HOMEWORK II

Machine Learning

Ahmed Hesham El Sheikh
1873337

Table of Contents

I. Introduction

The aim of this homework is to train a convolutional neural network on a medium size dataset, to classify between different kinds of boats. The dataset used was MarDCT, pictures of the boats were taken using a camera in Venice, Italy.

In this project, keras framework was used -using TensorFlow the as backend- The python scripts included handled the data in terms of preprocessing and feeding the images to the CNN to predict the boat types. Not only, keras with TensorFlow backend, matplot library for plotting of graphs, and showing different visuals, sklearn for the calculation of accuracy scores, and classification report, pandas for reading of the ground truth file, lastly, NumPy.

II. Dataset

The dataset used is MarDCT as mentioned before, it contains images of Venice boats, images taken from different angles, different lighting conditions. The dataset contain 4774 training images belongs to 24 training classes, and 1969 images belonging to 24 testing classes.

This was the first challenge faced in this project, as sum of the classes in the training set are not there in the testing set and vice versa, so in order to deal with this unbalanced dataset -in terms of classes, after figuring out which classes, I had to remove them. Those classes were ['Cacciapesca', 'Caorlina', 'Lanciamaggioredi10mMarrone', 'Sanpierota', 'VigilidelFuoco', 'SnapshotBarcaParziale', 'SnapshotBarcaMultipla', 'Mototopocorto'].

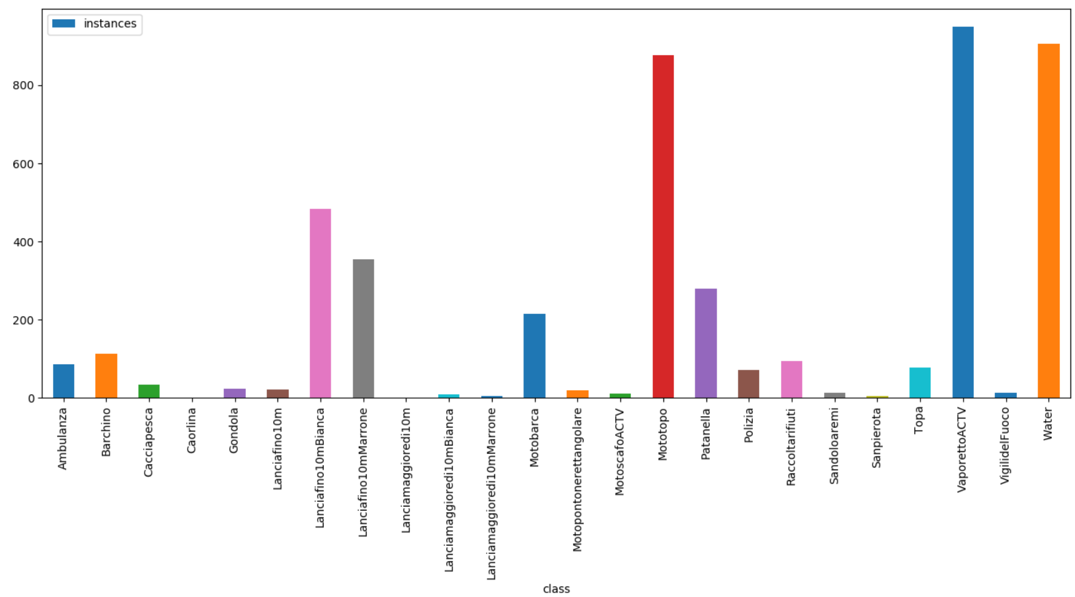


Figure 1: Number of instances before removing any classes

After taking a look on the previous figure, we can safely remove these classes, and work with the rest classes, which are ['Alilaguna', 'Ambulanza', 'Barchino', 'Gondola', 'Lanciafino10m', 'Lanciafino10mBianca', 'Lanciafino10mMarrone', 'Lanciamaggioredi10mBianca', 'Motobarca', 'Motopontonerettangolare', 'MotoscafoACTV', 'Mototopo', 'Patanella', 'Polizia', 'Raccoltarifiuti', 'Sandoloaremi', 'Topa', 'VaporettoACTV', 'Water'], after keeping those classes, we are left with only 19 classes, 4717 training images, and 1681 testing images.

III. Data Preprocessing

Before processing to training the model, we have to preprocess the data to make sure that our model gets different views of the same image, and this will add to our model, and help it to extract more general features, as well as, help it fight overfitting.

So, I applied different images augmentation techniques of rotating, width and height shifts, shear, horizontal and vertical flips, zooming in. As well as, data splitting into 80% for training and 20% for validation.

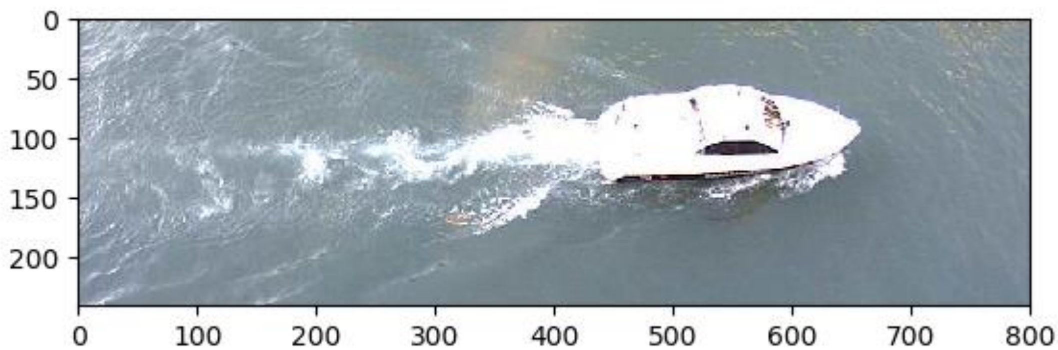


Figure 2: Original image with no changes in it.

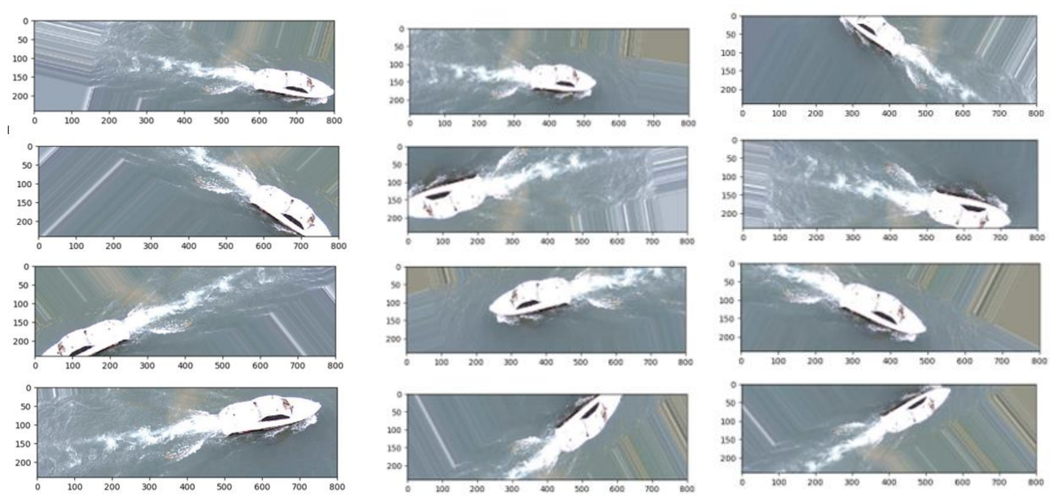


Figure 3:Few images of the samples generated after preprocessing

IV. Convolutional Neural Network

The approach taken in this project was to build a simple CNN, consisting of several layers 2 convolution layers (Conv2D layer -> ReLu Activation -> MaxPooling2D) and a (ZeroPooling2D) layer in between of both convolution layers, why? Because the layer progressively reduces the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. Batch Normalization layer was added between the Conv2D layer and the 3rd ReLu layer, as it normalizes the output of the Conv2D layer to the ReLu. A Flatten layer was used to turn the feature vector into a 1D to be used by the ANN classifier layer. Using a dropout layer with probability of 0.7 to switch off some neurons of our NN, to prevent overfitting.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 252, 32)      1472
_________________________________________________________________
activation (Activation)      (None, 254, 252, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 126, 32)      0
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 129, 128, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 127, 124, 32)      15392
_________________________________________________________________
activation_1 (Activation)    (None, 127, 124, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 62, 32)        0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 65, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 63, 60, 64)        30784
_________________________________________________________________
batch_normalization (BatchNo (None, 63, 60, 64)        256
_________________________________________________________________
activation_2 (Activation)    (None, 63, 60, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 30, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 59520)             0
_________________________________________________________________
dense (Dense)                (None, 64)                3809344
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 19)                1235
_________________________________________________________________
activation_4 (Activation)    (None, 19)                0
=================================================================
Total params: 3,858,483
Trainable params: 3,858,355
Non-trainable params: 128
```

Figure 4: Model A Summary

The images from the dataset was fed into the network of (width, height, channels) were (256, 256, 3), dividing the training images into 3783 training images and 934 validation images, and at last 1681 testing images.

The fitting process, was done using several parameters, using the training data, as well as, the validation data, epochs were set arbitrarily to 100 epochs, however, it was controlled using early stopping to prevent overfitting to the data, added to, keeping log throughout the whole training process, apart from all of that, the model was using a stochastic gradient descent with Nesterov momentum, a starting learning rate of $1e^{-6}$, momentum of 0.9 to help the model converge faster, finally, the steps per epoch and steps per validation will both determined using

$$Steps\ per\ epoch = {}^{number\ of\ samples\ in\ training\ set}/_{Batch\ size\ of\ training\ set}$$

$$Validation\ Steps\ = {}^{number\ of\ samples\ in\ validation\ set}/_{Batch\ size\ of\ validation\ set}$$

to ensure we use the minimum amount of time, the used multiprocessing, as well as, setting the script to run on the main thread.

Tweaking few of the above parameters resulted in different results of course, to start with, the number of epochs, I assigned 50 number of epochs arbitrarily, and I noticed that sometimes the model overfit, so I added the early stopping callback function for the model. Not to forget, having high learning rate may result in non-convergence approach because the model might skip the minima and get stuck out of it, as shown in the below figure learning rate (lr) of $1e^{-5}$ showed earlier convergence than the $1e^{-6}$ model, the learning rates did not affect only the convergence rate, it also had an effect on the overall accuracy of the model -on the test data- the as shown in figure 6, so as a solution to that I used an initial learning rate of $1e^{-6}$ and it modifies after every epoch which actually helped model A to converge and get good results, added to the use of Nesterov momentum with our optimizer yielded in higher convergence rate.

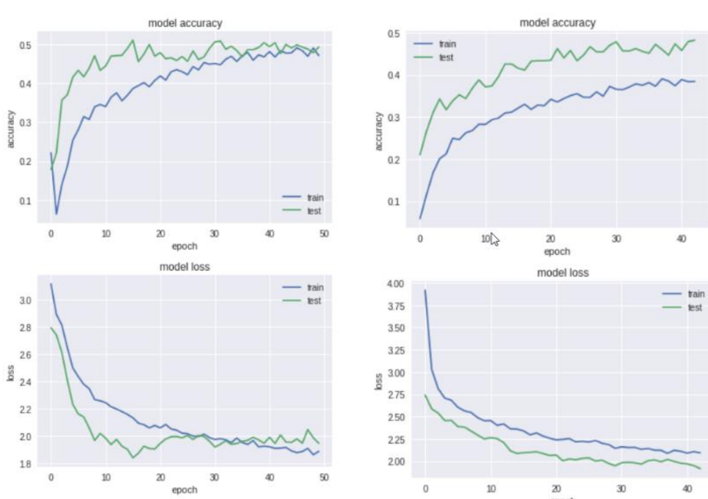


Figure 5: Comparison between learning rate of 1e^ (-5) on the left-hand image and 1e^ (-6) on the right-hand image

```
1671/1671 [==============================] - 13s 8ms/step

The Accuracy score is: 59.78%

The Classification Report:
              precision    recall  f1-score   support

           0       0.22      0.21      0.22        19
           1       0.20      0.05      0.07        22
           2       0.00      0.00      0.00        51
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         7
           5       0.41      0.62      0.50       217
           6       0.12      0.02      0.04       125
           7       0.00      0.00      0.00         6
           8       0.33      0.02      0.03        59
           9       0.00      0.00      0.00         3
          10       0.00      0.00      0.00         1
          11       0.45      0.62      0.52       274
          12       0.06      0.04      0.05        74
          13       0.00      0.00      0.00        15
          14       0.00      0.00      0.00        19
          15       0.00      0.00      0.00         3
          16       0.00      0.00      0.00        29
          17       0.73      0.97      0.83       325
          18       0.85      0.88      0.87       419

   micro avg       0.60      0.60      0.60      1671
   macro avg       0.18      0.18      0.16      1671
weighted avg       0.51      0.60      0.54      1671
```

```
1671/1671 [==============================] - 12s 7ms/step

The Accuracy score is: 55.42%

The Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.16      0.23        19
           1       0.00      0.00      0.00        22
           2       0.00      0.00      0.00        51
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         7
           5       0.34      0.49      0.40       217
           6       0.19      0.15      0.17       125
           7       0.00      0.00      0.00         6
           8       0.10      0.08      0.09        59
           9       0.00      0.00      0.00         3
          10       0.00      0.00      0.00         1
          11       0.42      0.55      0.47       274
          12       0.00      0.00      0.00        74
          13       0.00      0.00      0.00        15
          14       0.00      0.00      0.00        19
          15       0.00      0.00      0.00         3
          16       0.04      0.03      0.04        29
          17       0.72      0.93      0.81       325
          18       0.89      0.81      0.85       419

   micro avg       0.55      0.55      0.55      1671
   macro avg       0.16      0.17      0.16      1671
weighted avg       0.50      0.55      0.52      1671
```

Figure 6: On the left-hand side, we can see that having a lr = 1e^ (-5) having better overall performance, compared to 1e^ (-6)

Not to mention, the effects of having multiple Dropouts layers, given the fact that dropouts are used to regularize the CNN architecture used, but in our project, a medium sized dataset with a simple architecture dataset and more than a dropout layer resulted in huge drops in the performance rates, check Appendices C and D for training process, prediction process respectively

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 252, 32)	1472
activation (Activation)	(None, 254, 252, 32)	0
max_pooling2d (MaxPooling2D)	(None, 127, 126, 32)	0
zero_padding2d (ZeroPadding2	(None, 129, 128, 32)	0
conv2d_1 (Conv2D)	(None, 127, 124, 32)	15392
activation_1 (Activation)	(None, 127, 124, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 63, 62, 32)	0
dropout (Dropout)	(None, 63, 62, 32)	0
zero_padding2d_1 (ZeroPaddin	(None, 65, 64, 32)	0
conv2d_2 (Conv2D)	(None, 63, 60, 64)	30784
batch_normalization (BatchNo	(None, 63, 60, 64)	256
activation_2 (Activation)	(None, 63, 60, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 31, 30, 64)	0
dropout_1 (Dropout)	(None, 31, 30, 64)	0
flatten (Flatten)	(None, 59520)	0
dense (Dense)	(None, 64)	3809344
activation_3 (Activation)	(None, 64)	0
dropout_2 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 19)	1235
activation_4 (Activation)	(None, 19)	0

Figure 7: Adding more Dropout layers to model A

So, this will make us question why not to add more layers other than Dropout layers, what will happen? Actually, adding layers unnecessarily to any CNN will increase the number of parameters only for the smaller datasets. It's true for some reasons that on adding more hidden layers, it will give more accuracy. This is true for larger datasets, as more layers with less stride factor will extract more features for the input data. In CNN, how we play with the network's architecture is completely dependent on what are the requirements and how the dataset looks like. Increasing Unnecessary parameters will only result in overfit, which we are trying to combat all the time.

The above model which we will refer to as model A training process can be found in Appendix A, the following plots shows the accuracy plotted against the validation accuracy, as well as, loss plotted against the validation loss, the loss function used was categorical cross entropy as we are classifying 19 classes.

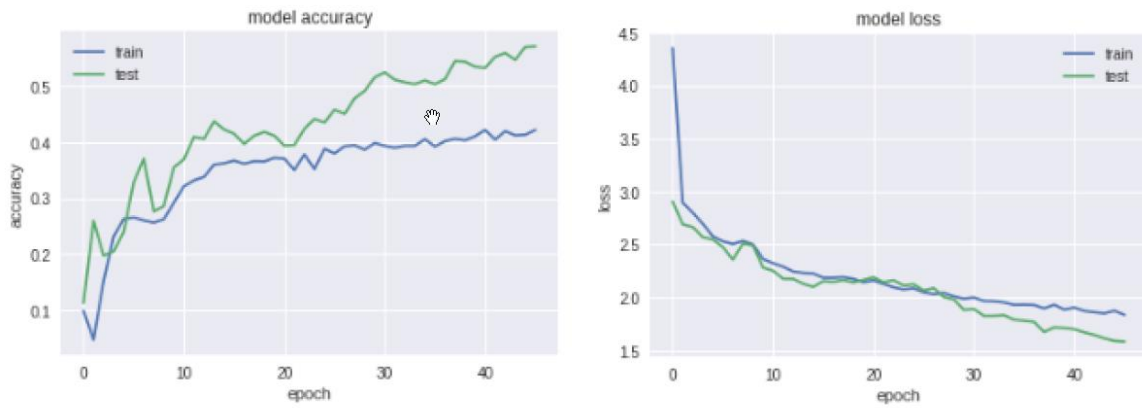
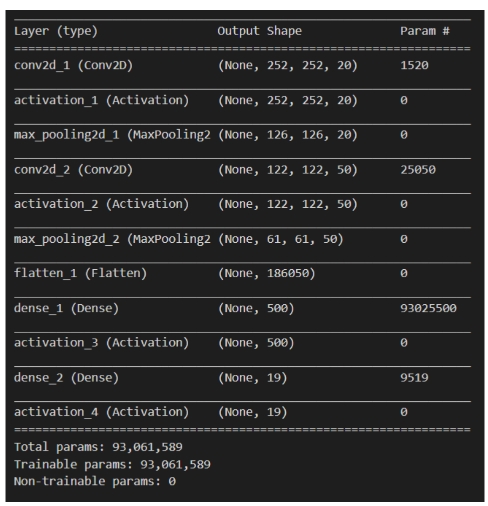


Figure 8: First model accuracy & loss values through the fitting process

Another model was used in this process which we will refer to as model B, was the LeNet architecture, here is a view of the model summary



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 252, 252, 20)	1520
activation_1 (Activation)	(None, 252, 252, 20)	0
max_pooling2d_1 (MaxPooling2	(None, 126, 126, 20)	0
conv2d_2 (Conv2D)	(None, 122, 122, 50)	25050
activation_2 (Activation)	(None, 122, 122, 50)	0
max_pooling2d_2 (MaxPooling2	(None, 61, 61, 50)	0
flatten_1 (Flatten)	(None, 186050)	0
dense_1 (Dense)	(None, 500)	93025500
activation_3 (Activation)	(None, 500)	0
dense_2 (Dense)	(None, 19)	9519
activation_4 (Activation)	(None, 19)	0

Total params: 93,061,589
Trainable params: 93,061,589
Non-trainable params: 0

Figure 9: LeNet model or model B

It yielded a very low-test accuracy of 1% which is even lower than the random guess, which is calculated as $^{1}/_{number\ of\ classes}$, in our case the random guess is 5.3% approximately, and this is due to being too simple, it has a convolutional layer less than model A, and has no batch normalization layer, and due to the very large trainable parameters model B used made it to memorize the parameters, and this concludes why the very low accuracy

V. Conclusion

The use of keras made it possible and easy to have a working model in few lines of code. CNN in general provides good results, as they were implemented for image, audio, videos classification. The training of a model depends on different parameters, the data augmentation, number of samples as per class, training time (or number of epochs), types of layers, activation functions used, as well as, types of loss function used, size of the network, however, simple networks may show better results.

The big number of layers is only good if our dataset is large thus the complex architecture, there is multiple ways to avoid overfitting by adding a dropout layer, using early stopping, using data generator, augmenting the present data. Using SGD with Nesterov momentum helps the network converge faster.

VI. Appendices

Appendix A – Model A training process

Epoch 1/100

118/118 [==============================] - 36s 306ms/step - loss: 4.3288 - acc: 0.1039 - val_loss: 2.9052 - val_acc: 0.1131

Epoch 2/100

118/118 [==============================] - 35s 300ms/step - loss: 2.8901 - acc: 0.0469 - val_loss: 2.6895 - val_acc: 0.2594

Epoch 3/100

118/118 [==============================] - 36s 302ms/step - loss: 2.7879 - acc: 0.1563 - val_loss: 2.6631 - val_acc: 0.1973

Epoch 4/100

118/118 [==============================] - 36s 303ms/step - loss: 2.6830 - acc: 0.2339 - val_loss: 2.5693 - val_acc: 0.2051

Epoch 5/100

118/118 [==============================] - 36s 306ms/step - loss: 2.5636 - acc: 0.2674 - val_loss: 2.5460 - val_acc: 0.2395

Epoch 6/100

118/118 [==============================] - 35s 299ms/step - loss: 2.5160 - acc: 0.2694 - val_loss: 2.4756 - val_acc: 0.3282

Epoch 7/100

118/118 [==============================] - 36s 302ms/step - loss: 2.4932 - acc: 0.2620 - val_loss: 2.3553 - val_acc: 0.3703

Epoch 8/100

118/118 [==============================] - 35s 300ms/step - loss: 2.5226 - acc: 0.2583 - val_loss: 2.5097 - val_acc: 0.2761

Epoch 9/100

118/118 [==============================] - 35s 300ms/step - loss: 2.4894 - acc: 0.2641 - val_loss: 2.4906 - val_acc: 0.2860

Epoch 10/100

118/118 [==============================] - 36s 302ms/step - loss: 2.3497 - acc: 0.2938 - val_loss: 2.2808 - val_acc: 0.3548

Epoch 11/100

118/118 [==============================] - 36s 302ms/step - loss: 2.3104 - acc: 0.3236 - val_loss: 2.2506 - val_acc: 0.3692

Epoch 12/100

118/118 [==============================] - 35s 301ms/step - loss: 2.2756 - acc: 0.3351 - val_loss: 2.1758 - val_acc: 0.4091

Epoch 13/100

118/118 [==============================] - 36s 302ms/step - loss: 2.2303 - acc: 0.3421 - val_loss: 2.1738 - val_acc: 0.4058

Epoch 14/100

118/118 [==============================] - 35s 301ms/step - loss: 2.2165 - acc: 0.3638 - val_loss: 2.1280 - val_acc: 0.4368

Epoch 15/100

118/118 [==============================] - 35s 299ms/step - loss: 2.2138 - acc: 0.3643 - val_loss: 2.0987 - val_acc: 0.4224

Epoch 16/100

118/118 [==============================] - 35s 300ms/step - loss: 2.1755 - acc: 0.3691 - val_loss: 2.1530 - val_acc: 0.4146

Epoch 17/100

118/118 [==============================] - 36s 303ms/step - loss: 2.1711 - acc: 0.3654 - val_loss: 2.1459 - val_acc: 0.3966

Epoch 18/100

118/118 [==============================] - 36s 301ms/step - loss: 2.1772 - acc: 0.3699 - val_loss: 2.1620 - val_acc: 0.4113

Epoch 19/100

118/118 [==============================] - 36s 301ms/step - loss: 2.1635 - acc: 0.3675 - val_loss: 2.1412 - val_acc: 0.4180

Epoch 20/100

118/118 [==============================] - 36s 303ms/step - loss: 2.1318 - acc: 0.3748 - val_loss: 2.1642 - val_acc: 0.4113

Epoch 21/100

118/118 [==============================] - 36s 302ms/step - loss: 2.1480 - acc: 0.3735 - val_loss: 2.1890 - val_acc: 0.3936

Epoch 22/100

118/118 [==============================] - 36s 302ms/step - loss: 2.1170 - acc: 0.3524 - val_loss: 2.1423 - val_acc: 0.3947

Epoch 23/100

118/118 [==============================] - 35s 300ms/step - loss: 2.0819 - acc: 0.3812 - val_loss: 2.1616 - val_acc: 0.4235

Epoch 24/100

118/118 [==============================] - 36s 302ms/step - loss: 2.0646 - acc: 0.3540 - val_loss: 2.1113 - val_acc: 0.4412

Epoch 25/100

118/118 [==============================] - 36s 302ms/step - loss: 2.0777 - acc: 0.3900 - val_loss: 2.1226 - val_acc: 0.4346

Epoch 26/100

118/118 [==============================] - 36s 302ms/step - loss: 2.0382 - acc: 0.3818 - val_loss: 2.0653 - val_acc: 0.4579

Epoch 27/100

118/118 [==============================] - 35s 301ms/step - loss: 2.0162 - acc: 0.3964 - val_loss: 2.0871 - val_acc: 0.4501

Epoch 28/100

118/118 [==============================] - 35s 301ms/step - loss: 2.0291 - acc: 0.3973 - val_loss: 2.0042 - val_acc: 0.4778

Epoch 29/100

118/118 [==============================] - 35s 300ms/step - loss: 1.9955 - acc: 0.3903 - val_loss: 1.9805 - val_acc: 0.4911

Epoch 30/100

118/118 [==============================] - 36s 301ms/step - loss: 1.9742 - acc: 0.4022 - val_loss: 1.8821 - val_acc: 0.5155

Epoch 31/100

118/118 [==============================] - 36s 302ms/step - loss: 1.9872 - acc: 0.3963 - val_loss: 1.8897 - val_acc: 0.5244

Epoch 32/100

118/118 [==============================] - 35s 296ms/step - loss: 1.9535 - acc: 0.3938 - val_loss: 1.8250 - val_acc: 0.5111

Epoch 33/100

118/118 [==============================] - 35s 297ms/step - loss: 1.9512 - acc: 0.3972 - val_loss: 1.8256 - val_acc: 0.5067

Epoch 34/100

118/118 [==============================] - 35s 297ms/step - loss: 1.9388 - acc: 0.3972 - val_loss: 1.8313 - val_acc: 0.5033

Epoch 35/100

118/118 [==============================] - 35s 300ms/step - loss: 1.9162 - acc: 0.4089 - val_loss: 1.7889 - val_acc: 0.5100

Epoch 36/100

118/118 [==============================] - 35s 298ms/step - loss: 1.9223 - acc: 0.3943 - val_loss: 1.7789 - val_acc: 0.5033

Epoch 37/100

118/118 [==============================] - 36s 301ms/step - loss: 1.9174 - acc: 0.4045 - val_loss: 1.7677 - val_acc: 0.5122

Epoch 38/100

118/118 [==============================] - 35s 299ms/step - loss: 1.8840 - acc: 0.4097 - val_loss: 1.6727 - val_acc: 0.5443

Epoch 39/100

118/118 [==============================] - 35s 300ms/step - loss: 1.9189 - acc: 0.4061 - val_loss: 1.7161 - val_acc: 0.5432

Epoch 40/100

118/118 [==============================] - 36s 301ms/step - loss: 1.8732 - acc: 0.4142 - val_loss: 1.7109 - val_acc: 0.5344

Epoch 41/100

118/118 [==============================] - 35s 299ms/step - loss: 1.8913 - acc: 0.4253 - val_loss: 1.6989 - val_acc: 0.5322

Epoch 42/100

118/118 [==============================] - 35s 300ms/step - loss: 1.8616 - acc: 0.4074 - val_loss: 1.6690 - val_acc: 0.5521

Epoch 43/100

118/118 [==============================] - 35s 298ms/step - loss: 1.8473 - acc: 0.4232 - val_loss: 1.6451 - val_acc: 0.5588

Epoch 44/100

118/118 [==============================] - 36s 302ms/step - loss: 1.8379 - acc: 0.4158 - val_loss: 1.6124 - val_acc: 0.5466

Epoch 45/100

118/118 [==============================] - 36s 303ms/step - loss: 1.8642 - acc: 0.4149 - val_loss: 1.5887 - val_acc: 0.5698

Epoch 46/100

118/118 [==============================] - 36s 303ms/step - loss: 1.8222 - acc: 0.4253 - val_loss: 1.5822 - val_acc: 0.5710

Epoch 00046: early stopping

Appendix B – Model A prediction process

1671/1671 [==============================] - 12s 7ms/step

1671/1671 [==============================] - 13s 8ms/step

The Accuracy is: 61.34%

The Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	19
1	0.00	0.00	0.00	22
2	0.00	0.00	0.00	51
3	0.00	0.00	0.00	3
4	0.00	0.00	0.00	7
5	0.62	0.20	0.30	217
6	0.37	0.56	0.45	125
7	0.00	0.00	0.00	6
8	0.00	0.00	0.00	59
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	1
11	0.42	0.90	0.58	274
12	0.00	0.00	0.00	74
13	0.00	0.00	0.00	15
14	0.00	0.00	0.00	19
15	0.25	0.33	0.29	3
16	0.00	0.00	0.00	29
17	0.92	0.89	0.90	325
18	0.73	0.89	0.81	419
micro avg	0.61	0.61	0.61	1671
macro avg	0.18	0.20	0.17	1671
weighted avg	0.54	0.61	0.55	1671

Appendix C – Model A with more Dropout layers training process

Epoch 1/100

118/118 [==============================] - 130s 1s/step - loss: 7.0944 - acc: 0.0665 - val_loss: 2.8853 - val_acc: 0.1250

Epoch 2/100

118/118 [==============================] - 127s 1s/step - loss: 5.0368 - acc: 0.0879 - val_loss: 2.9119 - val_acc: 0.1175

Epoch 3/100

118/118 [==============================] - 127s 1s/step - loss: 4.0721 - acc: 0.0805 - val_loss: 2.9200 - val_acc: 0.1153

Epoch 4/100

118/118 [==============================] - 126s 1s/step - loss: 3.7027 - acc: 0.0879 - val_loss: 2.9275 - val_acc: 0.1220

Epoch 5/100

118/118 [==============================] - 126s 1s/step - loss: 3.4974 - acc: 0.0783 - val_loss: 2.9308 - val_acc: 0.1109

Epoch 6/100

118/118 [==============================] - 127s 1s/step - loss: 3.3772 - acc: 0.0918 - val_loss: 2.9353 - val_acc: 0.0909

Epoch 7/100

118/118 [==============================] - 126s 1s/step - loss: 3.2650 - acc: 0.0905 - val_loss: 2.9363 - val_acc: 0.0909

Epoch 8/100

118/118 [==============================] - 127s 1s/step - loss: 3.2247 - acc: 0.0866 - val_loss: 2.9383 - val_acc: 0.1164

Epoch 9/100

118/118 [==============================] - 126s 1s/step - loss: 3.1661 - acc: 0.0925 - val_loss: 2.9388 - val_acc: 0.1164

Epoch 10/100

118/118 [==============================] - 126s 1s/step - loss: 3.1661 - acc: 0.0880 - val_loss: 2.9390 - val_acc: 0.1330

Epoch 11/100

118/118 [==============================] - 126s 1s/step - loss: 3.0865 - acc: 0.0866 - val_loss: 2.9403 - val_acc: 0.1175

Epoch 12/100

118/118 [==============================] - 126s 1s/step - loss: 3.0892 - acc: 0.0958 - val_loss: 2.9409 - val_acc: 0.1319

Epoch 13/100

118/118 [==============================] - 128s 1s/step - loss: 3.0721 - acc: 0.0950 - val_loss: 2.9413 - val_acc: 0.1785

Epoch 14/100

118/118 [==============================] - 136s 1s/step - loss: 3.0607 - acc: 0.0993 - val_loss: 2.9407 - val_acc: 0.2084

Epoch 15/100

118/118 [==============================] - 126s 1s/step - loss: 3.0599 - acc: 0.1080 - val_loss: 2.9410 - val_acc: 0.2084

Epoch 16/100

118/118 [==============================] - 126s 1s/step - loss: 3.0383 - acc: 0.1040 - val_loss: 2.9409 - val_acc: 0.2040

Epoch 17/100

118/118 [==============================] - 127s 1s/step - loss: 3.0324 - acc: 0.1101 - val_loss: 2.9410 - val_acc: 0.2018

Epoch 18/100

118/118 [==============================] - 126s 1s/step - loss: 3.0297 - acc: 0.1004 - val_loss: 2.9413 - val_acc: 0.1996

Epoch 19/100

118/118 [==============================] - 126s 1s/step - loss: 3.0326 - acc: 0.1020 - val_loss: 2.9411 - val_acc: 0.2051

Epoch 20/100

118/118 [==============================] - 126s 1s/step - loss: 3.0055 - acc: 0.1007 - val_loss: 2.9416 - val_acc: 0.2073

Epoch 21/100

118/118 [==============================] - 125s 1s/step - loss: 3.0149 - acc: 0.1052 - val_loss: 2.9415 - val_acc: 0.2073

Epoch 22/100

118/118 [==============================] - 126s 1s/step - loss: 3.0117 - acc: 0.0997 - val_loss: 2.9409 - val_acc: 0.2151

Epoch 00022: early stopping

Appendix D – Model A with more Dropout layers prediction process

```
1671/1671 [==============================] - 17s 10ms/step
1671/1671 [==============================] - 17s 10ms/step

The Accuracy score is: 20.95%

The Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        19
           1       0.00      0.00      0.00        22
           2       0.00      0.00      0.00        51
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         7
           5       0.50      0.00      0.01       217
           6       0.00      0.00      0.00       125
           7       0.00      0.00      0.00         6
           8       0.00      0.00      0.00        59
           9       0.00      0.00      0.00         3
          10       0.00      0.00      0.00         1
          11       0.14      0.10      0.12       274
          12       0.00      0.00      0.00        74
          13       0.00      0.00      0.00        15
          14       0.00      0.00      0.00        19
          15       0.00      0.00      0.00         3
          16       0.00      0.00      0.00        29
          17       0.21      0.85      0.33       325
          18       0.52      0.11      0.17       419

   micro avg       0.21      0.21      0.21      1671
   macro avg       0.07      0.06      0.03      1671
weighted avg       0.26      0.21      0.13      1671
```

Appendix D – Model B (LeNet) training process

Epoch 1/100

118/118 [==============================] - 108s 913ms/step - loss: 15.8304 - acc: 0.0167 - val_loss: 15.7360 - val_acc: 0.0237

Epoch 2/100

118/118 [==============================] - 103s 872ms/step - loss: 15.7296 - acc: 0.0241 - val_loss: 15.7250 - val_acc: 0.0244

Epoch 3/100

118/118 [==============================] - 103s 872ms/step - loss: 15.7296 - acc: 0.0241 - val_loss: 15.7250 - val_acc: 0.0244

Epoch 4/100

118/118 [==============================] - 103s 873ms/step - loss: 15.7296 - acc: 0.0241 - val_loss: 15.7250 - val_acc: 0.0244

Epoch 5/100

118/118 [==============================] - 103s 871ms/step - loss: 15.7296 - acc: 0.0241 - val_loss: 15.7250 - val_acc: 0.0244

Epoch 6/100

118/118 [==============================] - 103s 869ms/step - loss: 15.7296 - acc: 0.0241 - val_loss: 15.7250 - val_acc: 0.0244

Epoch 7/100

118/118 [==============================] - 102s 867ms/step - loss: 15.7296 - acc: 0.0241 - val_loss: 15.7250 - val_acc: 0.0244

Epoch 00007: early stopping

Appendix F – Model B (LeNet) Prediction

```
1671/1671 [==============================] - 14s 9ms/step
1671/1671 [==============================] - 16s 9ms/step

The Accuracy score is: 1.14%

The Classification Report:
              precision    recall  f1-score   support

           0       0.01      1.00      0.02        19
           1       0.00      0.00      0.00        22
           2       0.00      0.00      0.00        51
           3       0.00      0.00      0.00         3
           4       0.00      0.00      0.00         7
           5       0.00      0.00      0.00       217
           6       0.00      0.00      0.00       125
           7       0.00      0.00      0.00         6
           8       0.00      0.00      0.00        59
           9       0.00      0.00      0.00         3
          10       0.00      0.00      0.00         1
          11       0.00      0.00      0.00       274
          12       0.00      0.00      0.00        74
          13       0.00      0.00      0.00        15
          14       0.00      0.00      0.00        19
          15       0.00      0.00      0.00         3
          16       0.00      0.00      0.00        29
          17       0.00      0.00      0.00       325
          18       0.00      0.00      0.00       419

   micro avg       0.01      0.01      0.01      1671
   macro avg       0.00      0.05      0.00      1671
weighted avg       0.00      0.01      0.00      1671
```