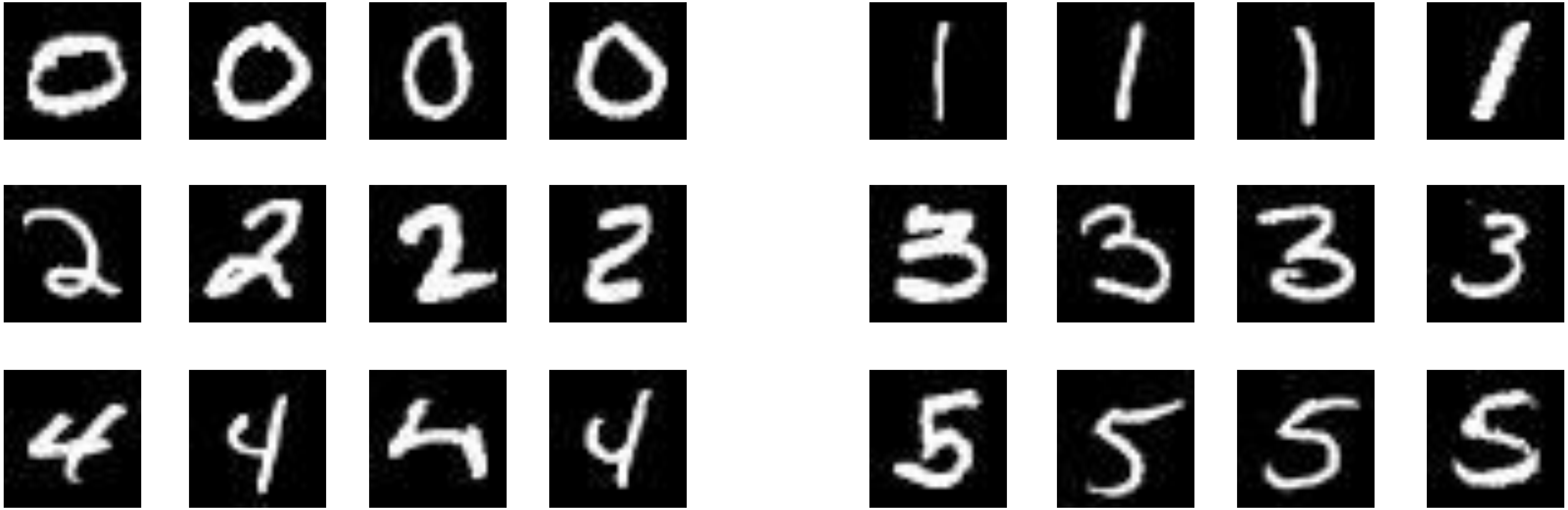
Python Tensorflow-Keras Trained Model For MNIST Hand Written Digits

Sanjay Singh

san.singhsanjay@gmail.com

Introduction & Dataset

- The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning.
- Following are some of its sample images:

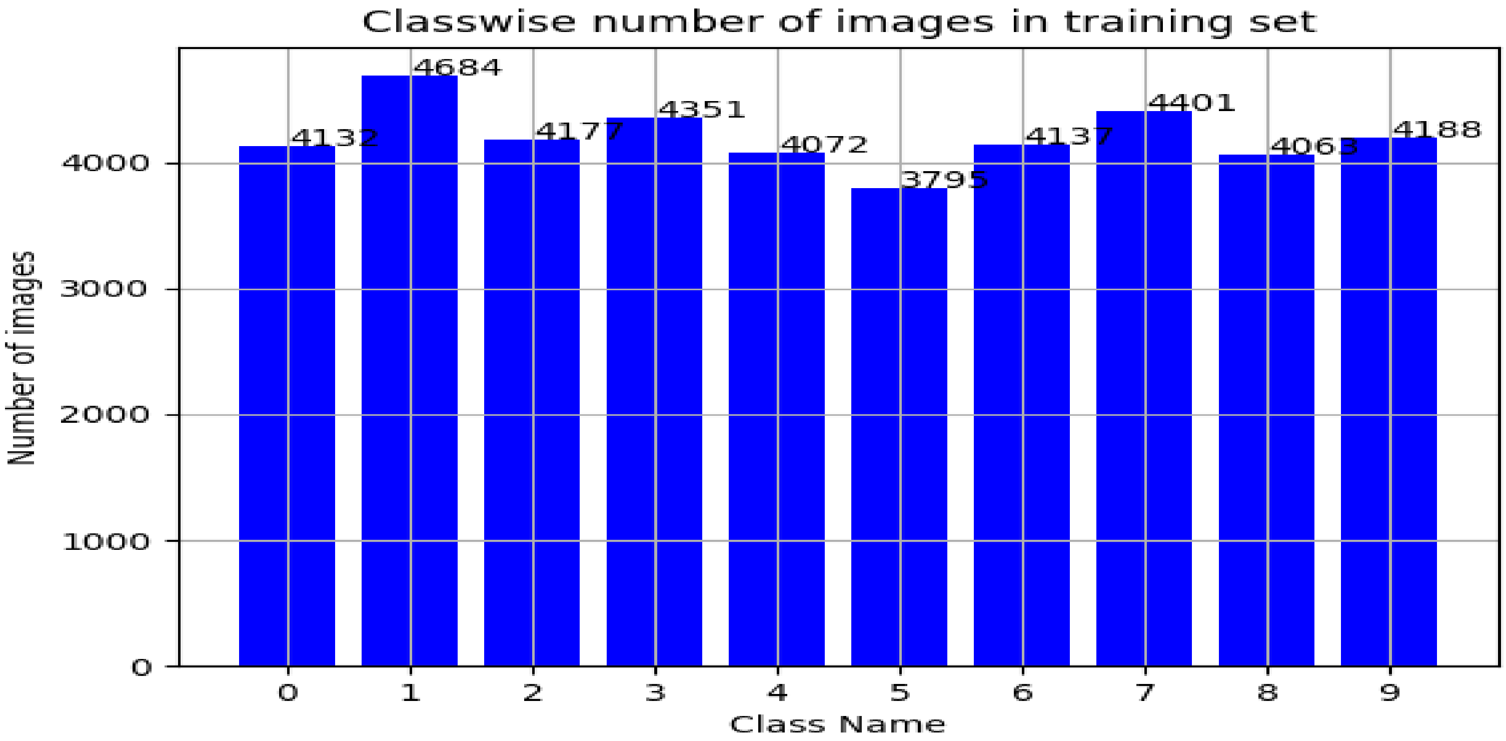


Introduction & Dataset…

- This dataset is downloaded from:
  https://www.kaggle.com/scolianni/mnistasjpg
- The size of this dataset is 38 MB.
- This dataset has total 70,010 images, out of which 28,000 images are given test set and 42,010 images are given in train set.
- Dimension of each given image is (28 x 28 x 3).

Introduction & Dataset…

- Following is a plot of provided training dataset:



- In case of test set, provided images are not labeled, thus it is hard to make classwise plot for test set.

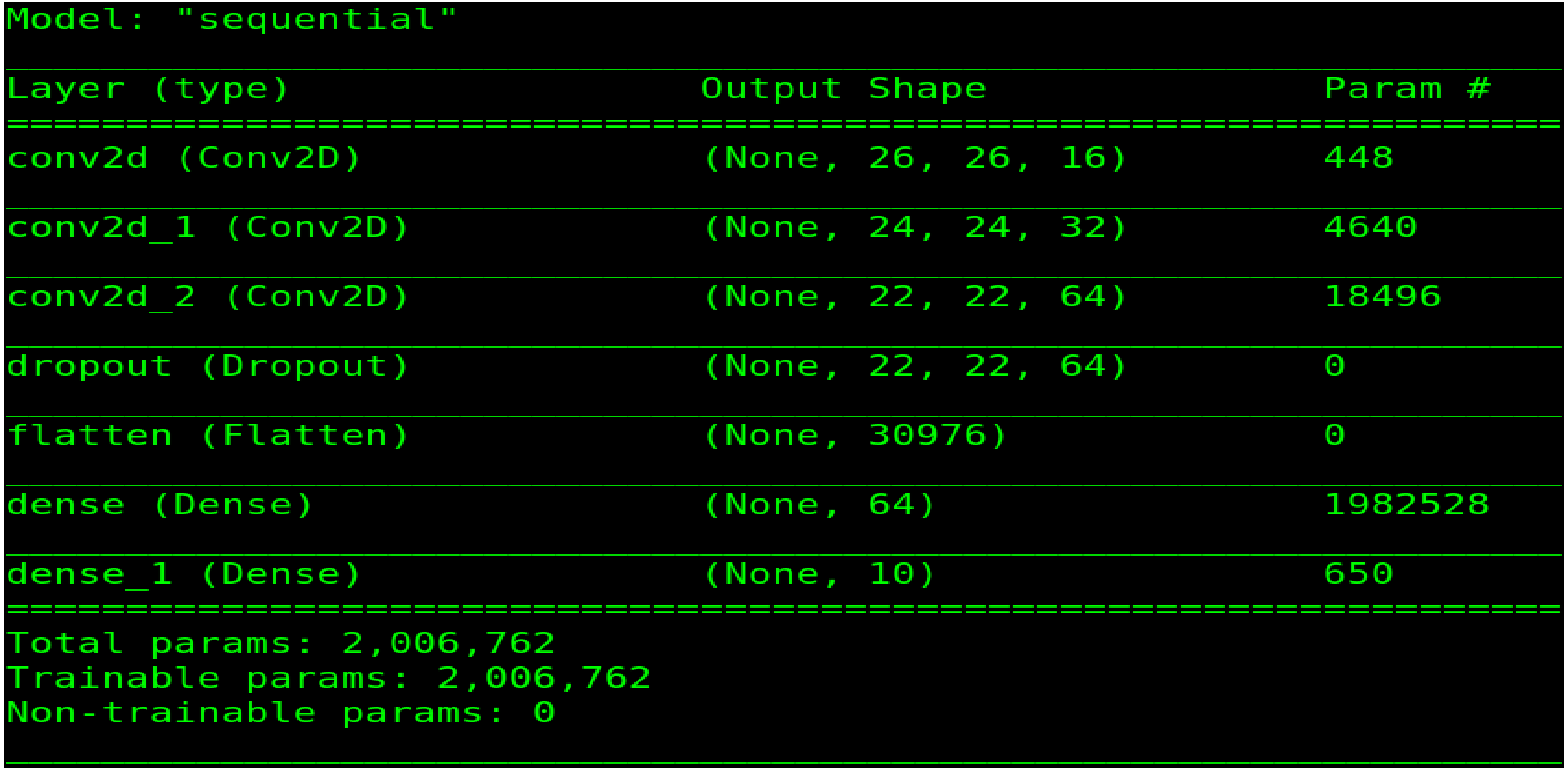
Scripts Execution Flow

- Following is the script execution flow:

1. “train_data_plot.py”: This script is generating a classwise plot of number of images in training set.
2. “make_train_val_test_set.py”: This script is segregating images for training, validation and test set. While doing this, this script is ensuring that the classes of the training and the validation dataset should be balanced.
3. “gen_train_val_plot.py”: This script is generating a plot for classwise number of images in training and validation dataset.
4. “train.py” and “train.ipynb”: This script is for training CNN on this MNIST hand written digits dataset. Final training is done by using “train.ipynb”, thus “train.ipynb” would be more updated than “train.py”.

CNN Architecture

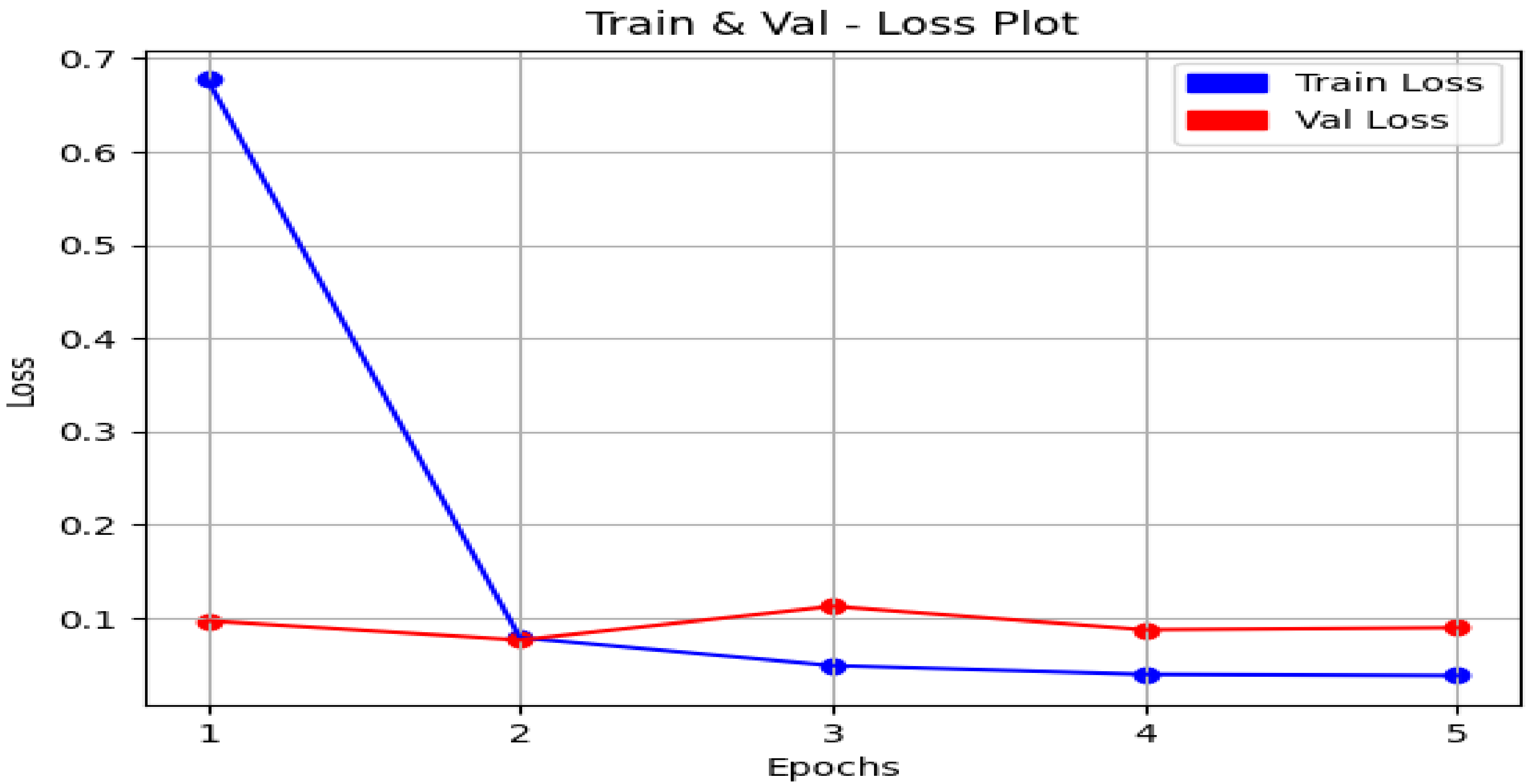
- Following is the CNN Architecture of model:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 16)        448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        4640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 22, 22, 64)        18496
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1982528
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 2,006,762
Trainable params: 2,006,762
Non-trainable params: 0
_________________________________________________________________
```

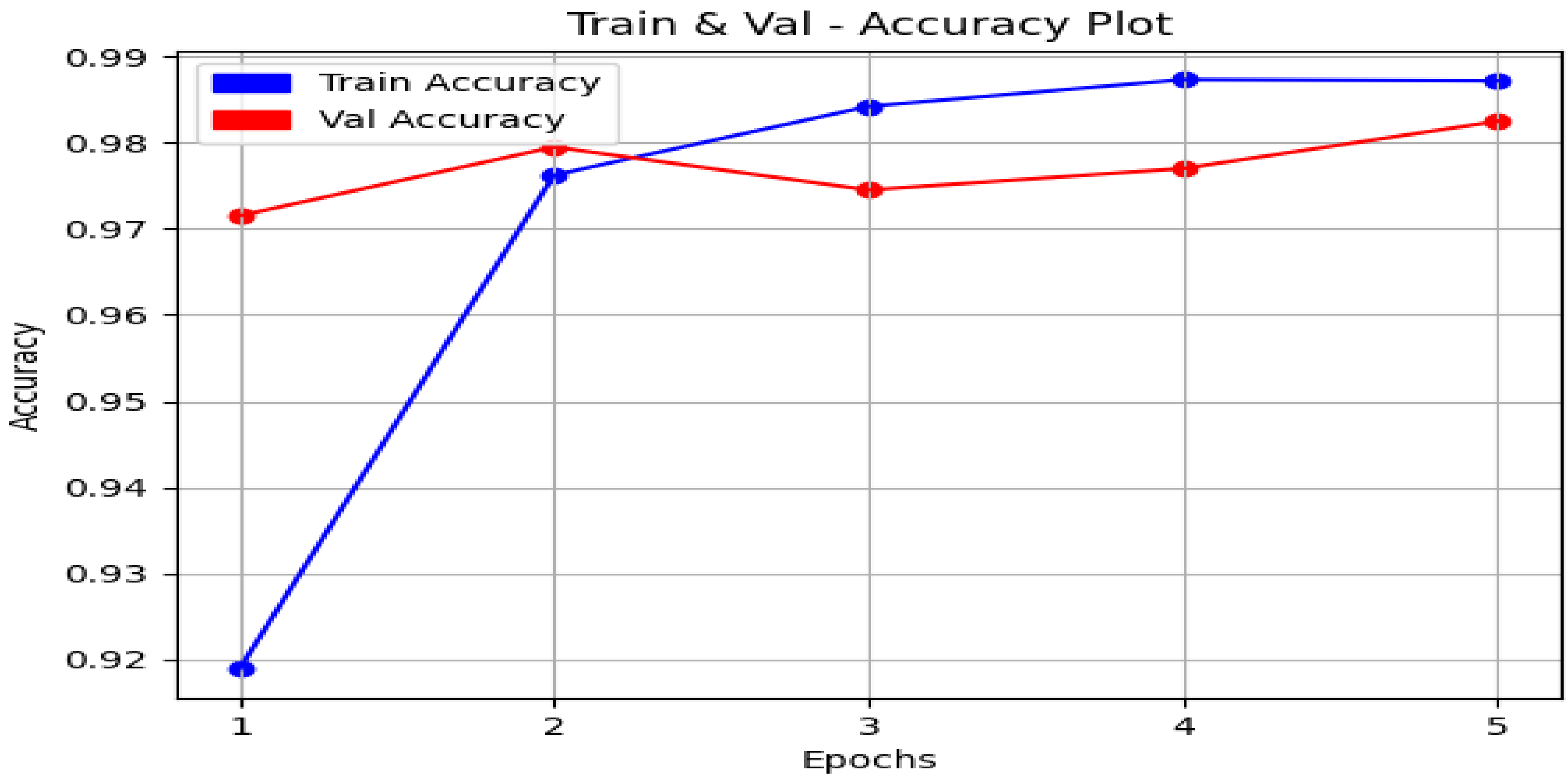
Training & Result

- Following is the plot of train loss and validation loss:



Training & Result…

- Following is the training and validation accuracy:



Training & Result…

- The model trained for 5 epochs.
- Following is the status after training of 5 epochs:
  - Train Accuracy: 98.72 %
  - Train Loss: 0.03841
  - Validation Accuracy: 98.25 %
  - Validation Loss: 0.08973

Template

- ➢ Template

Thank You