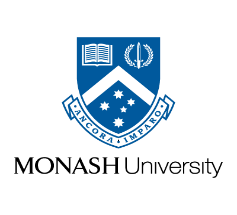
**KAGGLE COMPETITION REPORT**



**FIT 3181**

Student id: 30604265

Student name: Ayden Zein Khalil

Display name on Kaggle: Ayden Khalil

1. Set up

*tensorflow==2.10.0*

*numpy==1.23.3*

*pandas==1.5.0*

*imutils==0.5.4*

*ipykernel==6.15.3*

*notebook==6.4.12*

*opencv-python==4.6.0.66*

*scikit-learn==1.1.2*

*matplotlib==3.6.0*

*Codebase Guide:*

*GitHub link: https://github.com/AydenZK/kaggle3181*

***‘utils/data\_load.py’***

* *Loads in all the data expected to be saved in data/*
* *Extends on models.py*
* *These functions do the data loading and data manipulation ready for ML*

***‘utils/models.py’***

* *Baseline code provided by the teaching team*

***‘utils/solution.py’***

* *Contains the ‘create\_solution’ function which is able to be called after being provided a model. Makes predictions and then saves*

***‘main.ipynb’***

* *The main code runner, hyperparameter tuning, model extension notebook.*

1. Dataset

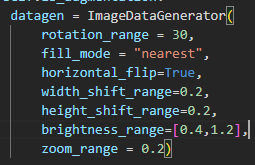
*A portion of the dataset code was utilised from the teaching team’s models.py file.*

*The train, valid, test split are as follows:*

**

*80% Training, 10% Test*

*It is important to mention that the out-of-sample test set was also saved and loaded in using the load\_test function in utils/data\_load.py.*

*Finally, my approach used iterations of grid search, some of which included data augmentation using the following schema:*

1. Methods
   1. Model architecture
      1. Chart

         Description automatically generatedSimilar to assignment (grid-search4.csv)

|  |  |
| --- | --- |
| Architecture | Additional notes |
| Input | 32x32 |
| Convolutional layers | 3x3, Strides=1, Padding = Same |
| Batch Normalisation | Momentum: 0.9 |
| Activation Functions | ReLU |
| Skip Connections | (True) |
| Average Pooling | Strides=2, Padding = Same |
| Dropout | Rate: 0.4 |
| Fully Connected (Flatten) | Units: 10, activation: softmax |

Feature Maps: 40, 80, 120  
 Optimiser: Adam

* + 1. Chart, bar chart

       Description automatically generatedDefault-Model

|  |  |
| --- | --- |
| Architecture | Additional notes |
| Input | 32x32 |
| Convolutional layers | 3x3, Strides=1, Padding = Same |
| Average Pooling | Strides=2, Padding = Same |
| Fully Connected (Flatten) | Units: 10, activation: softmax |

Feature Maps: 40, 80, 120  
 Optimiser: Adam

* 1. Techniques

*Text

Description automatically generatedData Augmentation:*

*As touched on previously, data augmentation was an important element of my solution. I have used rotations, width and height shifts, brightness and zoom dilutions as well as flips. The reasoning behind using this is due to the nature of the out of sample (Kaggle) test data which I analysed and compared carefully to the images in the training set. I could see that these data augmentations would align with the variety in the data of the out of sample test allowing the model to learn more effectively.*

1. Experiment and results
   1. Parameter Setting

I had 4 iterations of a lengthy grid search aided by using my local GPU. After each iteration I would a) train a model based on the best parameters, b) plot out the effects of the hyperparameters on the accuracies and c) analyse these plots to extend and tune the grid itself.

Text

Description automatically generatedChart, calendar, scatter chart

Description automatically generated

Above is an artifact of the second grid search alongside brief insights to tune the next grid.

* 1. Loss & training accuracy

*Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedGraphical user interface, chart

Description automatically generated*

Here are three model performances, we notice that train loss/train accuracy is very smooth while the validation performance is not that stable. The technique of early stopping was extremely useful here as the model would stop training after a failure to increase the validation accuracy after x number of epochs. (x was a tuneable hyperparameter).

We see that there are still traces of overfitting as we have accuracies of around 90 for train and 70 for validation and test.

1. Discussion and conclusion

*(Summarize things you have already learnt, problems you encounter and the way to solve them. Which technique(s) that you think is the most important for your implementation? What is the limitation of your method? Can we further improve the performance?)*

As touched on before, overfitting was a major issue as when submitted to Kaggle, most models would only receive an accuracy of 20-30 which reinforced the need to investigate techniques to reduce overfitting such as data augmentation, data mix-up etc…

I tried to counteract this by using more dropout, early stopping, and data augmentation however, with more time I would like to further analyse.

Given the low Kaggle accuracy there could also be an element of underfitting but also a discrepancy between the training data and the out-of-sample test data. Thus I would also spend some time exploring the limitations of my model and potentially using bigger models as well as more augmentation techniques to ensure that the training coincides with the test data on Kaggle.

1. References and Related works

* *Unit References, Assignment 1, Tutorials and Lectures*