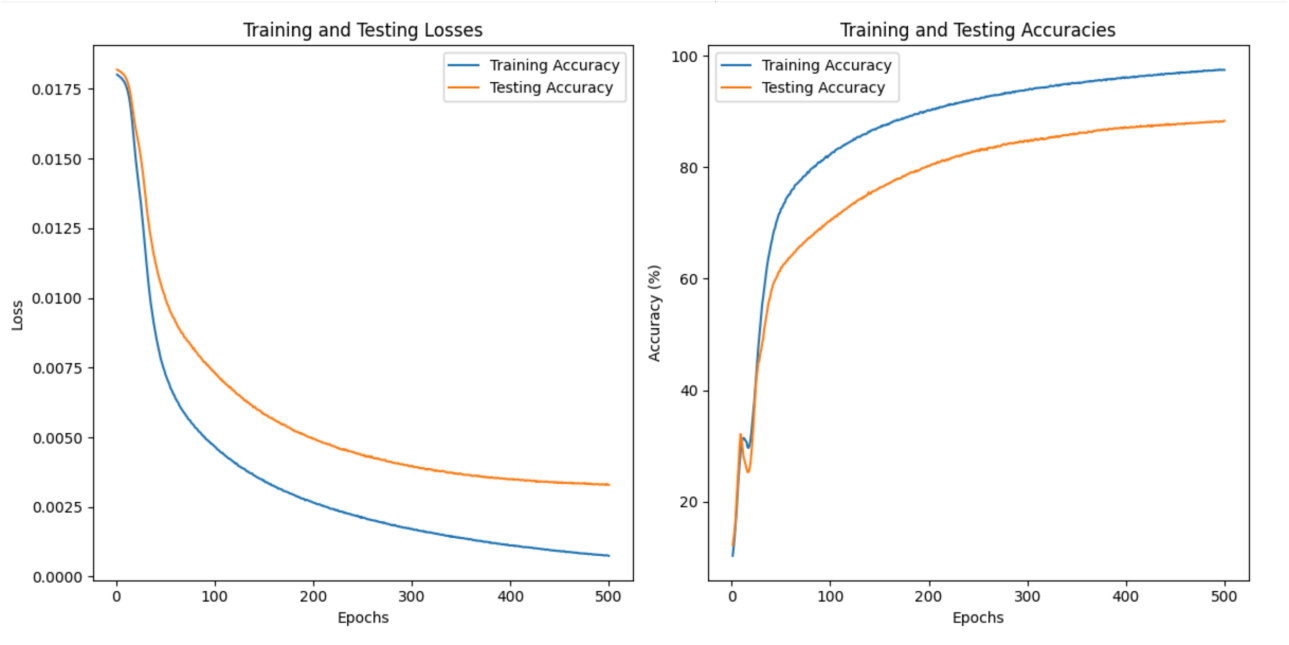
Conclusion:

The overall accuracies remain fairly high like the other two experiments. We can also see an increase in accuracy in the training and testing set as compared to experiment 1 which proves that the combination of increase in number of networks along with the addition of a dropout layer did increase the models accuracy.

Confusion Matrix:



Training and testing loss and accuracy graph:



Current limitations and Future Recommendations:

The current Multi Layer Perceptrons that have been implemented show signs of working but there exists a large scope of improvement.

1. **Fitting:** There exists a difference of about 9% to 11% between the accuracies of the training and testing data. This indicates room for improvement by the use of Regularisation techniques like L2 regularisation and early stopping. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset [4]

2. **Use of Convolutional Neural Networks:** Convolutional Neural Networks or CNNs for short, are used for finding patterns in images to recognise objects. They can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object. [5]

3. **Data Augmentation** - The use of various data augmentation techniques can help improve the models accuracy by training it to deal with variations in the testing data. It does this by performing tasks on the training images such as Geometric Transformations, Colour Space Transformations, Kernel filters. Random erasing, mixing images, etc. [6]

4. **Cross Validation techniques** - The use of cross validation techniques such as k fold can help the model be better equipped at handling unseen testing data and avoid overfitting. [7]

5. **Model Evaluation techniques** - Increasing and investigating various model evaluation techniques apart from accuracy such as F1 score, precision, and recall could uncover further insights into the models performance. [7]

Reference:

1. An introduction to the ReLU activation function - Bharath Krishnamurthy - Builtin - https://builtin.com/machine-learning/relu-activation-function

2. A comprehensive guide on Optimisers in Deep Learning - Ayush Gupta - Jan 23rd 2024 - Analytics Vidhya - https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

3. Stochastic Gradient Descent with Momentum - Vitaly Bushaev - Dec 5th 2017 - Towards Data Science - https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d

4. Use Early Stopping to Halt the Training of Neural Networks At the Right Time - Jason Brownlee - Aug 25 2020 - Machine Learning Mastery - https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

5. What is a Convolutional Neural Network - Mathworks - https://au.mathworks.com/discovery/convolutional-neural-network.html

6. A complete guide to data augmentation - Nov 2022 - Datacamp - https://www.datacamp.com/tutorial/complete-guide-data-augmentation

7. What are the best practices for evaluating the performance of a deep learning model for image classification? - https://www.linkedin.com/advice/3/what-best-practices-evaluating-performance-deep#:~:text=The%20first%20step%20in%20evaluating,%2Dscore%2C%20and%20confusion%20matrix.

8.