# Assignment 1B

## Problem 1

### CNN Model Design choices

For the first two parts (with and without data augmentation) of problem 1 used a convolution neural network made from scratch but the design is heavily influenced from the models used in the lectures and practicals. The layers of this model can be seen below in figure 1. The design of this model can be separated into 2 parts being made up of being your classic feedforward neural network and convolution/pooling parts. The convolution/pooling parts of the model are implemented directly after the first input layers (being of shape 32,32,3 meaning it is a 32,32-pixel image with 3 layers representing colour) and is repeated 3 times. The purpose of this layer is to compress the data down to a more manageable size and filter key features. Each of these 3 parts are made of the structure dot pointed below:

* 2x convolutional layers
* Normalisation layers
* Dropout layer
* Pooling layers

Each of these layers start with 2 convolutional layers with the purpose of extracting and filtering information from any sample. For the first pair of convolutional layers, it is design to produce 32 layers of dimensionality and this value is doubled per convolution/pooling parts. This is done in the hope that over each layer more combinations or patterns will be captured. A kernel size of (3,3) was chosen because the image itself is quite small being (32,32) so a large kernel size was not required.

After the 2x convolutional layers a layer dedicated to normalising the outputs of them is used. This layer is used to return values to a common scale after the data has been changed by the previous layers.

After a dropout layer is placed. This layer is used as removing some data at random may help to prevent overfitting to the training data.

The last part of each convolution/pooling parts is a pooling layer. Each of these layers have a pooling size of 2x2 each time the layer is used the dimension of the image is reduced by half. This is used to summarise features and reduce noise.

The final quarter of the model is made up of a feedforward section. To allow this, a layer is used to flatten the results of the previous convolution/pooling parts. This part of the model is made up of 5 layers where all but last layer uses the ‘relu’ activation function as it is a linear function and as I have no clue what the model will find a linear activation seemed like a reasonable place to start. The last layer has no activation function. The last layer was given no activation function as with trial and error it was found that without one the model would produce better results. The first 3 layers are of 128 neurons wide. Through trial and error, it was found that increasing the number of 128-layers would produce better results however as explained latter 3 seemed like a good balance between training time and accuracy. The next layer is a layer containing 64 neuron and is used to reduce the number of neurons gradually before entering the final layer which is made up of 10 neurons as there are 10 classes.

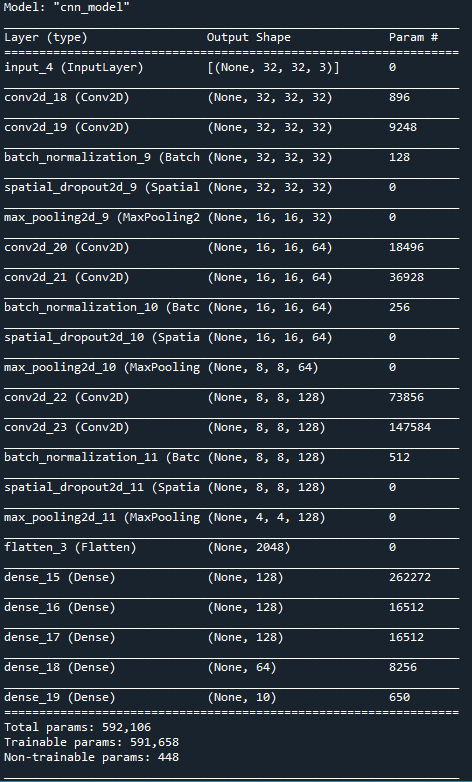


Figure : CNN model layers

### Loss Function

The loss function to be used for the models was the sparse\_categorical\_crossentropy loss function. This function was chosen as each sample will only belong to 1 of the classes where in similar functions samples may belong to multiple classes. This function was also chosen as labels came in the form of integers.

### Dataset

The dataset for this problem consisted of 1000 labelled images for training the 10000 for testing. It should be noted as seen in figures 3 to 8 that both training and testing datasets do not have the same amount of training and testing data for all possible labels. The data seems to follow the trend that samples labelled with 2 will have 80% the number of samples that samples labelled with one will have this will continue as samples labelled with 3 will have 80% of samples when compared to samples labelled with 2. This is significant as this bias in training data could be translated into the models.

### Data Augmentations

The data augmentations used only a few parameters were chosen to change/alter images in the training space. This is because in each of the samples used for training and testing the number was always clearly in the centre of the image and other numbers could be present in a sample therefore the location of the number could not be changed to much. For simplicity 4 parameters were chosen to alter being zoom, height and small changes to rotation and shear\_range as seen in table 1 below. Each of these parameters was chosen as they do not change the position (being the centre of image) of the labelled number except for height\_shift\_range. Height\_shift\_range was used as there was no images found in the training set with numbers above or below the labelled image as numbers are written left to right or horizontally.

Table 1: Data Augmentation used on Data

|  |  |
| --- | --- |
| **Data Augmentation** | **Range** |
| rotation\_rang | 10 degrees |
| zoom\_range | 80 to 120% |
| height\_shift\_range | 10% |
| shear\_range | 15% |

### Computational Constraints

Due to computational constraints the model will be limited to a small model with less then 30 layers. Creating a large model would produce huge training times that would be infeasible to train due to time constraints on the project. However, a high level of accuracy has been achieved with the model detailed above as seen under the comparison heading with the model able to achieve above 80% accuracy on the testing set. It should be noted that with a more complex model (more layers) a higher level of accuracy should be able to be achieved.

### Model description

Three models were used on the training and testing data provided. These being the CNN model detailed above both with and without data augmentations done the training data and a model developed for the CIFAR Dataset.

When trained with no data argumentations a batch size of 40 and 50 epochs was chosen to train the data on the model. These numbers where selected by using trial and error by comparing both testing and training dataset accuracy however a batch size of 40 seemed reasonable as there where only a small amount of training data to use and 50 epochs was chosen as there is only a small amount of data a high value would expose the data too much to the model and could have caused overfitting.

With the model trained on the dataset with data augmentation applied to it the number of epochs was greatly increase to 250 as the data could produce more training data and was more likely to avoid overfitting because of this.

The CIFAR model is named 'vgg\_2stage\_CIFAR\_small.h5 and was chosen because the input image shape and number of outputted classes where the same as the SHVN dataset. This allowed the model to be used for fine tuning on the SHVN dataset with no modification needed on the model itself. The data augmentation where also used on this model and therefore the same number of epochs and batch size where also used, being 40 for batch size and 250 epochs for the same reasons. This was because the models were very similar.

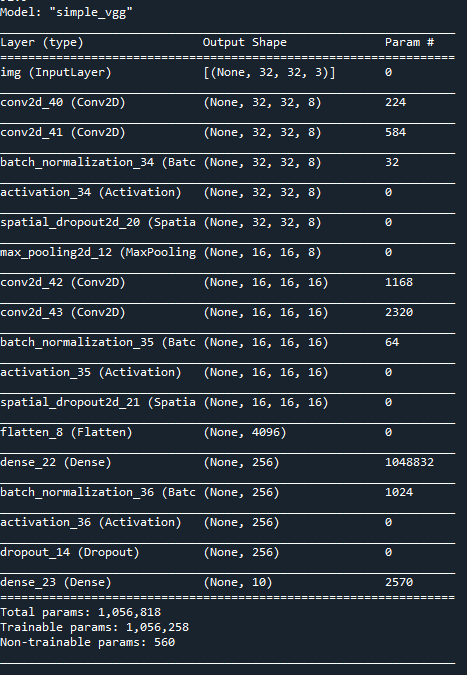


Figure 2: CIFAR model Layers

As seen in figure 2 the CIFAR model is very similar to the model create for this problem above seen under the heading CNN Model Design choices. The difference these models have are that the CIFAR model has 2 less pooling layers and 1 less convolutional pair. It also has an activation layer added between the normalisation and dropout layers however the greatest difference between these models is that where the model created for this project is designed to increase the number of dimensions (filters) further through the layers where the CIFAR layers decrease the number of dimensions the further into the network. It should also be noted that after the layers are flattened in the CIFAR model the widths (number of neurons per layer) of the new model is twice as large as the one created for this project and makes no attempt to decrease the layers gradually when close to the output layer.

### CNN with no Data Augmentation Results

The results for the CNN model with no data augmentation used can be seen in figures 3 for the training data and figure 4 for the testing data. The overall accuracy of the model can be seen below under model comparison on both the training and testing data given in table 2 which shows the percentage of correct samples classified being 100% on the training images and 82.66% on the testing dataset.

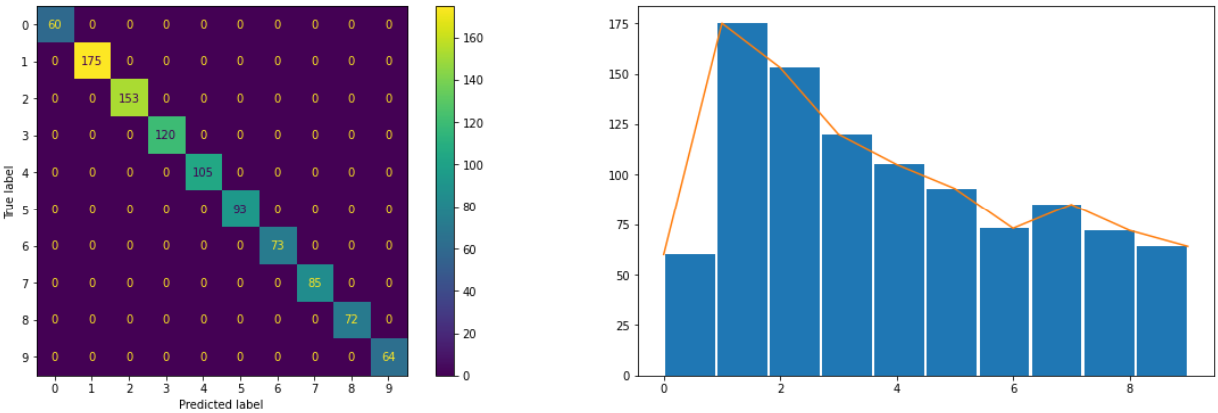


Figure 3: CNN with no Data Augmentation Training Dataset

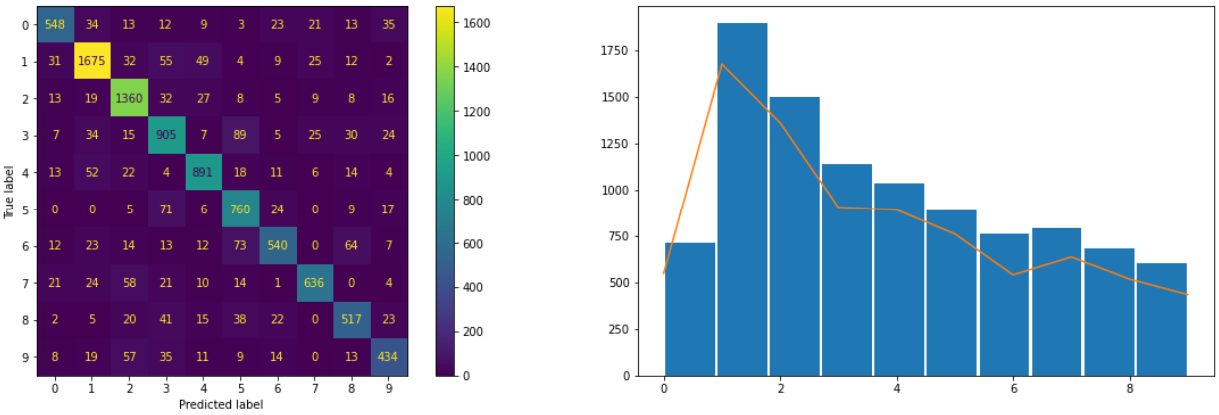


Figure 4: CNN with no Data Augmentation Testing Dataset

### CNN Data Augmentation Results

The results for the CNN model with data augmentation used can be seen in figures 4 for the training data and figure 5 for the testing data. The overall accuracy of the model can be seen below under model comparison on both the training and testing data given in table 2 which shows the percentage of correct samples classified. The overall accuracy of the model can be seen below under model comparison on both the training and testing data given in table 2 which shows the percentage of correct samples classified being 100% on the training images and 83.75% on the testing dataset.

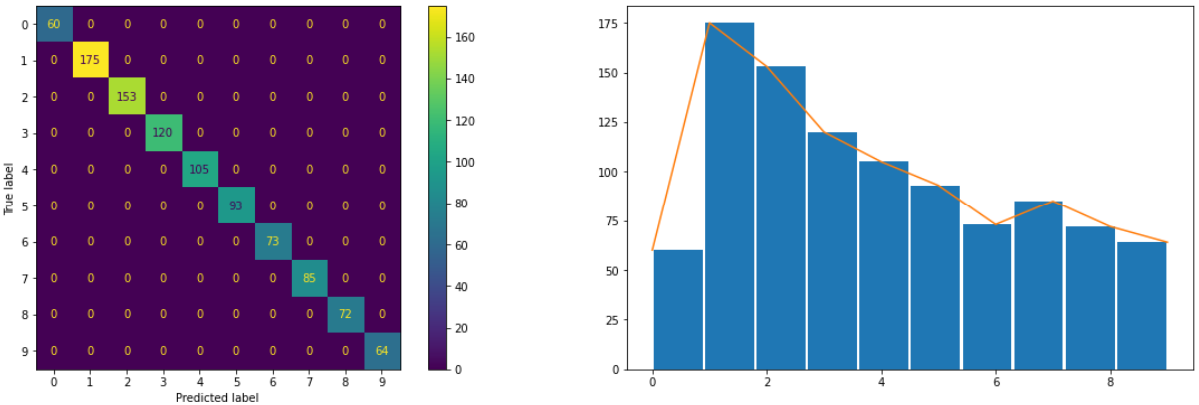


Figure 5: CNN with Data Augmentation Training Dataset

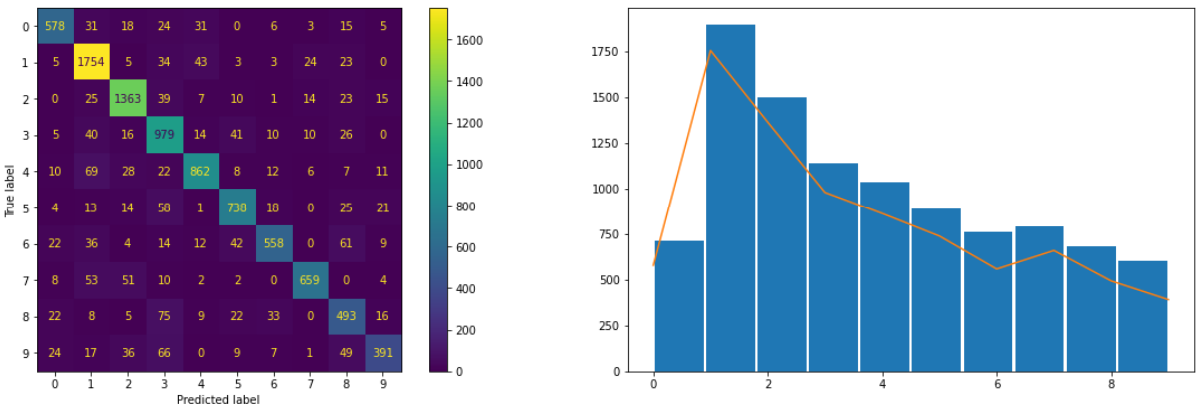


Figure 6: CNN with Data Augmentation Testing Dataset

### CIFAR with Data Augmentation

The results for the CIFAR model with data augmentation used can be seen in figures 6 for the training data and figure 7 for the testing data. The overall accuracy of the model can be seen below under model comparison on both the training and testing data given in table 2 which shows the percentage of correct samples classified. The overall accuracy of the model can be seen below under model comparison on both the training and testing data given in table 2 which shows the percentage of correct samples classified being 97% on the training images and 80.53% on the testing dataset.

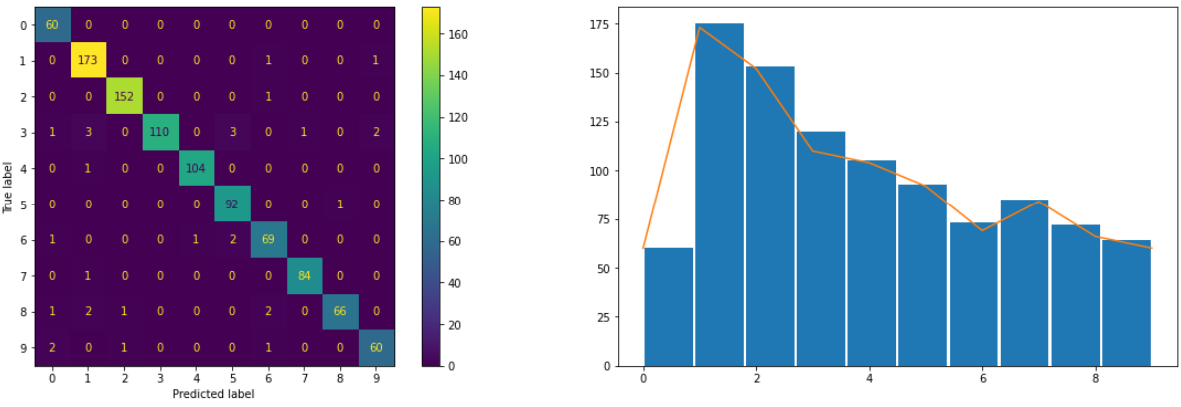


Figure 7: CIFAR with Data Augmentation Training Dataset

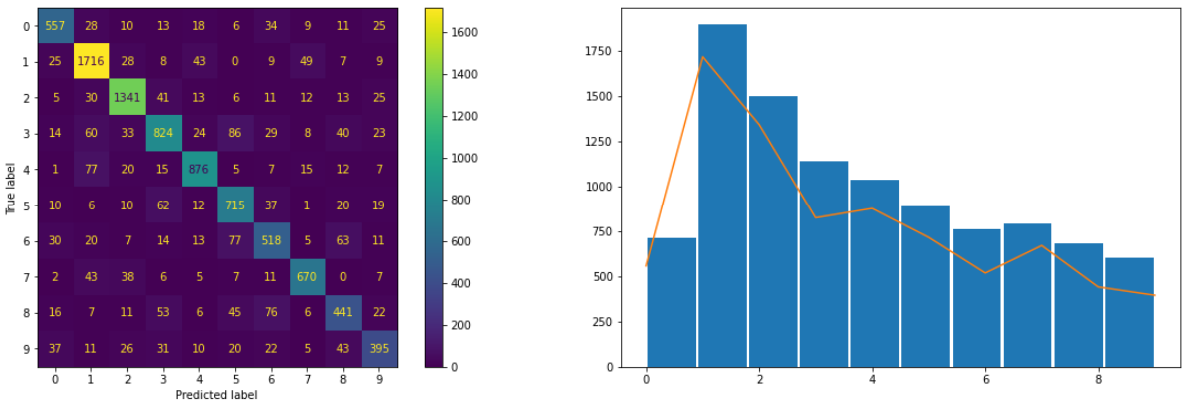


Figure 8: CIFAR with Data Augmentation Testing Dataset

### Comparison of models

• Your comparison should consider both raw performance and the different performance characteristics between methods. For example, do some methods work better on some classes than others, and why may this be? Are there cases that cause all methods to fail and if so, what are the characteristics of these?

• Include all relevant figures and/or tables to support the response.

2. No individual model discussion is given.

3. Needs to consider model validity in evaluation as well.

Table : Accuracy of Models on Datasets

|  |  |  |
| --- | --- | --- |
| Models | Training data Accuracy (%) | Testing data Accuracy (%) |
| CNN with no data augmentation | 100.0 | 82.66 |
| CNN with data augmentation | 100.0 | 83.75 |
| CNN CIFAR\_small model | 97.0 | 80.53 |