Name- Atul Priyank

Assessment : Labellerr.com - Image Classification Model

Thank You for this opportunity to give a brief detail of my assessment.

My system Requirements:

Operating system: Windows 11

Memory: 8 GB RAM (7.35 GB RAM usable)

Current resolution: 1920 x 1080, 144Hz

GPU: NVIDIA GeForce RTX 3050 Ti Laptop GPU

CPU: AMD Ryzen 5 5600H with Radeon Graphics

1) Dataset Acquired:

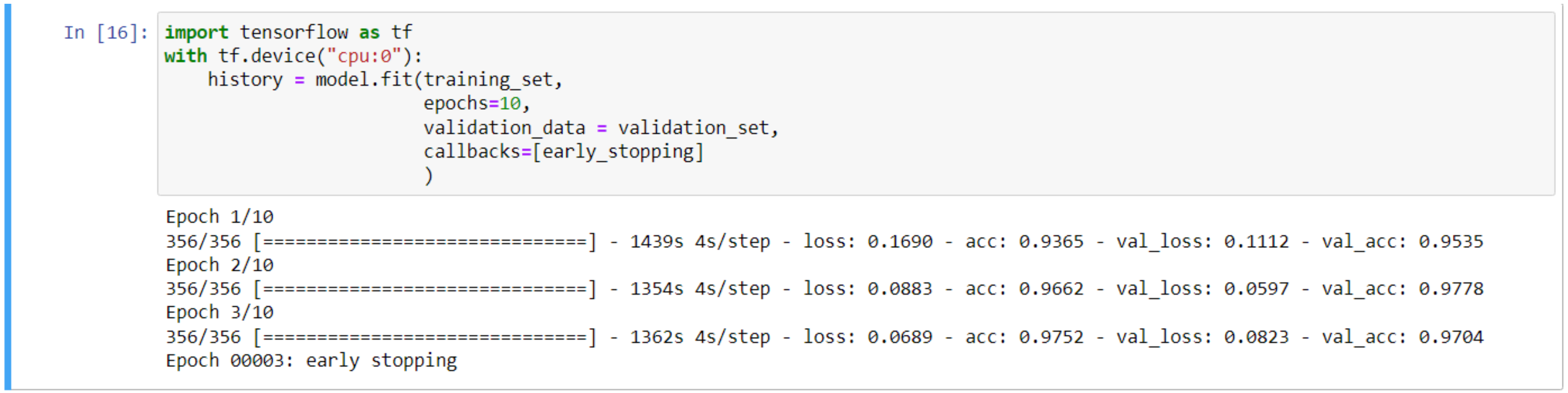
The dataset was acquired from Kaggle. Kaggle is actually a online community of machine learning, data scientists and many others to practice their way of with the real world data which is allowed to publish by the users. Data scientists, machine learning engineers can work on those data and build models in this data-science environment. The dataset includes Vehicles and non-vehicle photos are included in the vehicle detection image dataset. The folders are used to categorise the data. Up to x vehicle photographs and x non-vehicle images may be found in each folder. Images that are RGB have a size of (64x64x3).

a. Importing Necessary libraries:
   - OS : For directory managing
   - CV2 : For manage images
   - re : To manage filles
   - Pandas : To create dataframe
   - Matplolib : To visualize
   - Sklearn : For splitting and use confusion matrix
   - Tensorflow
   - Numpy : To do data structuring
   - Pandas : Data manipulation and analysis

2) Training the model:

For deep learning to work well on a specific task, selecting the appropriate optimizer is essential. There are various types of optimizers present but adam is the optimizer used here and loss categorical crossentropy because we use softmax activation function. The adam optimizer is used and early stopping callback function is also used. We also use validation set as validation data.

Early stopping is used to stop the model if the accuracy didn’t get high.

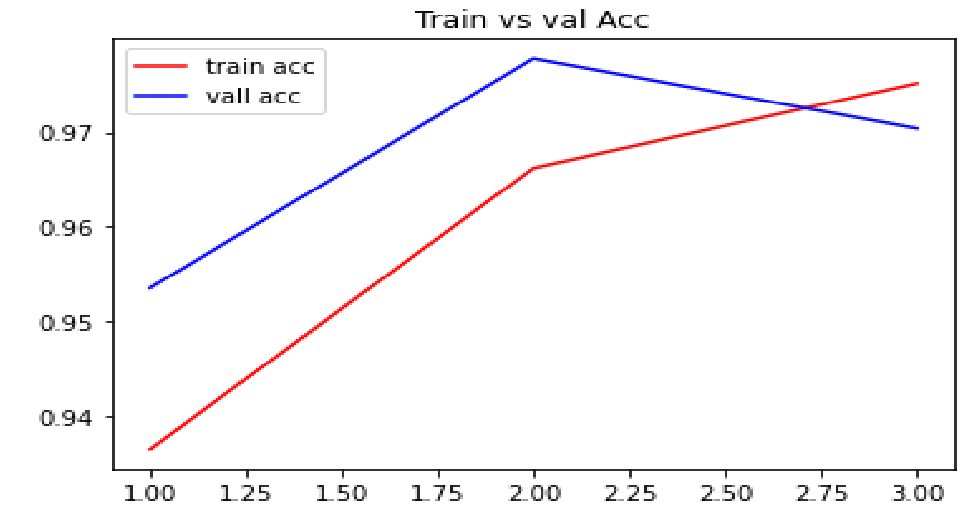


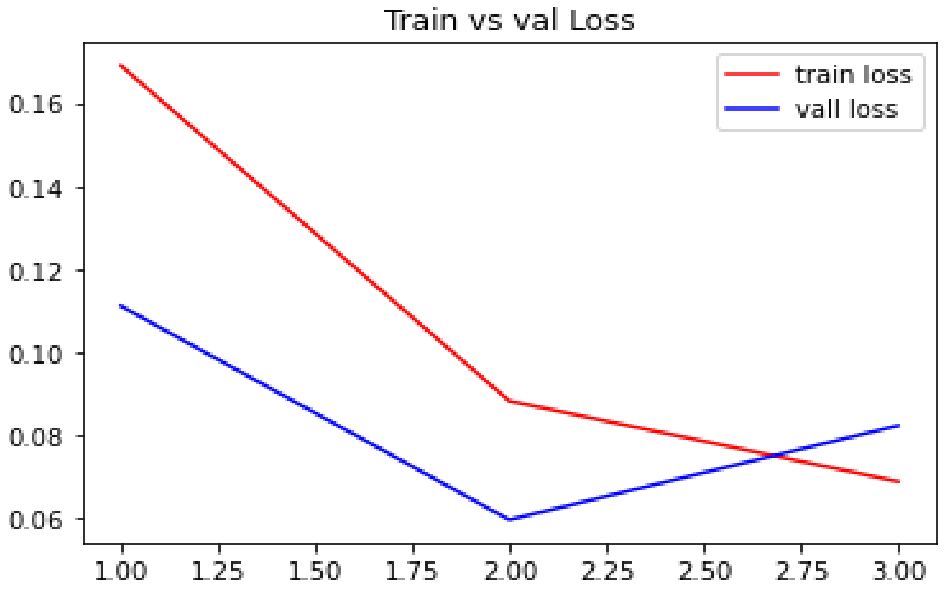
```
In [16]: import tensorflow as tf
         with tf.device("cpu:0"):
             history = model.fit(training_set,
                                 epochs=10,
                                 validation_data = validation_set,
                                 callbacks=[early_stopping]
                                 )
```

```
Epoch 1/10
356/356 [==============================] - 1439s 4s/step - loss: 0.1690 - acc: 0.9365 - val_loss: 0.1112 - val_acc: 0.9535
Epoch 2/10
356/356 [==============================] - 1354s 4s/step - loss: 0.0883 - acc: 0.9662 - val_loss: 0.0597 - val_acc: 0.9778
Epoch 3/10
356/356 [==============================] - 1362s 4s/step - loss: 0.0689 - acc: 0.9752 - val_loss: 0.0823 - val_acc: 0.9704
Epoch 00003: early stopping
```

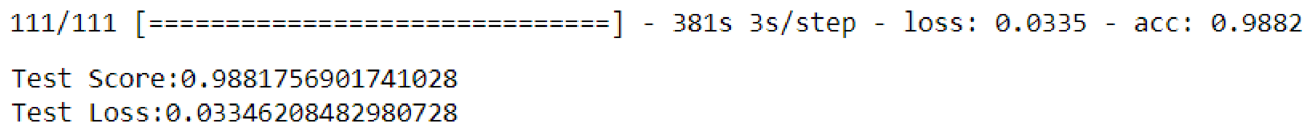
3) Evaluation of the model:

Here is the model’s training accuracy and loss graph with Adam optimizer using VGG-19.


Train vs val Acc
train acc
vall acc
0.97
0.96
0.95
0.94
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


Train vs val Loss
train loss
vall loss
0.16
0.14
0.12
0.10
0.08
0.06
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

Then the model is evaluated by using the model.evaluate() method:-



```
111/111 [==============================] - 381s 3s/step - loss: 0.0335 - acc: 0.9882

Test Score:0.9881756901741028
Test Loss:0.03346208482980728
```

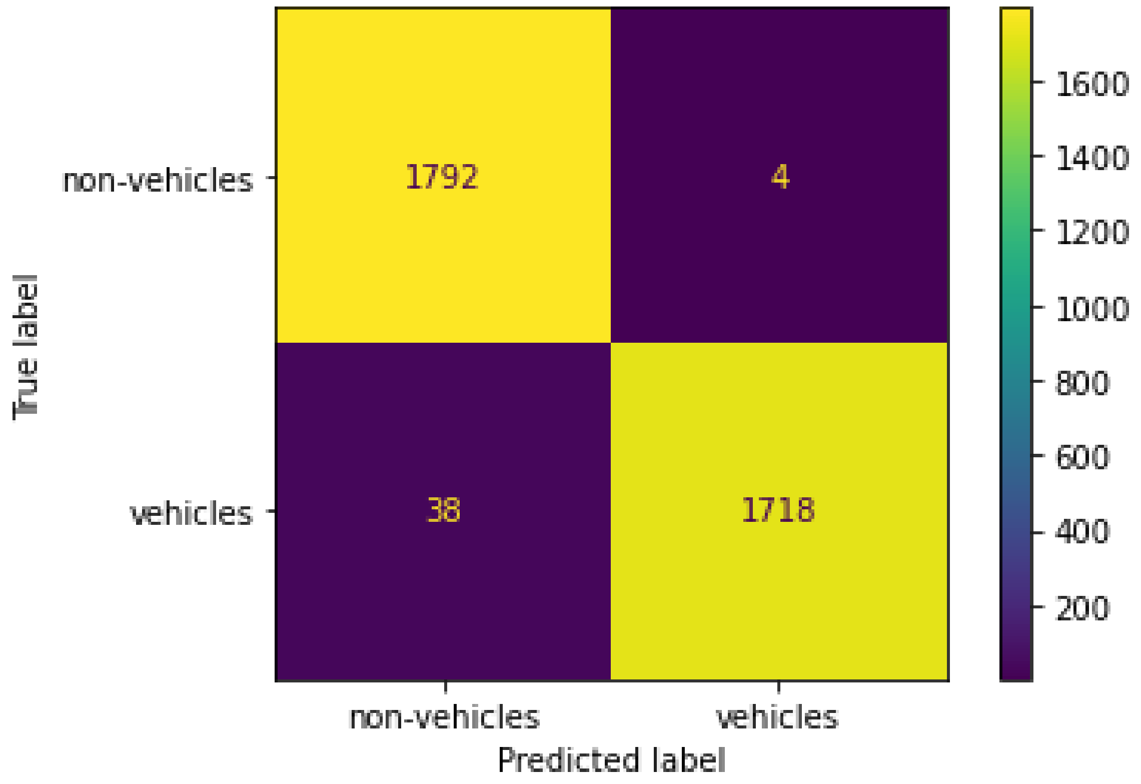
4) Performance Metrics:

Performance evaluation are a part of every day-to-day machine learning pipeline. Metrics are employed for monitoring and measuring the performance of the model. So, here is the performance metrics of Vgg-16 for Vehicles Image dataset.

- Accuracy Score: 0.9881756756756757
- Recall Score: 0.9783599088838268
- Precision Score: 0.9976771196283392

- F1-Score: 0.9879240943070731

Confusion Matrix:



We can see the True Positive(Tp) are 1792, False Positive(Fp) are 4, False Negative(Fn) are 38, True Negative(Tn) are 1718.

I used Jupyter Notebook for carrying out the code.

5) Analysis:

As we can see that the model performed really well. The model accuracy was around 98% and loss was around 6%. There might be some underfitting issues but we can see that the model stopped after the 3rd epoch. The pre-trained model gives good results, although some tweaks has to be made with the model for excellent results overcoming the fitting issues. The performance metrics showed good results.

Areas for Improvement:

- Need to do some more data preprocessing tasks
- Some tweaks to be made for the VGG-19 model.
- VGG-19 is kind of slow to train.
- VGG-19 takes quite a lot of disk space and bandwidth which makes it inefficient.