CS4287 - Assignment 2

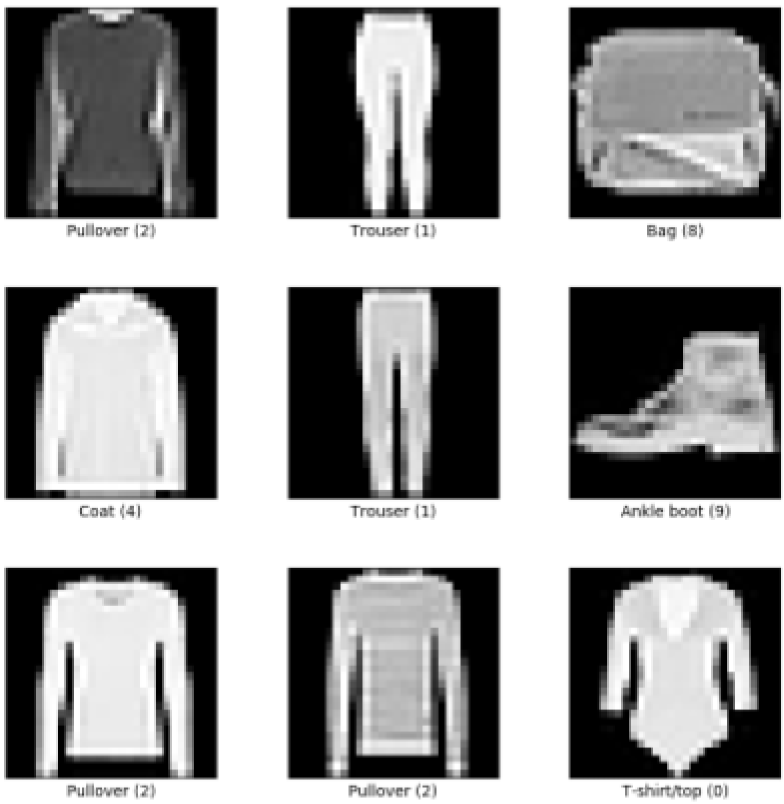
Alannah Ryan - 18232132

Dataset: https://www.tensorflow.org/datasets/catalog/fashion_mnist

The primary focus of my report is to implement a Convolutional Neural Network (CNN) based on popular CNN architectures.

The LeNet architecture is an excellent “first architecture” for Convolutional Neural Networks. It is small and easy to understand, yet large enough to provide interesting results. Furthermore, the combination of LeNet and MNIST is able to run on the CPU, making it easy for beginners.

The chosen dataset is fashion_mnist. This is a dataset of Zalando’s article images consisting of a training set of 60,000 example and a test set of 10,000 examples. Each example is a 28x28 greyscale image, associated with a label from 10 classes.



The Network Structure, Distinguishing Features, Weight Initialisation, Batch Normalisation, Regularisation, Transfer Learning & Other Hyperparameters.

The network structure of this CNN uses the LeNet-5 Architecture. This consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully connected layers and finally a softmax classifier.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Batch normalization is a layer that allows every layer of the network to do learning more independently. It is used to normalize the output of the previous layers. The activations scale the input layer in normalization.

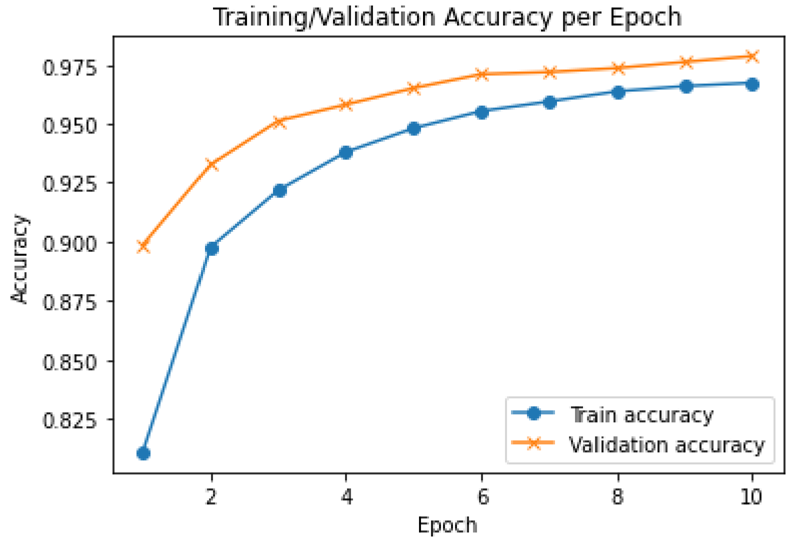
```
# Normalization #
model.add(BatchNormalization())
```

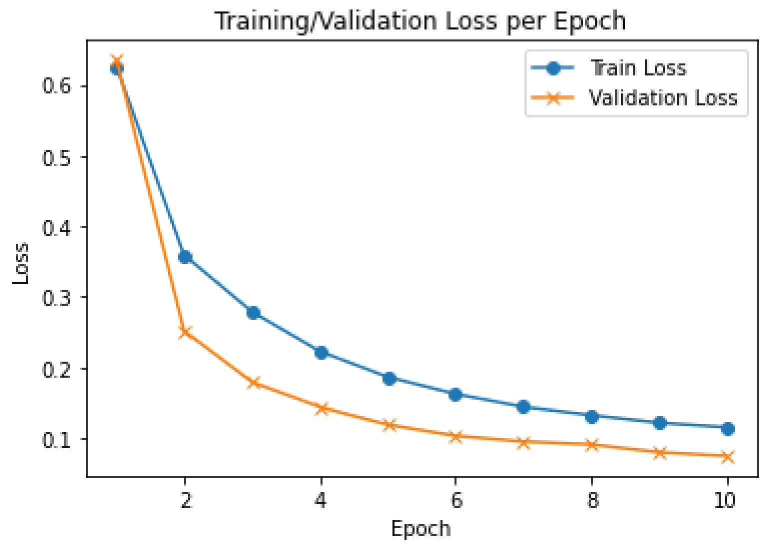
```
# Normalize value to [0, 1] #
# This is to get the mean as 0 and the standard deviation as 1, which will reduce training time #
x_train /= 255
x_test /=255
```

Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.

```
# Dropout Regularization #
model.add(keras.layers.Dropout(0.5))
```

Results





Evaluation of the Results

The Convolutional Neural Network had to be tested many times and hyperparameters had to be changed in order to see different results. I saw within these results that the accuracy changed drastically if you change some of the hyperparameters even the slightest and then with others the accuracy stayed the same. With each experiment, I decided to go for the approach that gave the best accuracy.

I kept the Epochs at 10 as this provided the most accurate results. This resulted in the accuracy percentage being approximately 97% with a loss of only 0.0751%.

Impact of Varying Hyperparameters

The impact of the varying hyperparameters was that we first experimented with the epochs equal to 10.

From this we can see that varying hyperparameters can have a huge effect on the accuracy of the dataset and so it is important to choose the correct number for your hyperparameters when testing the Convolutional Neural Network.

References

https://www.tensorflow.org/datasets/catalog/fashion_mnist

https://github.com/zalandoresearch/fashion-mnist

https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342

https://www.datasciencecentral.com/profiles/blogs/lenet-5-a-classic-cnn-architecture