Report on Homework 2

Gerardo Paladino

1918178

1 Introduction

1.1 Introduction

This paper is a report on Homework 2 of the Machine Learning course. The assignment involves solving an image classification problem to learn the behaviour of a racing car in a Gym environment. Two models based on CNN were built, discussed, compared, and the main results were reported.

1.2 Some considerations

On first inspection of the train and the test sets, it is clear that in both cases the set is unbalanced, as shown in Figure 1.

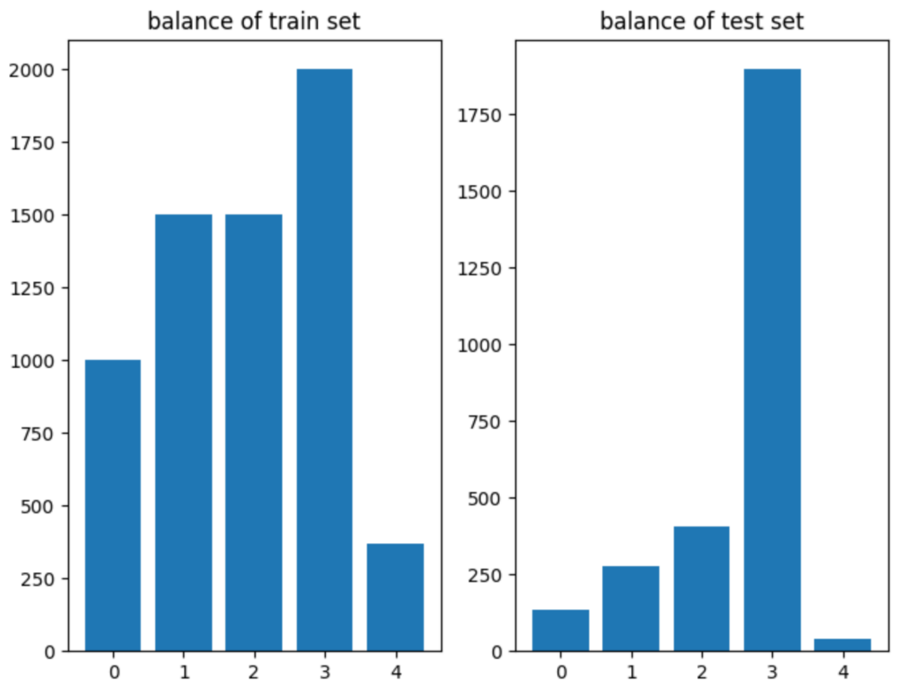


Figure 1: Diagram of 5 class. The x-axis represents a class, the y-axis represents a number of samples that have the corresponding class, for train and test set.

In particular, class '3' is oversampled while class '4' is undersampled. This imbalance is more pronounced in the test set, which may pose a challenge for the models to accurately capture the features of class '4'. To address this issue, I attempted to balance the datasets using various methods. However, each method resulted in an unexpected oscillation of the validation accuracy, to varying degrees. As a result, I ultimately decided not to balance the datasets. For this reason, neither model predicts the class '4'. In the Gym environment, class '4'

corresponds to 'brake', so I expect the car to start the turn without braking.
Another consideration is on the single samples of datasets. As shown in figure 2, the images within each class exhibit a high degree of similarity to each other; For instance, the second image belonging to class '1' bears a striking resemblance to the third image belonging to class '3'.

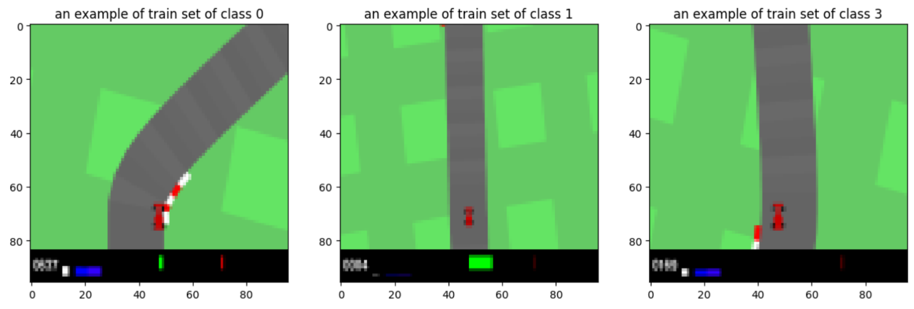

an example of train set of class 0
an example of train set of class 1
an example of train set of class 3
0
20
40
60
80

Figure 2: Examples of images in the datasets.

For this reason, it is expected that the models may encounter difficulty in capturing features for each class, resulting in lower accuracy.

2 Pre-Processing

Before starting, I performed data augmentation to increase the dataset by slightly modifying images. Following the guidelines provided, I normalized the pixels, applied rotation and zoom to the training set (with shuffle), and only normalized the test set.
Afterwards, I split the train set, using 20% of it as a validation set. The batch size for the training set is 256, while the batch size for the validation set is 128. The batch size for the test set is 2. The class mode is set to 'categorical' for all three sets.

3 Models Architectures

The main architecture is the same for both models, and is shown in figure 3. For Model 1, the first convolutional layer has 16 filters with a 3x3 kernel and uses the ReLU activation function, followed by max pooling. The second convolutional layer is identical, except it has 32 filters. After that, there is a flatten layer that converts the data into a vectorized format. Subsequently, there is the first dense layer with 64 neurons, which uses the ReLU activation function and has an L2 regularizer with a rate of 0.1. This layer is followed by a dropout that 'turns off' some neurons, and then by the second and final dense layer with 5 neurons (representing the number of classes) and SoftMax as the activation function. The optimizer chosen is Adam with a learning rate of 0.001, the loss function is categorical cross-entropy, and the metric used is accuracy.

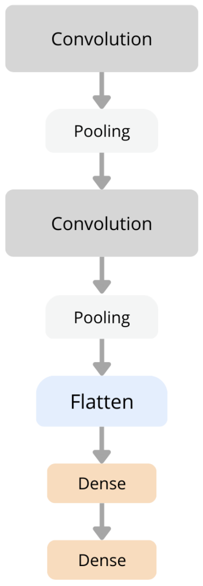
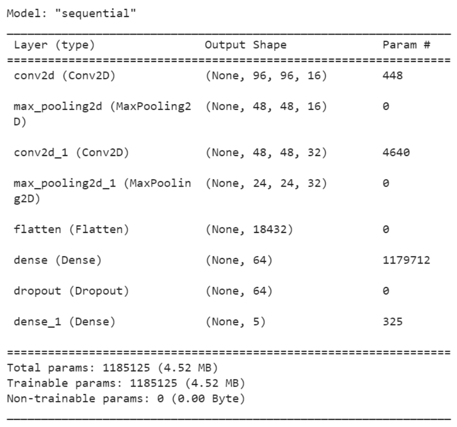


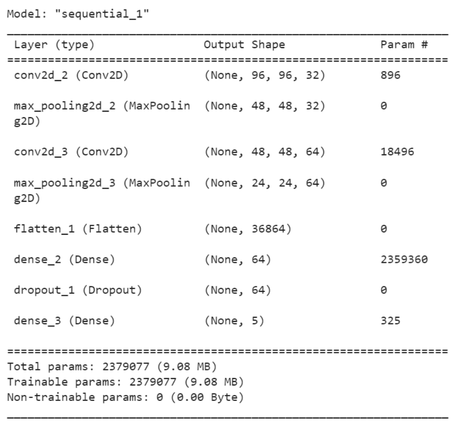
Figure 3: Model architecture.

Model 2 has the same two convolutional layers as Model 1, but with 32 filters for the first layer and 64 filters for the second layer. The rest of the model is the same, except for the use of a different regularizer, specifically L1 with a rate of 0.001.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 96, 96, 16)        448

 max_pooling2d (MaxPooling2  (None, 48, 48, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 48, 48, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 24, 24, 32)        0
 g2D)

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 64)                1179712

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 5)                 325

=================================================================
Total params: 1185125 (4.52 MB)
Trainable params: 1185125 (4.52 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

(a) Model 1.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 96, 96, 32)        896

 max_pooling2d_2 (MaxPoolin  (None, 48, 48, 32)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 48, 48, 64)        18496

 max_pooling2d_3 (MaxPoolin  (None, 24, 24, 64)        0
 g2D)

 flatten_1 (Flatten)         (None, 36864)             0

 dense_2 (Dense)             (None, 64)                2359360

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 5)                 325

=================================================================
Total params: 2379077 (9.08 MB)
Trainable params: 2379077 (9.08 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

(b) Model 2.

Figure 4

4 Evaluations and Comparison

4.1 Training

The two models were trained for 10 epochs. Figures 5 and 6 show the accuracy metric and loss history for the training and validation sets.

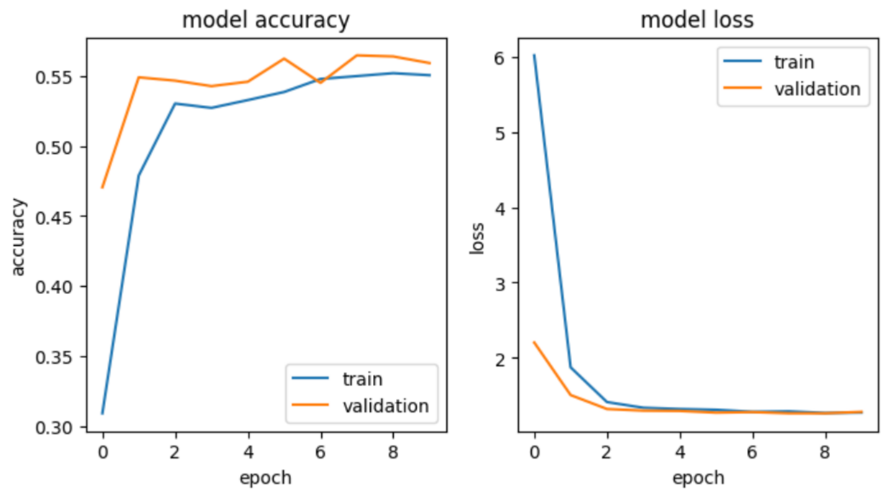


Figure 5: Accuracy and loss plots for model 1.

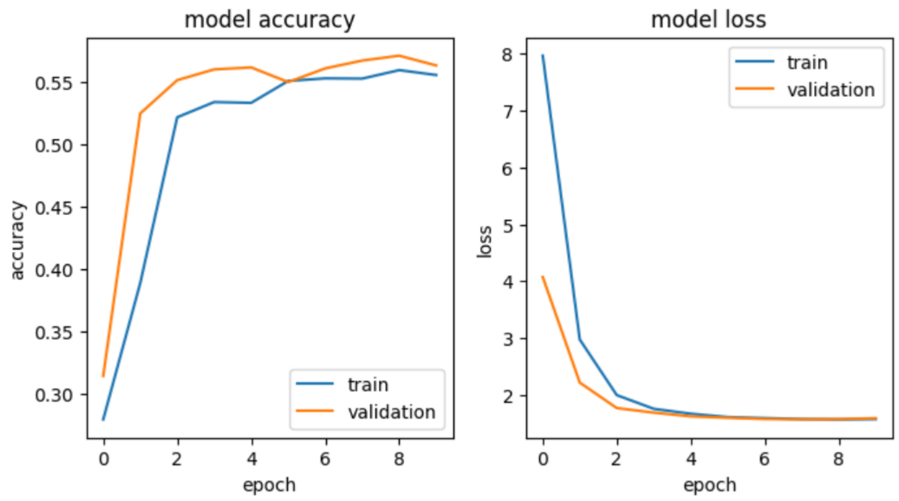
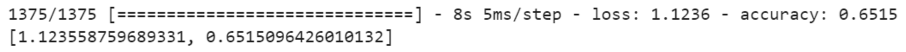


Figure 6: Accuracy and loss plots for model 2.

The loss decreases monotonically in both models, while the accuracy growth is not completely linear but still acceptable for both the train and validation sets. The regularizers do not affect the shape of the loss function, but they prevent overfitting.

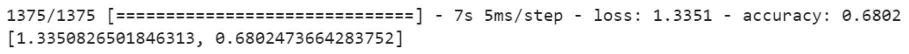
4.2 Test set

Although the accuracy on the train set and validation set is not optimal, it is acceptable on the test set, how show the figures 7 and 8.



```
1375/1375 [==============================] - 8s 5ms/step - loss: 1.1236 - accuracy: 0.6515
[1.123558759689331, 0.6515096426010132]
```

Figure 7: Accuracy on test set for model 1.

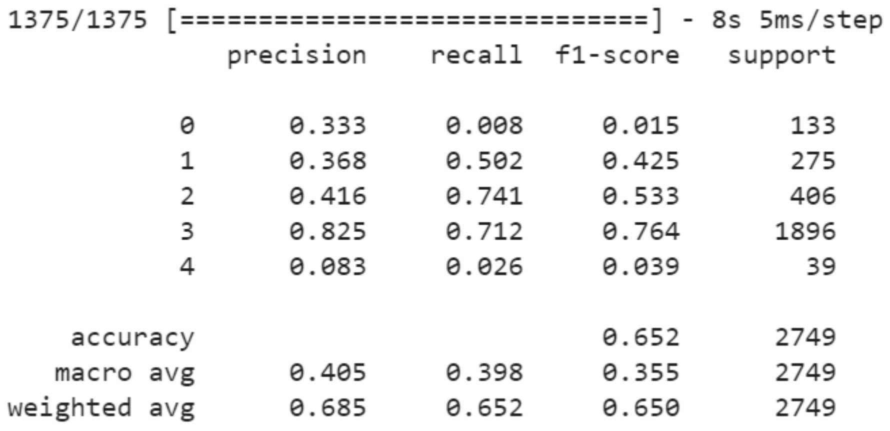


```
1375/1375 [==============================] - 7s 5ms/step - loss: 1.3351 - accuracy: 0.6802
[1.3350826501846313, 0.6802473664283752]
```

Figure 8: Accuracy on test set for model 2.

4.3 Other metrics

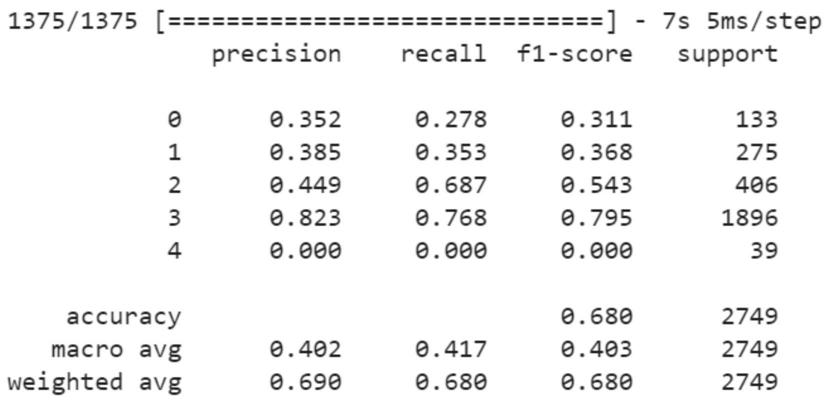
With these other metrics, it is evident that the class '4' was never predicted. This is apparent in the classification report (figures 9 and 10) and confusion matrix (figures 11 and 12).



```
1375/1375 [==============================] - 8s 5ms/step
              precision    recall  f1-score   support

           0      0.333     0.008     0.015       133
           1      0.368     0.502     0.425       275
           2      0.416     0.741     0.533       406
           3      0.825     0.712     0.764      1896
           4      0.083     0.026     0.039        39

    accuracy                          0.652      2749
   macro avg      0.405     0.398     0.355      2749
weighted avg      0.685     0.652     0.650      2749
```

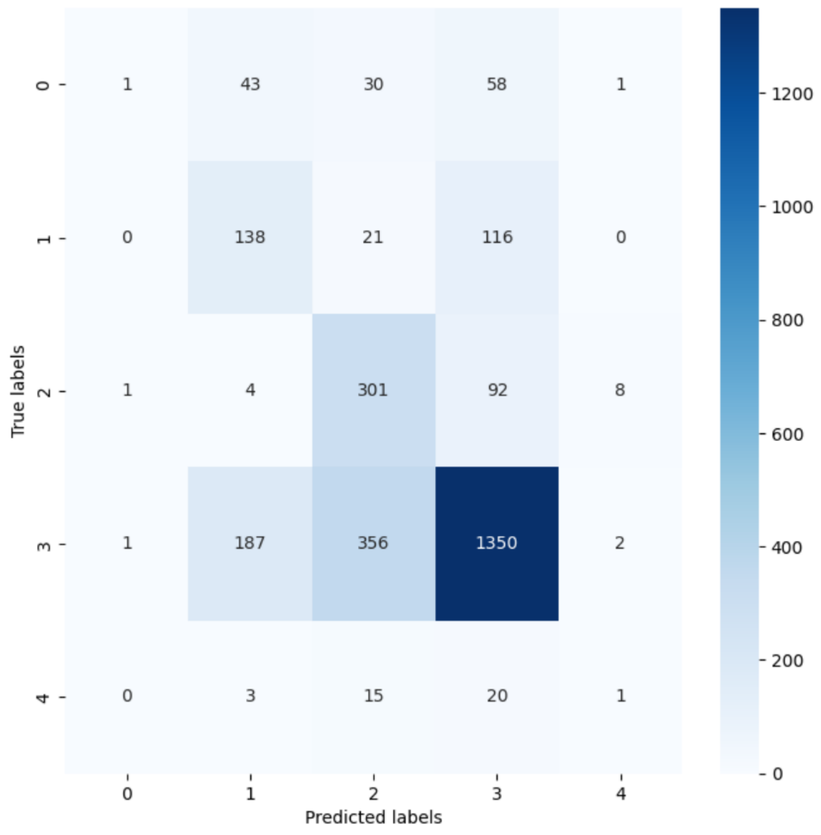
Figure 9: Classification report of model 1.

In this case, the model rarely predicts class '4', and only does so correctly once. Although the precision, recall, and f1-score are not zero, they are slightly higher. Additionally, the model predicts class '3' with good precision due to



```
1375/1375 [==============================] - 7s 5ms/step
              precision    recall  f1-score   support

           0      0.352     0.278     0.311       133
           1      0.385     0.353     0.368       275
           2      0.449     0.687     0.543       406
           3      0.823     0.768     0.795      1896
           4      0.000     0.000     0.000        39

    accuracy                          0.680      2749
   macro avg      0.402     0.417     0.403      2749
weighted avg      0.690     0.680     0.680      2749
```

Figure 10: Classification report of model 2.



True labels \ Predicted labels	0	1	2	3	4
0	1	43	30	58	1
1	0	138	21	116	0
2	1	4	301	92	8
3	1	187	356	1350	2
4	0	3	15	20	1

Figure 11: Confusion matrix of model 1.

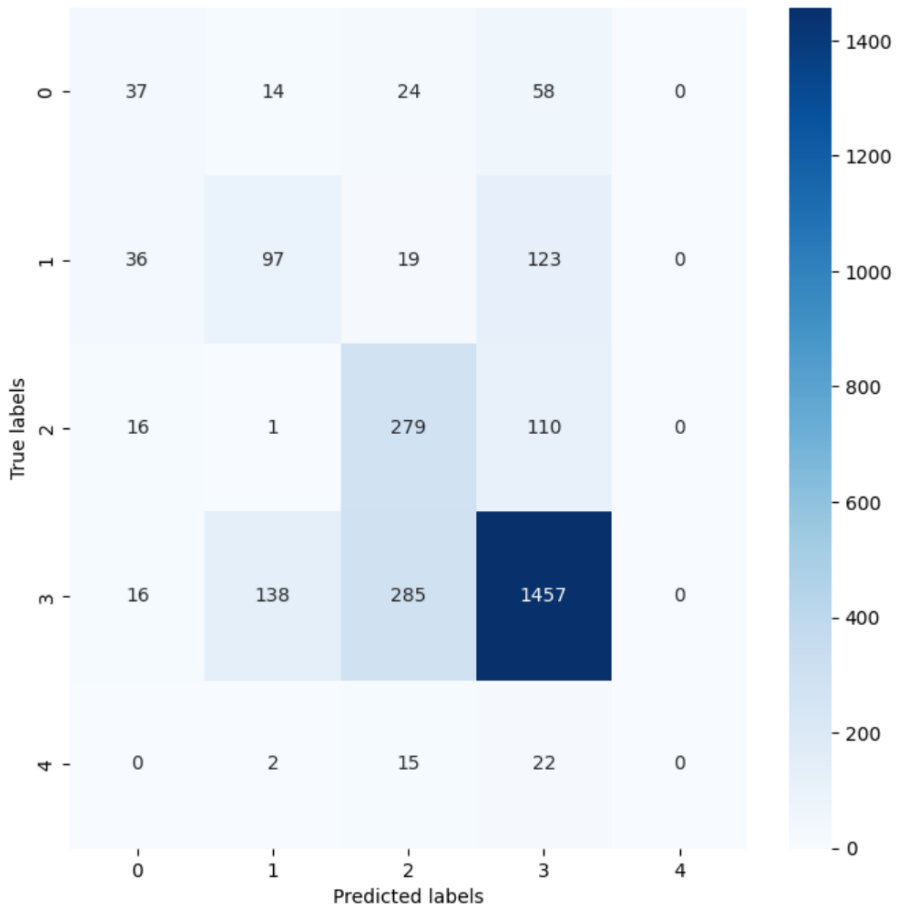


Figure 12: Confusion matrix of model 2.

oversampling.
In this other case (figure 10), the model never predicts class '4'. The metrics for this class are zero. Two confusion matrices are presented, one for each model. The first model correctly predicts class '0' and '4' only once, while the second model correctly predicts class '0' more often but never predicts class '4'. It is noteworthy that both models predict the same number of samples as class '2' that have a true class of either '2' or '3'.
Another important aspect is ensuring non-misclassification of class '1' (steer left) with class '2' (steer right). In both models, the number of samples predicted as class '1' but are actually class '2', and the number of samples predicted as class '2' but are actually class '1' is very low.

5 Conclusion

Both models are very similar, but I have chosen the second model due to its slightly higher accuracy on both the train and test sets. However, it is important to note that the second model uses twice as much memory as the first model due to the number of filters and the use of the absolute value of the norm for regularizer L1. The regularizer chosen and the number of filters are the discriminant for two models. In a CNN, the regularizer influences weight regularization during training, reducing overfitting. Weight regularization limits the magnitude of weights through techniques like L1 regularization (chosen by model 2) or L2 regularization (chosen by model 1), while dropout randomly disables neurons to promote more robust representations (present in both models). Both techniques improve model generalization. Furthermore, both models require the same amount of training time (when the GPU is available), which is approximately 1 minute per epoch.
In conclusion, it is reiterated that the accuracy of the models could be improved by better discriminating the initial dataset.