**IIIT Delhi MIDAS Summer Internship/RA Task 2021**

**Report**

(Task 2)

**Name**: Kiran Davuluri

**Email Id:** dkiran238@gmail.com

**Task**: 2

**Task domain**: CNN and classification of image datasets

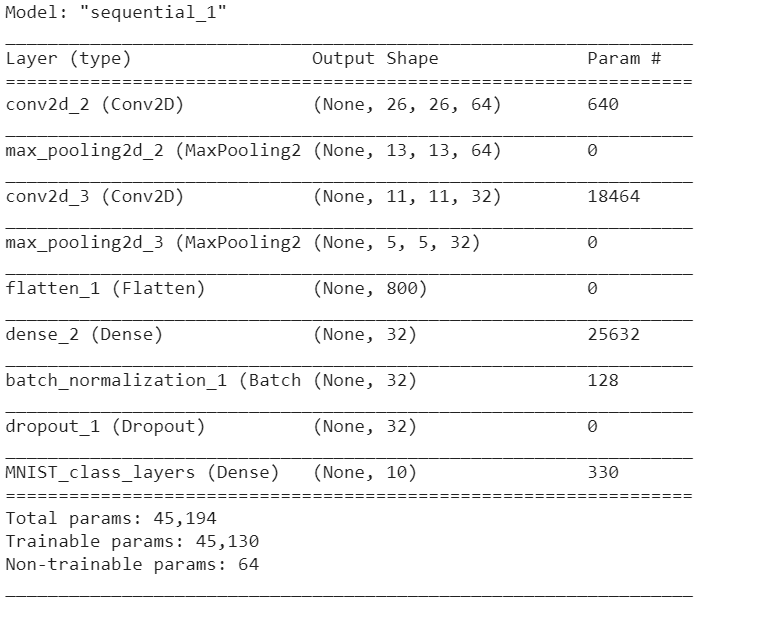
**Description:**

**Part 1:**

* Have used dataset (<https://www.dropbox.com/s/pan6mutc5xj5kj0/trainPart1.zip>) to train the model. The model architecture is described in the github repo.
* Prepocessing:
* Convert to grayscale
* Normalizing by dividing by 255
* Data Augmentation using ‘ImageDataGenerator’ class
* The ‘train:test:val’ ratio has been set at 8:1;1 so as to provide optimum results.
* Model Building:
* Conv2D, MaxPooling, Flatten and Dense layers have been used to build a simple model to optimize train and validation accuracy.
* Have experimented with various other architectures by adequately optimizing layers. This is the model with best overall accuracy scores.
* Training:
* Training of the model has been performed over the dataset for 150 epochs. This is since the model would be optimized by 150 epochs, and beyond that the model starts overfitting.
* Batch Size of 32 is taken and the images are sheared, zoomed and rotated with respective scales. The images are also flipped horizontally.
* The model is at last saved and h5 file is obtained.
* Checkpoints have been set to monitor the performance of the training and the best accuracy of the training has been taken.
* Initially the model accuracy was very low, which might be due to large size of images. But after reduction of input image size, there was significant increase in the accuracy.

**Part 2:**

Only 0-9 training images from the model from part 1 have been taken. The pretrained network in part 1 has been taken for training in part 2. The MNIST dataset has been taken to train the model. A randomly initialized weights model was taken to train on MNIST dataset.



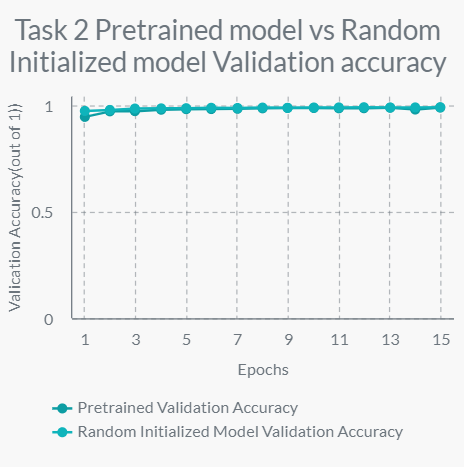
**Analysis for convergence:**

* + The second (randomly initialized weights) model converged faster than the first one (pretrained) , both trained on MNIST dataset.
  + This is because the pretrained model learnt poorly (around 67% accuracy). The pretrained model consisted of 62 classes. However the randomly initialized model had only 10 classes. The pretrained model had to extra-learn for 52 other classes as compared to the randomly initialized model.
  + Hence the rate of convergence for second model i.e. randomly initialized model was faster compared to that of pretrained model, trained on MNIST dataset.
  + Random initialized:-

Train\_ Accuracy: 98.39% Val\_Accuracy:-98.95%

* + Pretrained model:-

Train\_ Accuracy: 91.86% Val\_Accuracy:-98.76%



The accuracy seems to converge from 12th epoch for both the models.



**Quality Metrics:**

**F1 Score:**

* + - * Random Initialized Network: 0. 95
      * Pretrained model: 0.925

**Both have high precision and recall leading to high F1 score.**

**Part 3:**

` The following dataset ([https://www.dropbox.com/s/otc12z2w7f7xm8z/mnistTask3.zip](https://www.dropbox.com/s/otc12z2w7f7xm8z/mnistTask3.zip?dl=0)), has been taken to train the model. A pretrained model from part 1 was trained and another random initialized network was trained.

Preprocessing:

* Convert to grayscale
* Reduce the image size to (28, 28)
* Normalizing by dividing by 255
* Data Augmentation using ‘ImageDataGenerator’ class
* The ‘train:test:val’ ratio has been set at 8:1;1 so as to provide optimum results.

Model Building:

* The model trained in part 1 has been used to train in part 2

**Results and Accuracies:-**

* + Random initialized:-

Train\_ Accuracy: 10.67% Val\_Accuracy:-10.98%

* + Pretrained model:-

Train\_ Accuracy: 10.56% Val\_Accuracy:-10.75%

**Inference:**

Both the accuracies are same. The model is getting under fitted. The model from task 1 is not that complex enough for the model to learn better. More sample datasets are required for higher accuracy

The model has been tested on MNIST dataset and the results have been included in the ipynb. file.

**Conclusion:** Task 2 Part 1,2 and 3 have been successfully completed and the results have been obtained with apt inference.

**References:**

[1] <https://pypi.org/project/split-folders/>

[2] <https://nextjournal.com/gkoehler/digit-recognition-with-keras>

[3] <https://keras.io/examples/vision/mnist_convnet/>

[4] <https://keras.io/api/callbacks/model_checkpoint/>

[5] <https://stackoverflow.com/questions/54313305/load-parts-of-models-weights-from-previously-saved-hdf5-file>

[6] <https://stackoverflow.com/questions/47266383/save-and-load-weights-in-keras>

[7] <https://medium.com/apprentice-journal/evaluating-multi-class-classifiers-12b2946e755b>

[8] <https://create.piktochart.com/printable/editor/547>