CNN PROJECT

TRANSFER LEARNING – ResNet50

(TensorFlow + Keras)

Performed By – PARTH SHARMA, AYUSH VASHISTHA

UCID – 31543562, 31537574

Mail ID – ps75@njit.edu , av64@njit.edu

Brief Summary -

Dataset that we have is having following structure:

1) Train Data
   - Audi
   - Lamborghini
   - Mercedes

2) Test Data
   - Audi
   - Lamborghini
   - Mercedes

We have divided this project working into 2 parts.

In Part-1,

(i) ResNet50 model has been built without using existing weights that comes with Transfer Learning model.
(ii) Later, model has been created and trained on Train Data.
(iii) Model has been saved.

In Part-2,

Saved model has been loaded again and then used to predict brand class name of a new example image.

Steps Included

1. Importing Tensorflow & Keras
2. Importing necessary modules from Keras
3. Importing other libraries
4. Setting Path for Train data & Test Data
5. Setting up ResNet 50 network pre-trained over ImageNet weights
6. Form our model by providing our own prediction layers to pre-trained network.
7. Summary of model
8. Compile the model
9. Fit the model using training data
10. Plot 'Train Loss' vs 'Validation Loss'
11. Plot 'Training Accuracy' vs 'Validation Accuracy'
12. Saving the model
13. Load the saved model
14. Load a new sample image
15. Convert img into an array
16. Normalize array
17. Preprocess your array
18. Predict the class of Car-Brand of test image using the saved model.

ENTIRE WORKFLOW

Step 1 – Import TensorFlow & Keras

- Check version of TensorFlow & Keras
- (NOTE – TensorFlow version should be atleast 2.2)

```
In [1]: import tensorflow as tf

In [2]: print(tf.__version__)
        2.2.0

In [49]: import keras

In [50]: print(keras.__version__)
         2.4.3
```

Step 2 – Import necessary modules from Keras.

```
In [4]: from keras.layers import Input,Lambda,Dense,Flatten
        from keras.models import Sequential
        from keras.applications.resnet50 import ResNet50
        # from keras.applications.vgg19 import VGG19
        from keras.preprocessing import image
        from keras.preprocessing.image import ImageDataGenerator,load_img
        from keras.models import load_model,Model
```

Step 3 - Importing other libraries

```
In [5]: import numpy as np
        from glob import glob
        import matplotlib.pyplot as plt
        %matplotlib inline
```

- Glob stands for Global
- It is used to return all the file paths that match a specific pattern.

Step 4 – Setting up size for an image

```
In [6]: img_size = [224,224]
```

Step 5 - Setting Path for Train data & Test Data

```
In [7]: train_path = 'Datasets/Train'

In [8]: test_path = 'Datasets/Test'
```

Step 6 - Setting up ResNet 50 network pre-trained over imagenet weights

```
In [9]: resnet = ResNet50(input_shape=img_size+[3],weights='imagenet',include_top=False)

In [10]: resnet.summary()
```

Step 7 – Form our model by providing our own prediction layers to pre-trained network.

```
In [15]: x = Flatten()(resnet.output)

In [16]: prediction = Dense(len(folders),activation='softmax')(x)

In [17]: model = Model(inputs = resnet.input, outputs=prediction)
```

Step 8 – Compile Model

```
In [19]: model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

Step 9 – Setting up Training Data & Test Data

```
In [20]: train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)

         test_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

```
In [21]: train_data = train_datagen.flow_from_directory('Datasets/Train',
                                                       target_size=(224,224),
                                                       batch_size=32,
                                                       class_mode='categorical'
                                                      )
```

```
Found 64 images belonging to 3 classes.
```

```
In [22]: test_data = test_datagen.flow_from_directory('Datasets/Test',
                                                     target_size=(224,224),
                                                     batch_size=32,
                                                     class_mode='categorical')
```

```
Found 59 images belonging to 3 classes.
```

Step 10 – Fit the model over Training Data

```
In [23]: r = model.fit_generator(train_data,
                                 validation_data=test_data,
                                 epochs=50,
                                 steps_per_epoch = len(train_data),
                                 validation_steps=len(test_data))
```

```
Please use Model.fit, which supports generators.
Epoch 1/50
2/2 [==============================] - 5s 2s/step - loss: 3.5569 - accuracy: 0.3438 - val_loss: 14.8280 - val_accuracy: 0.322
Epoch 2/50
2/2 [==============================] - 4s 2s/step - loss: 9.5810 - accuracy: 0.3906 - val_loss: 5.8801 - val_accuracy: 0.1864
Epoch 3/50
2/2 [==============================] - 4s 2s/step - loss: 2.8016 - accuracy: 0.4062 - val_loss: 5.2977 - val_accuracy: 0.5085
Epoch 4/50
2/2 [==============================] - 4s 2s/step - loss: 6.0330 - accuracy: 0.3281 - val_loss: 3.0278 - val_accuracy: 0.5085
Epoch 5/50
2/2 [==============================] - 4s 2s/step - loss: 2.6525 - accuracy: 0.3594 - val_loss: 3.5332 - val_accuracy: 0.3559
Epoch 6/50
```

.....

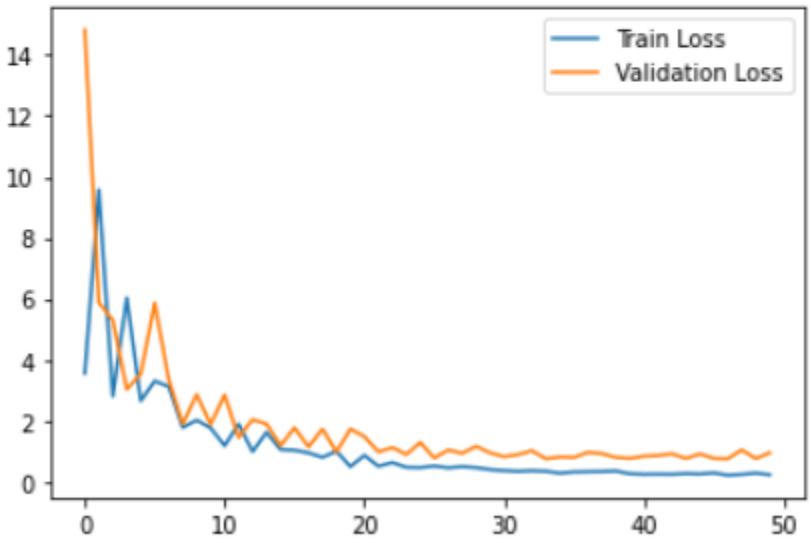
.....

.....

```
Epoch 45/50
2/2 [==============================] - 5s 2s/step - loss: 0.2496 - accuracy: 0.9375 - val_loss: 0.9103 - val_accuracy: 0.6271
Epoch 46/50
2/2 [==============================] - 5s 2s/step - loss: 0.2842 - accuracy: 0.9219 - val_loss: 0.7621 - val_accuracy: 0.7288
Epoch 47/50
2/2 [==============================] - 5s 2s/step - loss: 0.2112 - accuracy: 0.9219 - val_loss: 0.7498 - val_accuracy: 0.6610
Epoch 48/50
2/2 [==============================] - 5s 2s/step - loss: 0.2356 - accuracy: 0.9375 - val_loss: 1.0398 - val_accuracy: 0.6271
Epoch 49/50
2/2 [==============================] - 5s 2s/step - loss: 0.2774 - accuracy: 0.8594 - val_loss: 0.7566 - val_accuracy: 0.7458
Epoch 50/50
2/2 [==============================] - 5s 2s/step - loss: 0.2248 - accuracy: 0.9531 - val_loss: 0.9461 - val_accuracy: 0.7288
```

Step 11- Plot Loss

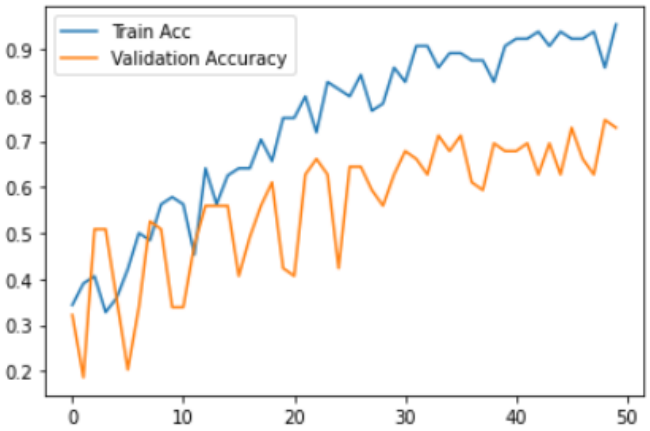
In [24]:
```
# Plot loss
plt.plot(r.history['loss'], label='Train Loss')
plt.plot(r.history['val_loss'], label = 'Validation Loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```




```
<Figure size 432x288 with 0 Axes>
```

Step 12 – Plot Accuracy

```
In [25]: plt.plot(r.history['accuracy'], label='Train Acc')
         plt.plot(r.history['val_accuracy'],label='Validation Accuracy')
         plt.legend()
         plt.show()
```



Step 13 – Save the Model

Saving the model

```
In [27]: from keras.models import load_model

In [28]: model.save('resnet50_car.h5')
```

Step 14 – Model Prediction using Test data

```
In [29]: y_pred = model.predict(test_data)

In [30]: y_pred

Out[30]: array([[1.51304007e-01, 3.11928988e-01, 5.3676
```

```
In [31]: y_pred = np.argmax(y_pred,axis=1)

In [32]: y_pred

Out[32]: array([2, 1, 2, 2, 1, 2, 0, 2, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1,
                0, 0, 2, 2, 2, 0, 2, 1, 1, 0, 2, 1, 1, 1, 2, 2, 1, 0, 2, 2, 2, 1,
                2, 1, 2, 1, 0, 2, 2, 1, 1, 0, 1, 1, 0, 1, 1], dtype=int64)
```

Step 15 – Loading saved model

```
In [33]: model = load_model('resnet50_car.h5')
```

Step 16 – Loading a sample image to make a prediction

```
In [33]: model = load_model('resnet50_car.h5')

In [34]: img = image.load_img('C:/Users/smast/Pytho

In [35]: img
```

Out[35]:

Step 17 – Converting ‘img’ into array

```
In [36]: x = image.img_to_array(img)

In [37]: x

Out[37]: array([[[124., 115., 108.],
                 [ 85.,  76.,  69.],
                 [112., 103.,  96.],
                 ...,
                 [221., 224., 229.],
                 [224., 227., 232.],
                 [231., 234., 239.]],

                [[130., 121., 114.],
                 [ 90.,  81.,  74.],
                 [116., 107., 100.],
                 ...,
                 [221., 224., 229.],
                 [225., 228., 233.],
```

Step 18 – Normalize img_array to keep values in (0-1) range.

```
In [38]: x = x/255
         x

Out[38]: array([[[0.4862745 , 0.4509804 , 0.42352942],
                 [0.33333334, 0.29803923, 0.27058825],
                 [0.4392157 , 0.40392157, 0.3764706 ],
                 ...,
                 [0.8666667 , 0.8784314 , 0.8980392 ],
                 [0.8784314 , 0.8901961 , 0.9098039 ],
                 [0.90588236, 0.91764706, 0.9372549 ]],

                [[0.50980395, 0.4745098 , 0.44705883],
                 [0.3529412 , 0.31764707, 0.2901961 ],
```

```
In [39]: x.shape

Out[39]: (224, 224, 3)
```

Step 19 – preprocessing image array

```
In [40]: x = np.expand_dims(x,axis=0)

In [41]: x.shape
Out[41]: (1, 224, 224, 3)

In [42]: from keras.applications.resnet50 import preprocess_input

In [43]: img_data = preprocess_input(x)

In [44]: img_data

In [45]: img_data.shape
Out[45]: (1, 224, 224, 3)
```

Step 20 – Predict brand Class for car image

```
In [46]: model.predict(img_data)
Out[46]: array([[0.03919638, 0.3698617 , 0.59094197]], dtype=float32)

In [47]: a = np.argmax(model.predict(img_data),axis=1)

In [48]: a
Out[48]: array([2], dtype=int64)
```

Step 21 – Created Flask App (app.py file)

Here we will be choosing a random image file on our Flask App and our saved model will be giving its prediction over chosen image.

(Build this just for expanding our skills, otherwise there was no need of this Flask App to be included in this project.)

We felt this Flask App will provide more user-centric approach to the created model.

```
C:\Users\smast>cd C:\Users\smast\Python Projects\Deep-Learning-Car-Brand-master

C:\Users\smast\Python Projects\Deep-Learning-Car-Brand-master>python app.py
```

RESULTS

127.0.0.1:5000

AI Demo

Image Classifier

Choose...

Predict!

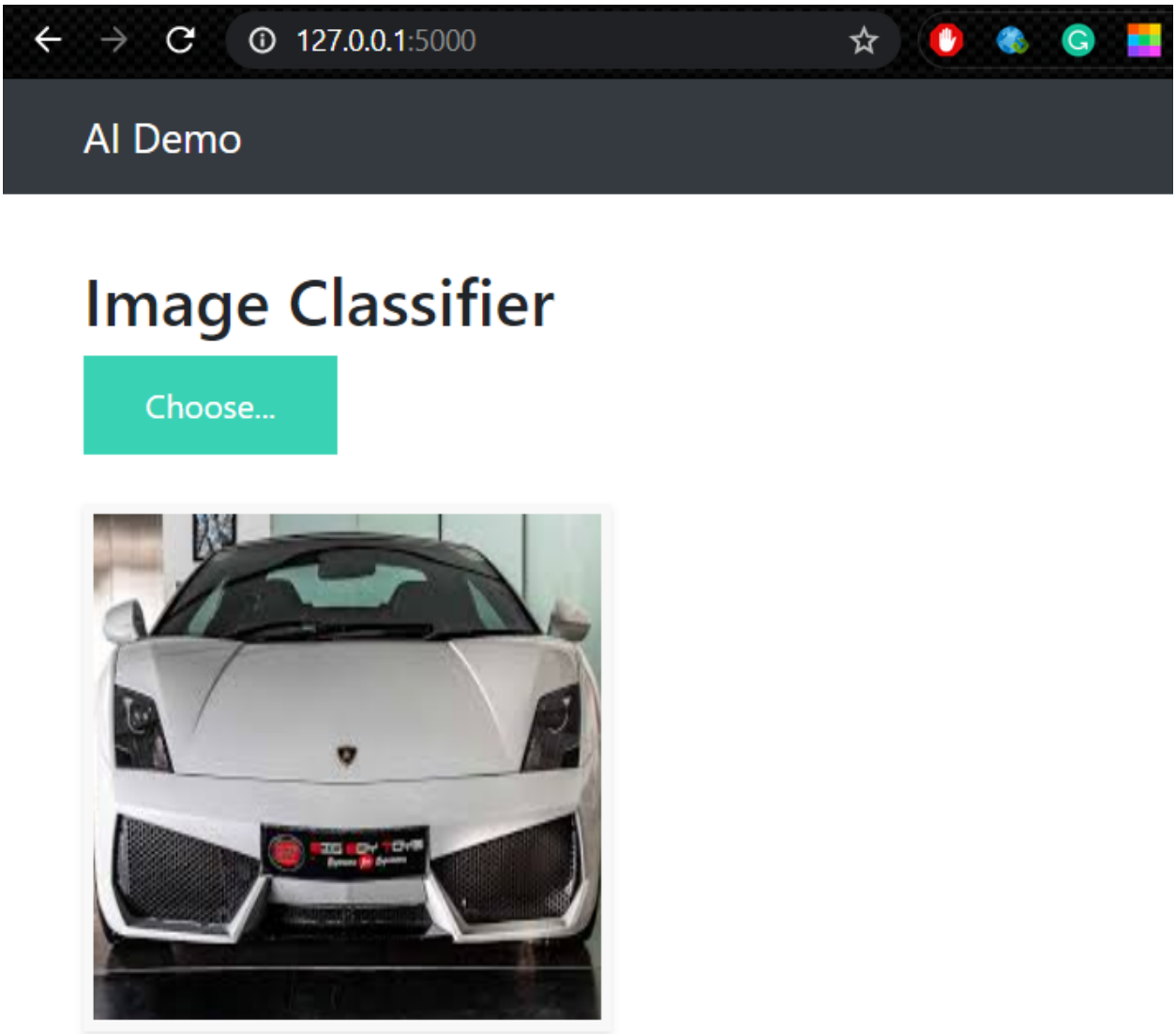
127.0.0.1:5000

AI Demo

Image Classifier

Choose...

Result: The Car is Mercedes



NOTE – 1) To run this program as a python file in your local env use – Resnet50_Final_Project.py

Command – python Resnet50_Final_Project.py

NOTE – 2) To run Flask app, use command – python app.py

1. Command to create Virtual Env with TensorFlow & Keras.

```
conda create -n yourenvname python=x.x anaconda
```

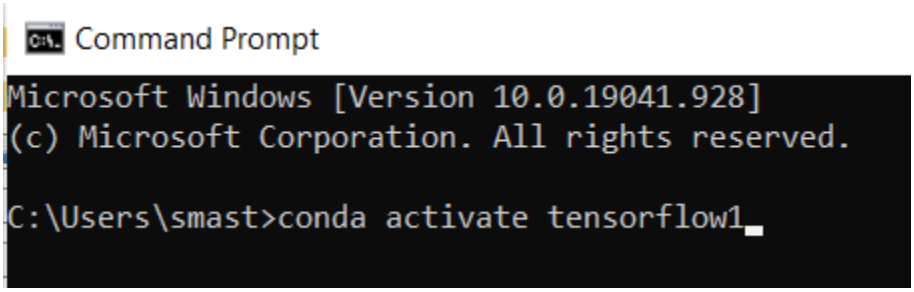
2. Install Tensorflow in your environment

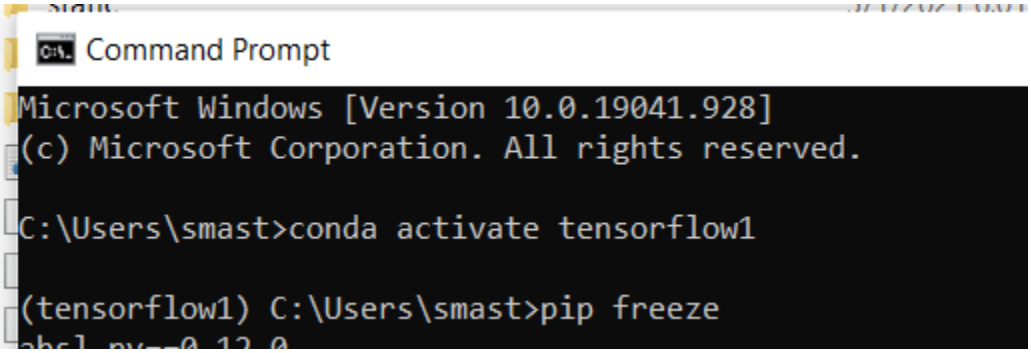
    - conda install tensorflow

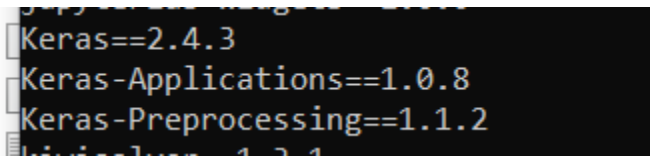
    - conda install keras

3. To Check all your libraries present in virtual env

    - pip freeze







```
tensorflow==2.2.0
tensorflow-estimator==
```

```
(tensorflow1) C:\Users\smast>cd C:\Users\smast\OneDrive\Documents\Data Mining\P

(tensorflow1) C:\Users\smast\OneDrive\Documents\Data Mining\Parth_Sharma_Final_
```