# **Image Classification with Convolutional Neural Networks**

## **(Keras and/or TensorFlow)**

### Description

The second report of the image classification with convolutional Neural Networks is about discussing the results and analysis of problem 1 and 2.

The description of problem 1 is about using a dataset of image classification of not less then 5 classes and to perform different deep learning operations which also includes some preprocessing tasks as well. Some of which can be considered as optional, which actually depends on the nature of the dataset. Foe example, if the quantity of each class in a dataset is not balanced then therefore, there is a need to perform the operation to balance the dataset to reduce large biasness from the deep learning model. Then after performing certain required pre-processing on the dataset. We then will apply three different deep learning models, where each CNN (Convolutional Neural Network) algorithm possesses their own aspect of parameters and CNN architecture. In the end, we are required to evaluate the results of each of these three models.

The description of problem 2 states that we are required to perform evaluation on the above three models. Where evaluation metrics to determine the performances used are.

1. Confusion Metrics
2. Classification Report
3. Precision
4. Recall
5. F1-score
6. Support ROC/AUC
7. Precision-Recall curve

Then after that, another important task is to implement ensembling.

### Result Analysis

In this section, we are going to discus about the results and their analysis for both problems.

1. **Problem 1:** The results we gather in the problem 1 are based on three different models. Where the total number of epochs for each model are same: 50. Total 50 iterations were executed in order to assess the results. In the 1st model, we achieve the training accuracy up to 84% and validation accuracy up to 68 %. Which states that, if we want to achieve more promising results, we then have to add more iterations or on the other hand, we can either tune the hyper-parameters of our model. Which we did in the second model.

In the 2nd model, we tune the hyper-parameters of the previous network topology, so that we might get any chance to achieve more good results. After executing the model with 50 iterations or 50 epochs the performance we achieved had the train accuracy up to 85% and validation accuracy up to 69%. Hence, the clouds were a bit clear that tuning the hyper-parameters helped us to achieve some improved results. Therefore, we again tune the hyper-parameter and changes the architecture of our CNN model. Which eventually help us to achieve some good results. Where, the training accuracy was up to 96% and validation accuracy was up to 81%.

How great it would be if we are able to plot the results of the above three models. Certainly, with the help of python’s Matplotlib library, we plotted the graphs of training accuracy, validation accuracy, training loss and validation loss. Having said that, all the three models were being plotted parallel to each other. In the Fig 1 below, we can notice, in the graph of training and validation accuracy, the line plot of each model is almost like each other in parallel.

Let us consider the training accuracy graph first, the performance curve for each of the three models are similar where they are rising gradually to achieve the best accuracy unit. In the validation accuracy graph, the variations in models 1 and 2 are quite a bit much. Hence, it is also showing that the accuracy is also increasing, unlike in the 3rd model, where the performance increase at the exponential rate and then it remained flat.

In the loss graphs, we noticed that, in the training loss part, the loss decreased at the exponential rate and then it remained flat throughout the 50th iteration. Similar pattern was also being noticed in the validation loss, where we just saw a single large spike of loss, otherwise, the pattern remained almost as it is, just with a few variations.

Chart

Description automatically generated with medium confidence

Fig 1.0 (accuracy and loss performance for both, training and loss)

1. **Problem 2:** The second problem is about using different scoring metrics, mentioned above, and then to evaluate their results. Then after that, to apply ensemble average technique to again evaluate the results. Let us understand the performance achieved by evaluation metrics.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** | **Precision** | **Recall** | **F1-Score** | **AUC** |
| **ADAM** | **0.69** | **0.63** | **0.59** | **0.76** |
| **SGD** | **0.716** | **0.56** | **0.56** | **0.73** |
| **RMSprop** | **0.73** | **0.73** | **0.72** | **0.83** |

**It is to be noticed that all the evaluation metrics for each of the models are almost the same. Like considering the first model with the ADAM optimizer, the score for precision, recall and f1-score are almost the same. When considering other models, we can also experience the similar kind of pattern in results just like the ADAM.**

**The scoring metrics always play a crucial role, when performing the image classification task. Because, sometimes, it is not a good practice to always use the accuracy metrics to evaluate the test results.**

**Another part of this second problem is using the ensemble learning approach to evaluate the results of the models. This operation somehow becomes useful when we tend to use different models for a similar type of dataset. The technique for this operation is to gather the results from all the three models and then we sum up all the results and then evaluate their scores based on average grading. The score we are achieving through the technique of ensembling method is**

1. **Accuracy 🡪 0.18**
2. **Precision 🡪 0.18**
3. **Recall 🡪 0.18**
4. **F1-Score 🡪 0.17**

**The above results show us that the performance by evaluating the combining the results of each of the models and the average them gives us the less score in accuracy.**

### Conclusion

Last but not the least, every model has its own set of limits to train themselves for better results. By doing the hyper-parameter tuning and calculating their performances based on different performance metrics can gives us more prominent results, which is also based on the requirements. Every aspect of machine learning is considered the main part of the project. Especially building the model. Every Deep learning model is the engine of the project and data is the fuel. Less the quality of the fuel, lesser the performance of the car in output. Tuning the engine and enhancing the quality of the fuel can result in somewhat not equal but closer to ideal plan or dream. Where this concept can also be applied in deep learning also.