Jose Tenorio

Transfer Learning

EE 5353 Neural Networks

Introduction:

The purpose of this program is to utilize the free cloud service Google Colab. This service allows the user two write and run executable documents. Google Colab connects the user's notebook to a cloud base runtime and execute python code without any setup on the user's machine.

This program executes python code on **Google Colab** that identifies imagines using **convolutional neural nets** (CNN). CNN's can capture spatial and temporal dependencies of an image through the application of filters. Convolutional neural nets reduce the image into a form which is easier to process without losing features which are critical for a good prediction. This process is done with the use of the **Kernel filter**. This filter is also able to extract high-level features such as edges from the input image. The **pooling layer** is responsible for reducing the spatial size of the convolved feature. This helps decrease the computational power required to process the data. This layer is also useful for extracting dominant features which are rotational and positional invariant. Lastly, the **dropout layer** refers to ignoring units during the training phase of certain neurons which are selected at random. By ignoring, I mean they are not considered during a particular forward or backward pass. We do this to prevent over-fitting.

Procedure:

1. Create a CNN with the layers mentioned below. Include the Image augmentation code which inputs image directly from directory and write the prediction code.

Task 1 (training, validation and testing data is independent)
Create a CNN It should with the following layers
- Convolutional layer with 128 filters, Size of the filters is 3, 3, stides = 1 and relu activation
- Convolutional layer with 128 filters, Size of the filters is 3, 3, stides = 1 and relu activation
- Pooling layer with pool size 3,3
- Convolutional layer with 256 filters, Size of the filters is 3, 3, stides = 1 and relu activation
- Convolutional layer with 256 filters, Size of the filters is 3, 3, stides = 1 and relu activation
- Flattening
- Dense layer fully connected with 64 hidden units and relu activations
- Dropout layer with rate 0.5
- Dense layer fully connected with 64 hidden units and relu activations
- Dropout layer with rate 0.5
- Final dense fully connected layer with number of classes and softmax activation.

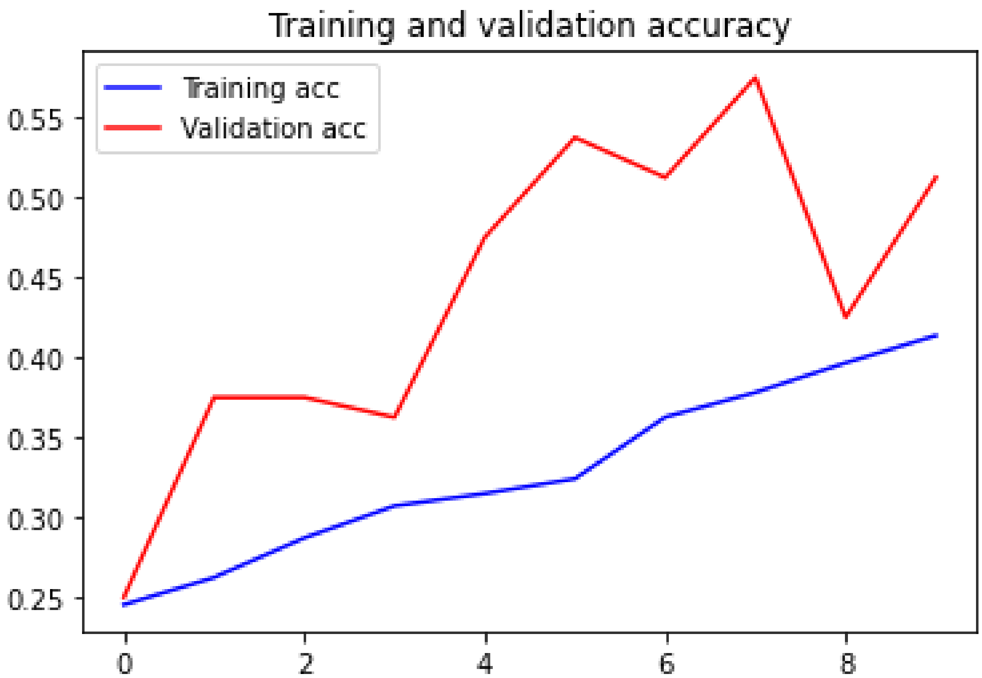
2. Create transfer learning python file and write the prediction code.

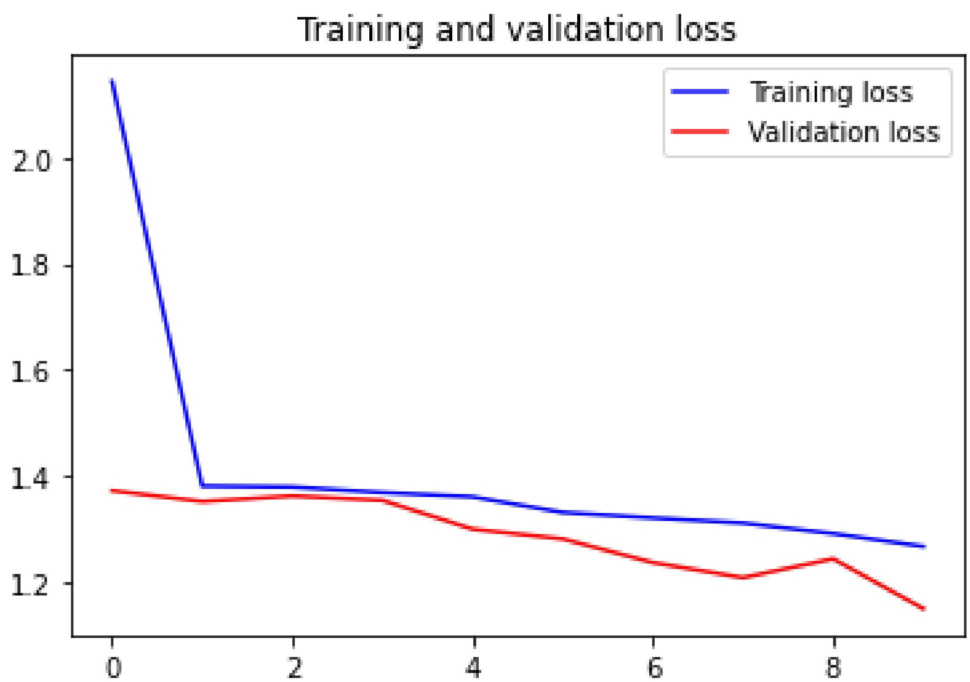
Part I CNN with Augmentation Training

Training Results:

```
#################################################################
Load Datasets

Data Directory List 1- >  /content/drive/My Drive/Colab
Notebooks/Program9/cats_dogs_horse_humans/train
Data Directory List 2- >  /content/drive/My Drive/Colab
Notebooks/Program9/cats_dogs_horse_humans/validation
Data Directory List 3- >  /content/drive/My Drive/Colab
Notebooks/Program9/cats_dogs_horse_humans/test

#################################################################
Create Model

Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 254, 254, 128)     3584
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 252, 252, 128)     147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 84, 84, 128)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 82, 82, 256)       295168
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 80, 80, 256)       590080
_________________________________________________________________
flatten_3 (Flatten)          (None, 1638400)           0
_________________________________________________________________
dense_9 (Dense)              (None, 64)                104857664
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_10 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 260
=================================================================
Total params: 105,898,500
Trainable params: 105,898,500
Non-trainable params: 0
_________________________________________________________________
Found 648 images belonging to 4 classes.
Found 80 images belonging to 4 classes.

#################################################################
Train Model

Epoch 1/10
12/12 [==============================] - 926s 71s/step - loss: 2.8518 - acc: 0.2650 - val_loss:
1.3720 - val_acc: 0.2500
Epoch 2/10
12/12 [==============================] - 922s 71s/step - loss: 1.3751 - acc: 0.2885 - val_loss:
1.3515 - val_acc: 0.3750
Epoch 3/10
12/12 [==============================] - 898s 69s/step - loss: 1.3696 - acc: 0.2868 - val_loss:
1.3619 - val_acc: 0.3750
Epoch 4/10
12/12 [==============================] - 899s 69s/step - loss: 1.3534 - acc: 0.3309 - val_loss:
1.3537 - val_acc: 0.3625
Epoch 5/10
12/12 [==============================] - 905s 70s/step - loss: 1.3593 - acc: 0.3064 - val_loss:
1.2994 - val_acc: 0.4750
Epoch 6/10
```

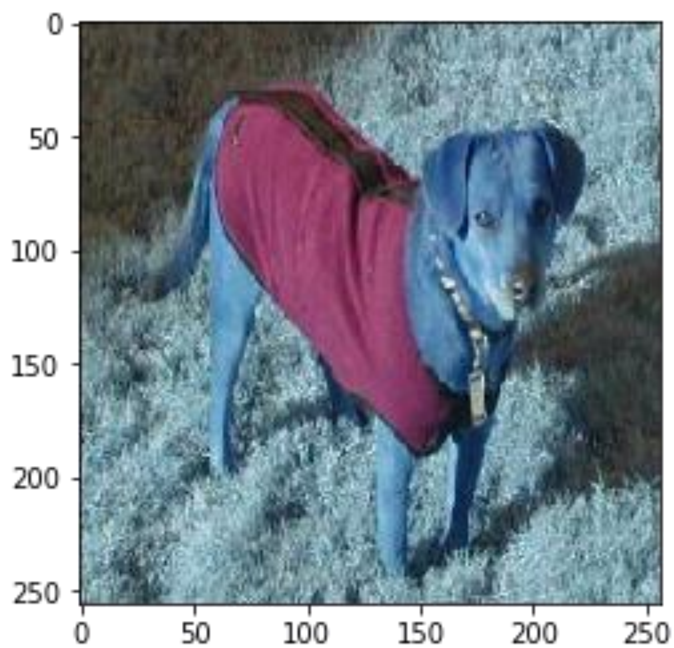
```
12/12 [==============================] - 908s 70s/step - loss: 1.3224 - acc: 0.3328 - val_loss:
1.2810 - val_acc: 0.5375
Epoch 7/10
12/12 [==============================] - 907s 70s/step - loss: 1.3186 - acc: 0.3706 - val_loss:
1.2363 - val_acc: 0.5125
Epoch 8/10
12/12 [==============================] - 909s 70s/step - loss: 1.3091 - acc: 0.3473 - val_loss:
1.2078 - val_acc: 0.5750
Epoch 9/10
12/12 [==============================] - 900s 70s/step - loss: 1.2747 - acc: 0.3918 - val_loss:
1.2434 - val_acc: 0.4250
Epoch 10/10
12/12 [==============================] - 904s 70s/step - loss: 1.2861 - acc: 0.4014 - val_loss:
1.1495 - val_acc: 0.5125
```



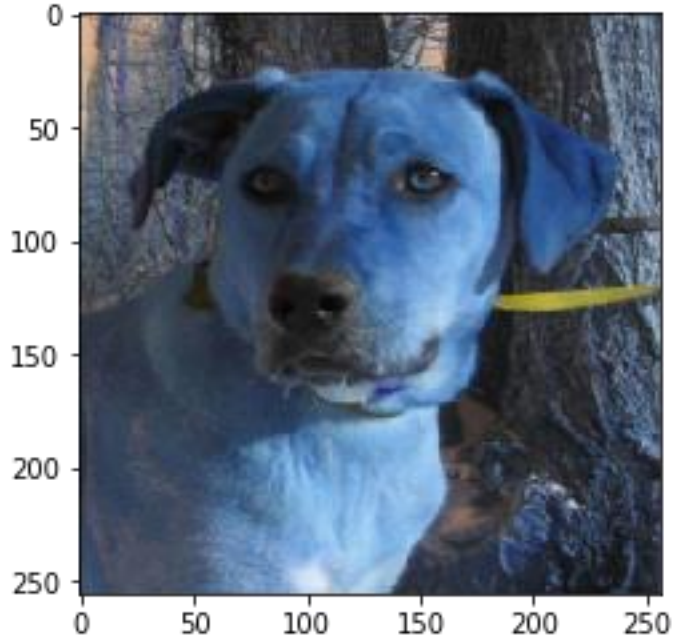


Testing Results:

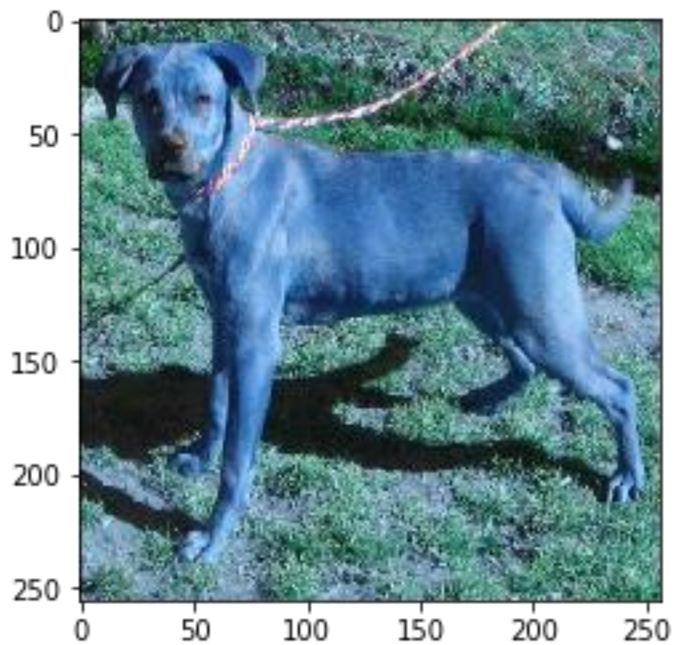
```
######################################################################
Test model with testing data

3/3 [==============================] - 29s 9s/step - loss: 1.3721 - acc: 0.3000
Testing accuracy - > 30.000001192092896

The Predicted Testing image is = humans verify below
```



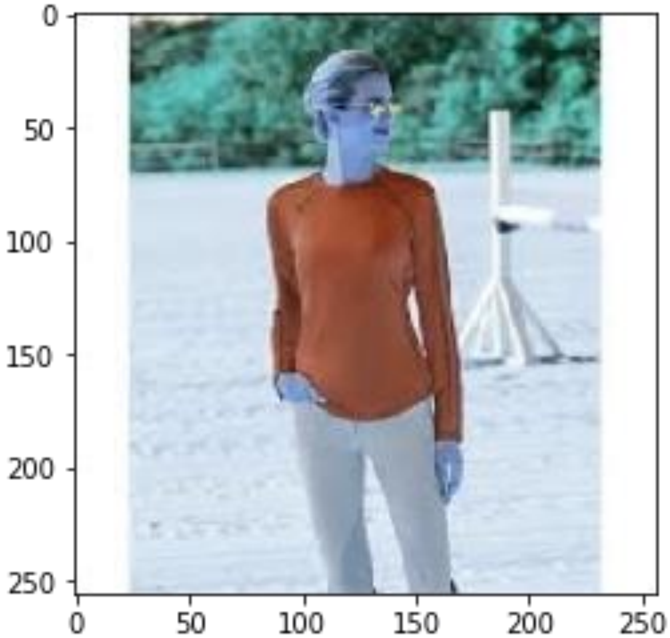

```
The Predicted Testing image is = dogs verify below
```



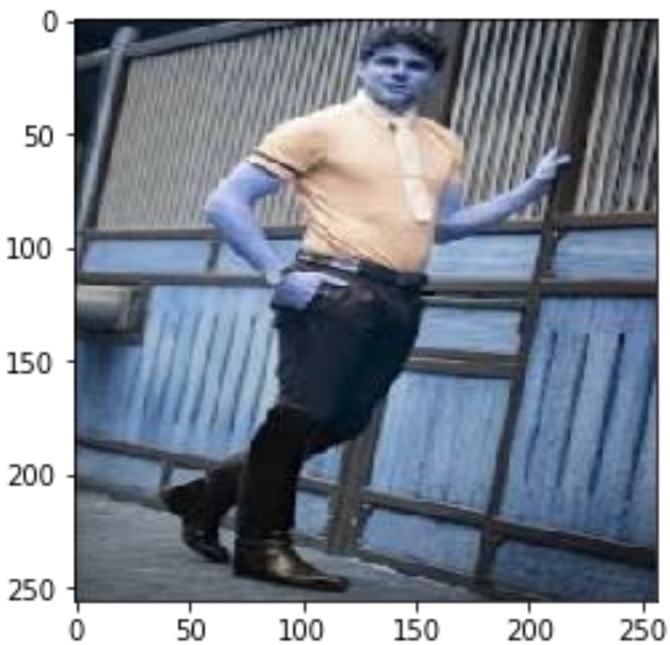
The Predicted Testing image is = humans verify below



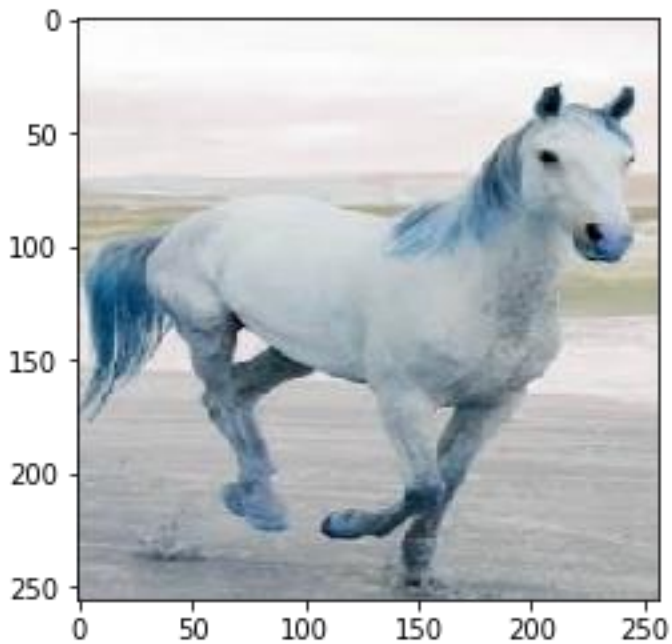
The Predicted Testing image is = horses verify below



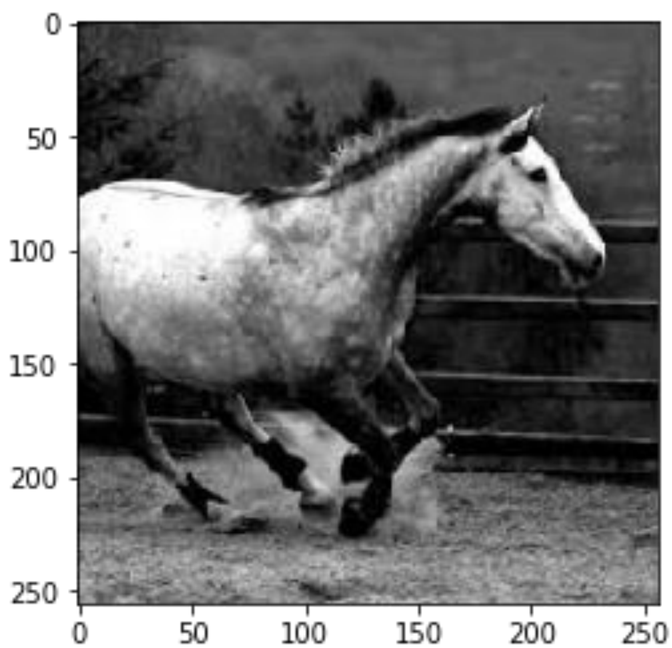
The Predicted Testing image is = dogs verify below



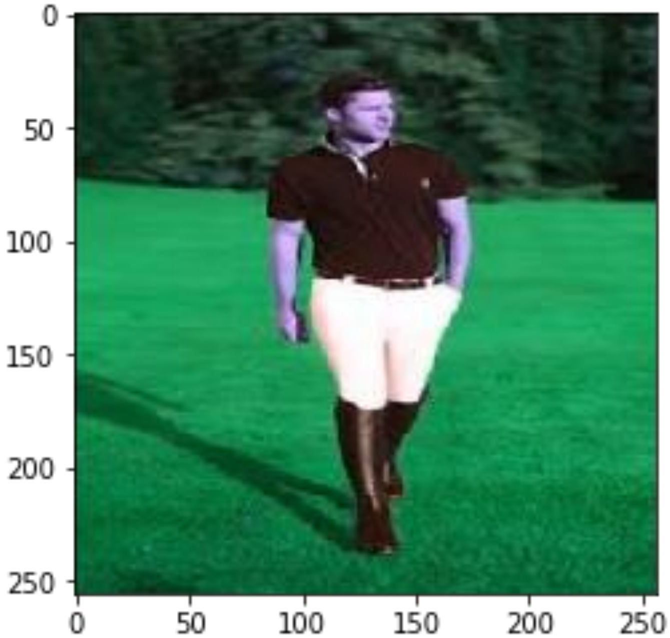
The Predicted Testing image is = cats verify below



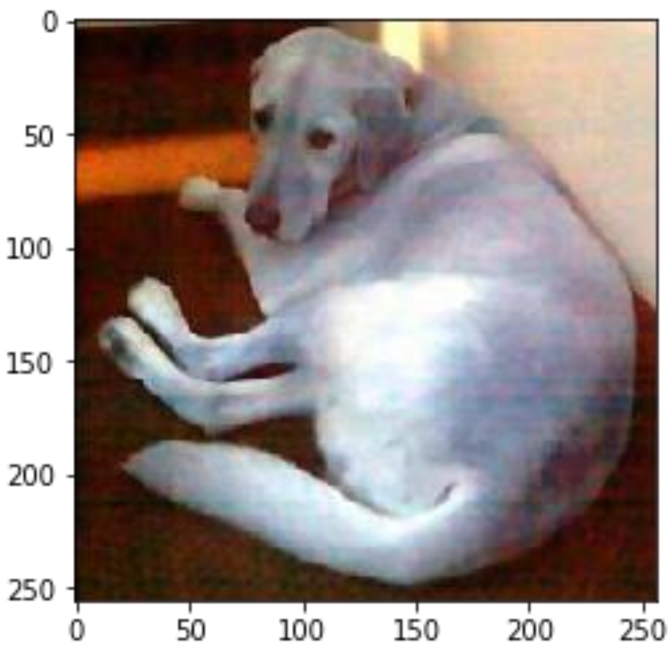
The Predicted Testing image is = humans verify below



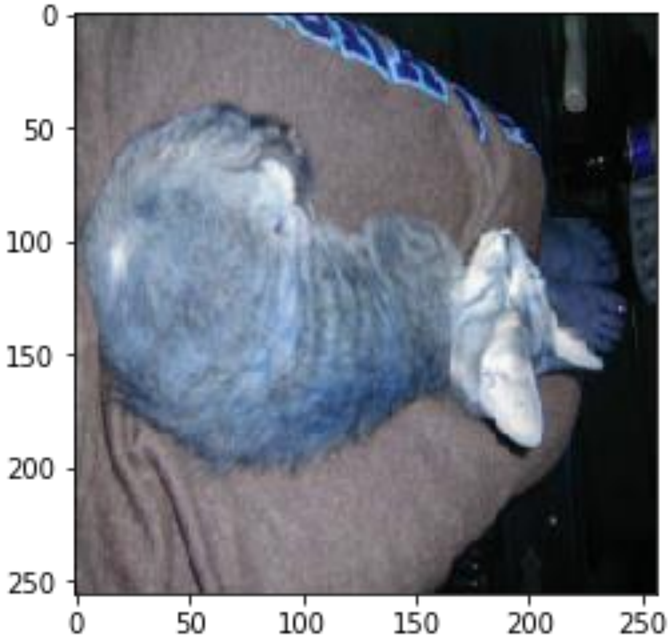
The Predicted Testing image is = humans verify below



The Predicted Testing image is = cats verify below



The Predicted Testing image is = dogs verify below



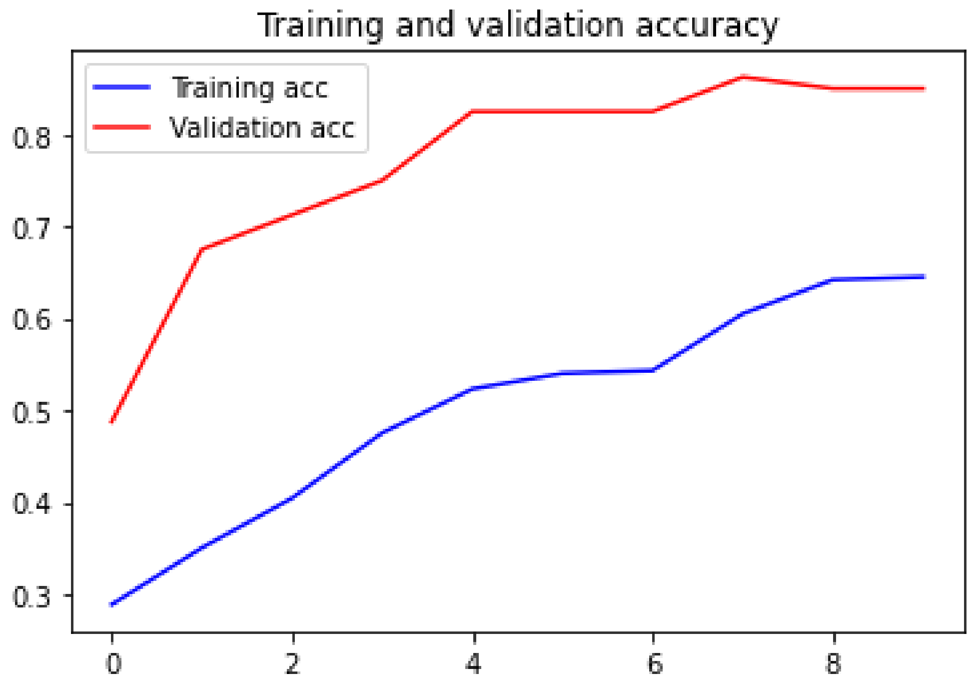

```
Execution Time = 9184 [s]
```

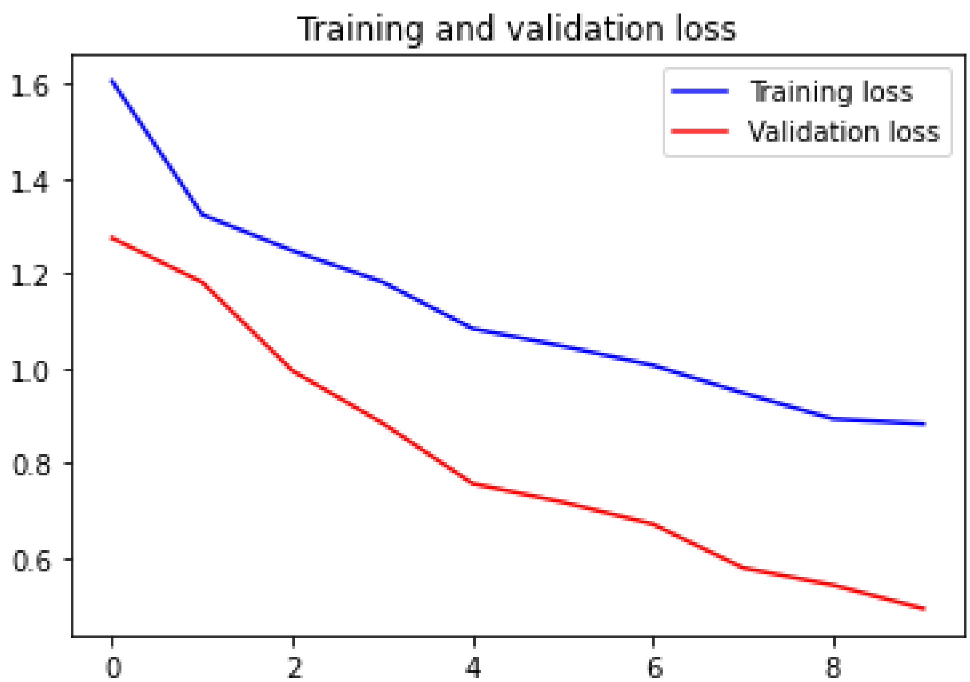
Part II VGG16 Transfer Learning

Training Results:

```
###################################################################
Load Datasets

Data Directory List 1- >  /content/drive/My Drive/Colab
Notebooks/Program9/cats_dogs_horse_humans/train
Data Directory List 2- >  /content/drive/My Drive/Colab
Notebooks/Program9/cats_dogs_horse_humans/validation
Data Directory List 3- >  /content/drive/My Drive/Colab
Notebooks/Program9/cats_dogs_horse_humans/test


###################################################################
Create Model


<tensorflow.python.keras.engine.input_layer.InputLayer object at 0x7fcbe3e975d0> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe3ed07d0> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe3ed0c90> False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7fcbe0c6af10> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c73e90> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c78e50> False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7fcbe0c84b50> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c8aed0> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c91ad0> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c84110> False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7fcbe0c9af50> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c23c90> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c28fd0> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c95250> False
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7fcbe0c2df90> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c35090> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c2ba10> False
<tensorflow.python.keras.layers.convolutional.Conv2D object at 0x7fcbe0c3dcd0> True
<tensorflow.python.keras.layers.pooling.MaxPooling2D object at 0x7fcbe0c42c50> True
```

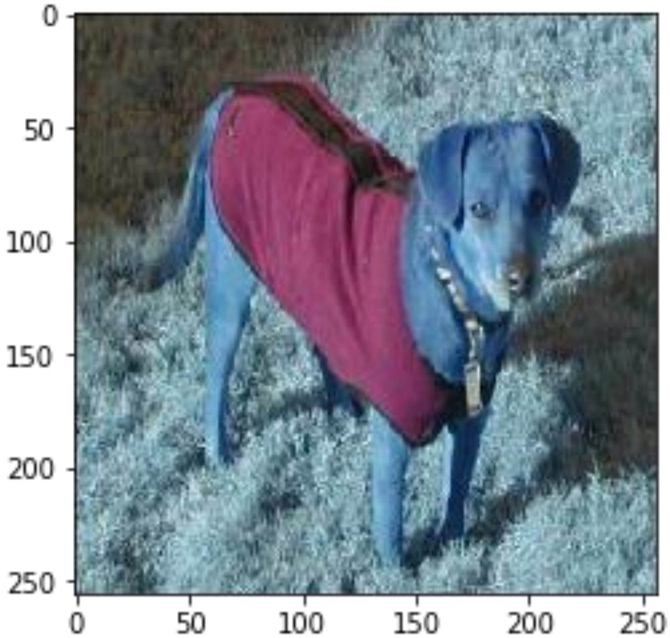
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 64)                2097216
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260
=================================================================
Total params: 16,816,324
Trainable params: 4,461,444
Non-trainable params: 12,354,880
_________________________________________________________________
Found 648 images belonging to 4 classes.
Found 80 images belonging to 4 classes.

###################################################################
Train Model

Epoch 1/10
12/12 [==============================] - 511s 39s/step - loss: 1.8536 - acc: 0.2674 - val_loss:
1.2744 - val_acc: 0.4875
Epoch 2/10
12/12 [==============================] - 494s 38s/step - loss: 1.3461 - acc: 0.3246 - val_loss:
1.1812 - val_acc: 0.6750
Epoch 3/10
12/12 [==============================] - 494s 38s/step - loss: 1.2634 - acc: 0.3992 - val_loss:
0.9949 - val_acc: 0.7125
Epoch 4/10
12/12 [==============================] - 493s 38s/step - loss: 1.2111 - acc: 0.4574 - val_loss:
0.8844 - val_acc: 0.7500
Epoch 5/10
12/12 [==============================] - 494s 38s/step - loss: 1.0913 - acc: 0.5208 - val_loss:
0.7569 - val_acc: 0.8250
Epoch 6/10
12/12 [==============================] - 496s 39s/step - loss: 1.0597 - acc: 0.5498 - val_loss:
0.7179 - val_acc: 0.8250
Epoch 7/10
12/12 [==============================] - 493s 38s/step - loss: 1.0075 - acc: 0.5294 - val_loss:
0.6715 - val_acc: 0.8250
Epoch 8/10
12/12 [==============================] - 492s 38s/step - loss: 0.9363 - acc: 0.6063 - val_loss:
0.5792 - val_acc: 0.8625
Epoch 9/10
12/12 [==============================] - 493s 38s/step - loss: 0.8926 - acc: 0.6428 - val_loss:
0.5430 - val_acc: 0.8500
Epoch 10/10
12/12 [==============================] - 494s 38s/step - loss: 0.8721 - acc: 0.6500 - val_loss:
0.4934 - val_acc: 0.8500
```



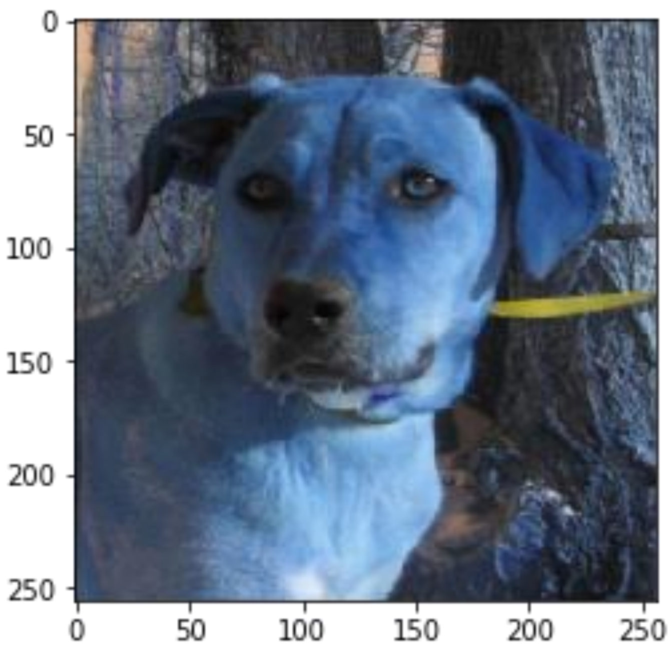


Testing Results:

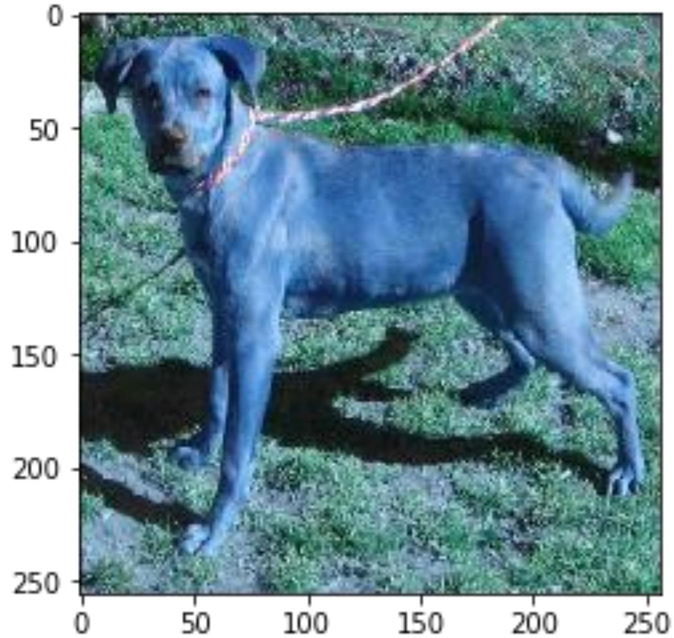
```
################################################################
Test model with testing data

3/3 [==============================] - 51s 15s/step - loss: 0.5020 - acc: 0.9000
Testing accuracy - > 89.99999761581421

The Predicted Testing image is = dogs verify below
```



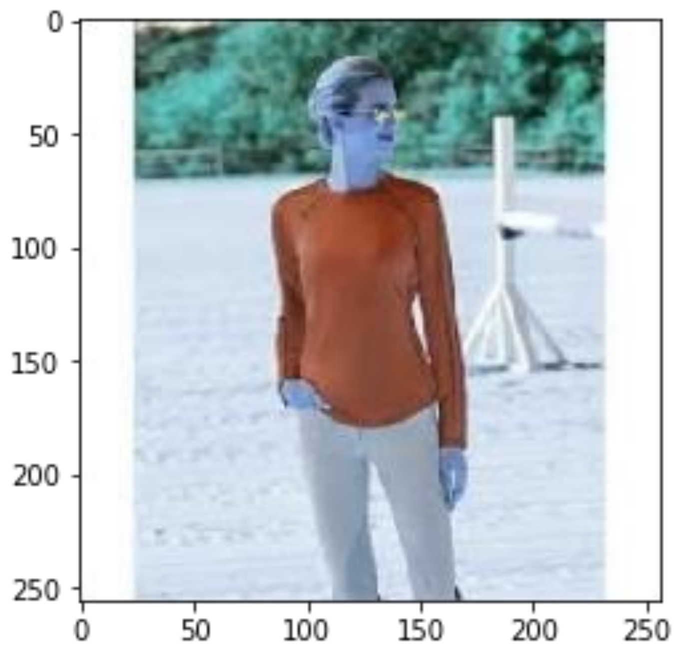

```
The Predicted Testing image is = dogs verify below
```



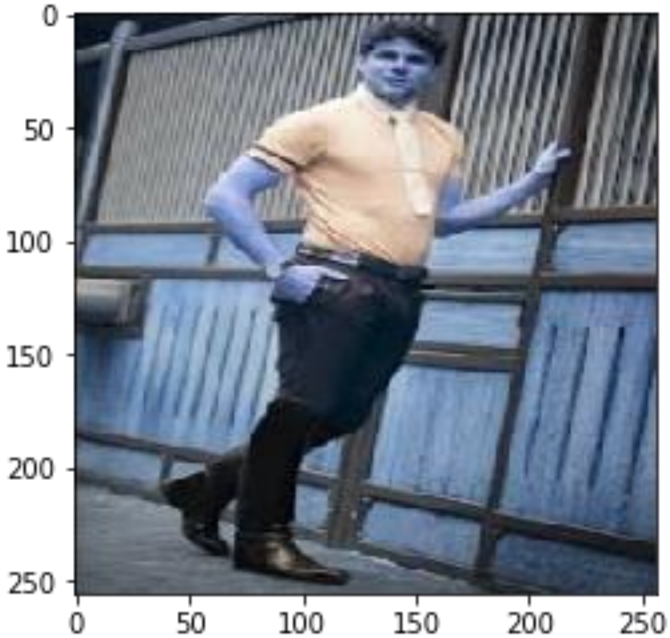

```
The Predicted Testing image is = horses verify below
```



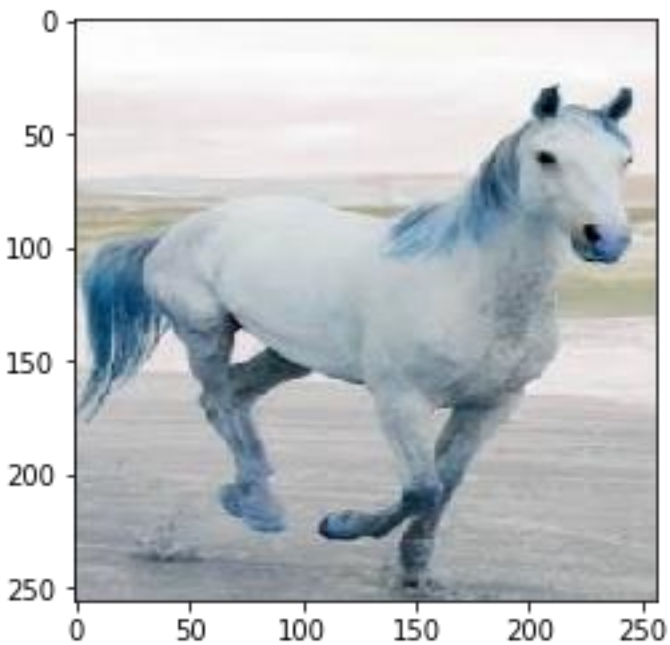
The Predicted Testing image is = humans verify below



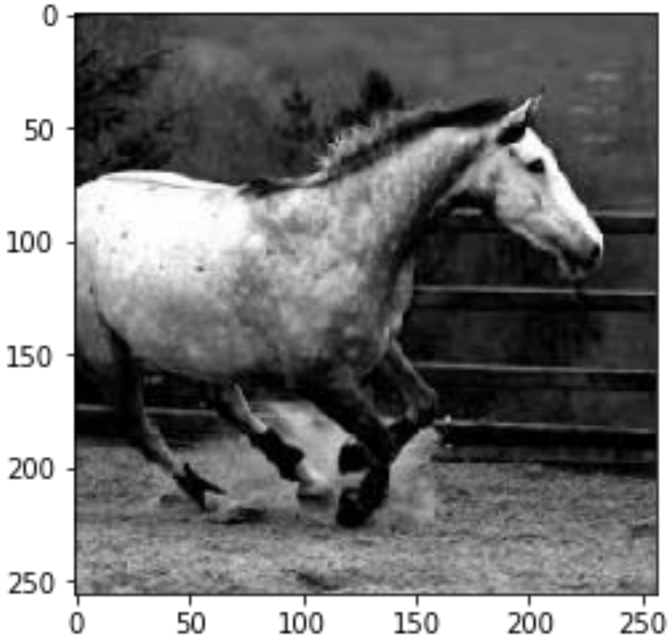
The Predicted Testing image is = humans verify below



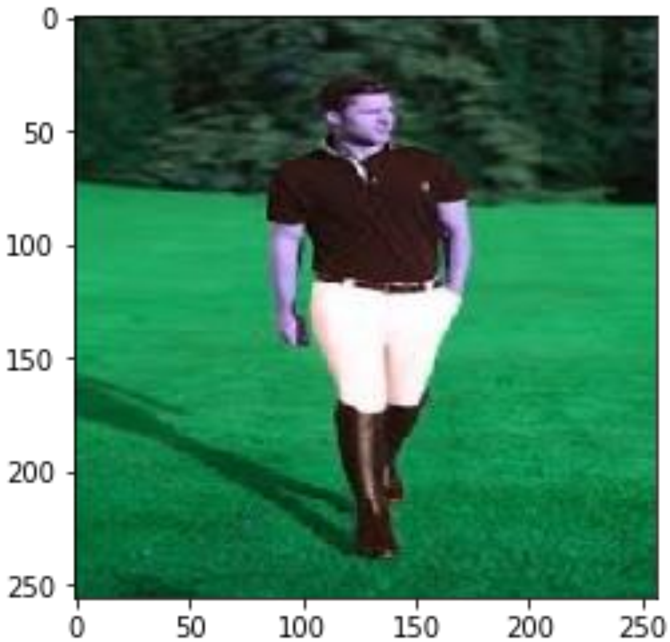
The Predicted Testing image is = horses verify below



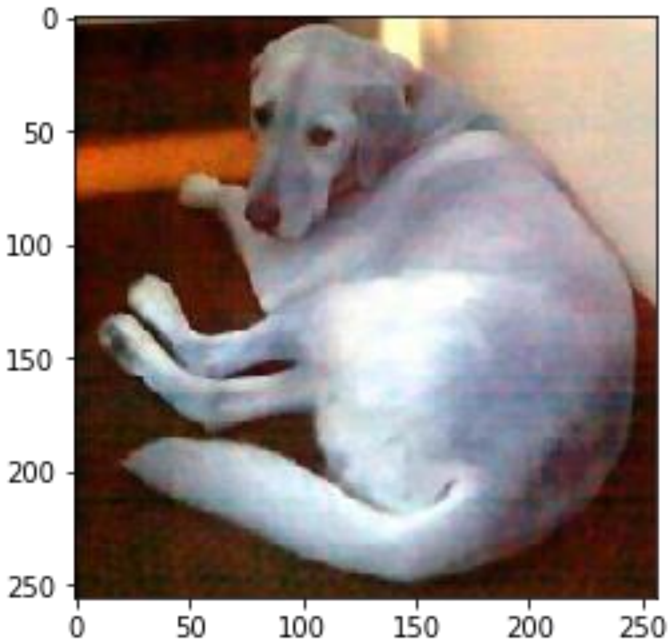
The Predicted Testing image is = horses verify below



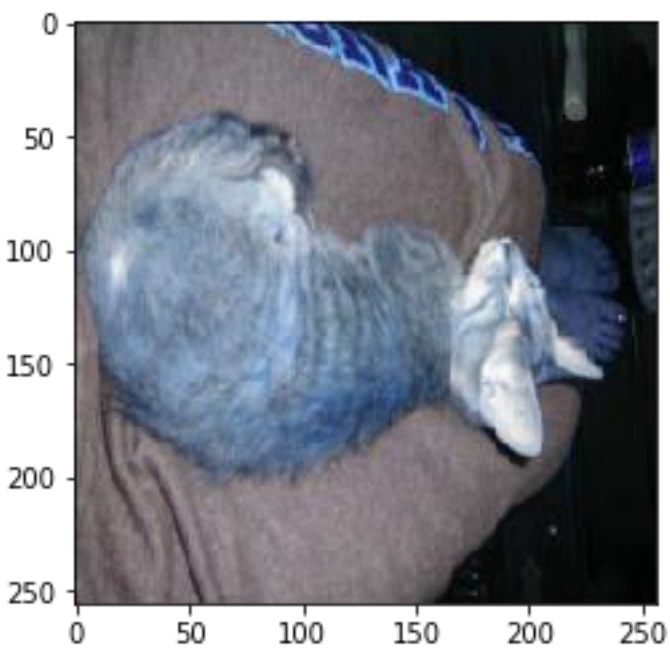
The Predicted Testing image is = humans verify below



The Predicted Testing image is = dogs verify below




```
The Predicted Testing image is = dogs verify below
```




```
Execution Time = 5120 [s]
```

Input/Output Processing Explanation:

Step 1: Import data
Step 2: Define number of classes
Step 3:

For data set in directory
    For images in image list
        Read image
        Change color space
        Resize
        Append label
    End
End
Step 4: Output unique labels
Step 5: Shuffle dataset
Step 6: Divide data into train & test
Step 7: Normalize data
Step 8: Reshape data to fit model
Step 9: Add convolutionary, max pooling, and dropout layers
Step 10: Compile Model
Step 11: Fit model on training Data
Step 12: Evaluate model on test data
Step 13: Test accuracy
Step 14: Print images vs predicted

Code:

https://github.com/jnn-dev/ML_Transfer_Learning

Conclusion:

This programing exercise utilized transfer learning on a vgg16 pre-trained network to adapt the network for the purpose of classifying dogs, cats, humans, and horses. The transfer learning technique resulted in a significant improvement to the validation and testing accuracy. Additionally, the computational time needed to train the network was significantly reduced. When the training dataset is small, it is advantageous to use transfer learning since this network can leverage all of the low-level details it has already learned.