Open in Colab

Image Classification Human vs Horse

Problem Definition - Main Objective

Company XYZ would like to build a Deep Learning model to classify horse vs human images.

With pre-trained models like InceptionV3 or VGG16, the model can be trained faster with Transfer Learning.

In [ ]:
```
# import required libraries
import os
import zipfile
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras import layers, optimizers, losses, metrics, Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import RMSprop
```

Data Set collection

In [ ]:
```
# Get the Horse or Human dataset
!wget --no-check-certificate https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip -O /tmp

# Get the Horse or Human Validation dataset
!wget --no-check-certificate https://storage.googleapis.com/laurencemoroney-blog.appspot.com/validation-horse-or-human.
```

```
--2022-05-16 04:54:18--  https://storage.googleapis.com/laurencemoroney-blog.appspot.com/horse-or-human.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 142.251.12.128, 142.250.4.128, 172.217.194.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|142.251.12.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 149574867 (143M) [application/zip]
Saving to: ‘/tmp/horse-or-human.zip’

/tmp/horse-or-human 100%[===================>] 142.65M  88.6MB/s    in 1.6s

2022-05-16 04:54:20 (88.6 MB/s) - ‘/tmp/horse-or-human.zip’ saved [149574867/149574867]

--2022-05-16 04:54:20--  https://storage.googleapis.com/laurencemoroney-blog.appspot.com/validation-horse-or-human.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.4.128, 172.217.194.128, 74.125.200.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.4.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11480187 (11M) [application/zip]
Saving to: ‘/tmp/validation-horse-or-human.zip’

/tmp/validation-hor 100%[===================>]  10.95M  --.-KB/s    in 0.1s

2022-05-16 04:54:21 (84.8 MB/s) - ‘/tmp/validation-horse-or-human.zip’ saved [11480187/11480187]
```

In [ ]:

```
# all the required train and validation files are available in a zip file. Unzip the files for training and validation
local_zip = '//tmp/horse-or-human.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp/training')
zip_ref.close()

local_zip = '//tmp/validation-horse-or-human.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp/validation')
zip_ref.close()
```

In [ ]:

```
# Lets assign dedicated folder for training and validation
train_horses_dir = '/tmp/training/horses'
train_humans_dir = '/tmp/training/humans'
validation_horses_dir = '/tmp/validation/horses'
validation_humans_dir = '/tmp/validation/humans'

train_horses_fnames = os.listdir(train_horses_dir)
train_humans_fnames = os.listdir(train_humans_dir)
validation_horses_fnames = os.listdir(validation_horses_dir)
validation_humans_fnames = os.listdir(validation_humans_dir)
```

```
print('Total Horse Training files',len(train_horses_fnames))
print('Total Horse Training files',len(train_humans_fnames))
print('Total Horse Training files',len(validation_horses_fnames))
print('Total Horse Training files',len(validation_humans_fnames))
```

```
Total Horse Training files 500
Total Horse Training files 527
Total Horse Training files 128
Total Horse Training files 128
```

In [ ]:

```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

pic_index = 2

next_horse = [os.path.join(train_horses_dir, fname)
                for fname in train_horses_fnames[pic_index-2:pic_index]]
next_human = [os.path.join(train_humans_dir, fname)
                for fname in train_humans_fnames[pic_index-2:pic_index]]


for i, img_path in enumerate(next_horse+next_human):
  #print(img_path)
  img = mpimg.imread(img_path)
  plt.imshow(img)
  plt.axis('Off')
  plt.show()
```

In [ ]:
```
# Define our example directories and files
```

```
train_dir = '/tmp/training'
validation_dir = '/tmp/validation'

# Add our data-augmentation parameters to ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255.,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True
                                   )

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(# Your Code Here
                                  rescale = 1./255.,
                                  )

# Flow training images in batches of 20 using train_datagen generator
train_generator = train_datagen.flow_from_directory(# Your Code Here
                                                    train_dir,
                                                    batch_size = 20,
                                                    class_mode = 'binary',
                                                    target_size = (150, 150)
                                                    )

# Flow validation images in batches of 20 using test_datagen generator
validation_generator =  test_datagen.flow_from_directory( # Your Code Here
                                                         validation_dir,
                                                         batch_size = 20,
                                                         class_mode = 'binary',
                                                         target_size = (150, 150)
                                                         )
```

```
Found 1027 images belonging to 2 classes.
Found 256 images belonging to 2 classes.
```

Dataset Description

We have got 1027 images for training and 256 images for validation.

Below is the highlevel break down for each these classes (Horse vs Human)

- Total Horse Training files 500

- Total Horse Training files 527
- Total Horse Training files 128
- Total Horse Training files 128

Model building

Reusable methods

In [ ]:
```
# method to display loss vs accuracy graph

import matplotlib.pyplot as plt

def plot_graph(history):
  acc = history.history['accuracy']
  val_acc = history.history['val_accuracy']
  loss = history.history['loss']
  val_loss = history.history['val_loss']

  epochs = range(len(acc))

  plt.plot(epochs, acc, 'r', label='Training accuracy')
  plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
  plt.title('Training and validation accuracy')
  plt.legend(loc=0)
  plt.figure()

  plt.show()
```

In [ ]:
```
# Define a Callback class that stops training once accuracy reaches 99.9% We dont want to overfit our model.
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('accuracy')>0.999):
      print("\nReached 99.9% accuracy so cancelling training!")
      self.model.stop_training = True

# Callback for early stopping. If training loss didnt improve after 10 epoch stop training
es_callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=10)
```

Basic CNN for Image Classification

We will use a basic CNN first to explore the model building before we use Transfer learning

In [ ]:
```
# Basic CNN Model building
cnn_model = tf.keras.models.Sequential([
# YOUR CODE HERE
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    # Only 1 output neuron. It will contain a value from 0-1 where 0 for 1 class ('horse') and 1 for the other ('human'
    tf.keras.layers.Dense(1, activation='sigmoid')
])

cnn_model.compile(optimizer=RMSprop(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
cnn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 16)      448

 max_pooling2d (MaxPooling2D  (None, 74, 74, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 flatten (Flatten)           (None, 18496)             0

 dense (Dense)               (None, 512)               9470464

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
BATCH_SIZE = 20
TRAIN_STEPS_PER_EPOCH = len(train_horses_fnames)//BATCH_SIZE
VAL_STEPS_PER_EPOCH = len(validation_horses_fnames)//BATCH_SIZE
callbacks = myCallback()
cnn_history = cnn_model.fit(
                    train_generator,
                    validation_data = validation_generator,
                    steps_per_epoch = TRAIN_STEPS_PER_EPOCH * 2,
                    epochs = 100,
                    validation_steps = VAL_STEPS_PER_EPOCH,
                    verbose = 1,
                    callbacks=[callbacks, es_callback]
                    )
```

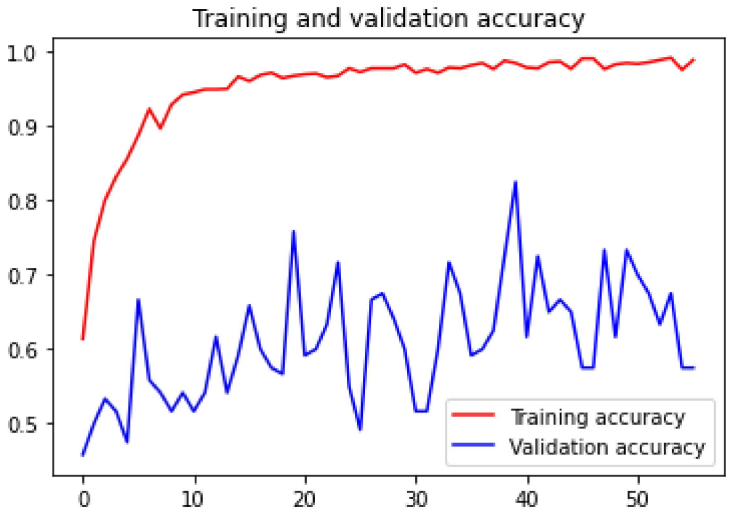
```
Epoch 1/100
50/50 [==============================] - 17s 229ms/step - loss: 0.8494 - accuracy: 0.6140 - val_loss: 1.5633 - val_accu
racy: 0.4583
Epoch 2/100
50/50 [==============================] - 10s 203ms/step - loss: 0.5895 - accuracy: 0.7457 - val_loss: 1.7347 - val_accu
racy: 0.5000
Epoch 3/100
50/50 [==============================] - 10s 201ms/step - loss: 0.4393 - accuracy: 0.8004 - val_loss: 2.5188 - val_accu
racy: 0.5333
Epoch 4/100
50/50 [==============================] - 10s 206ms/step - loss: 0.3536 - accuracy: 0.8318 - val_loss: 4.0049 - val_accu
racy: 0.5167
Epoch 5/100
50/50 [==============================] - 10s 204ms/step - loss: 0.3385 - accuracy: 0.8561 - val_loss: 2.3794 - val_accu
racy: 0.4750
Epoch 6/100
50/50 [==============================] - 11s 214ms/step - loss: 0.2583 - accuracy: 0.8875 - val_loss: 1.7181 - val_accu
racy: 0.6667
Epoch 7/100
50/50 [==============================] - 10s 202ms/step - loss: 0.1962 - accuracy: 0.9230 - val_loss: 3.5932 - val_accu
racy: 0.5583
Epoch 8/100
50/50 [==============================] - 10s 201ms/step - loss: 0.2175 - accuracy: 0.8967 - val_loss: 4.0138 - val_accu
racy: 0.5417
Epoch 9/100
50/50 [==============================] - 10s 202ms/step - loss: 0.1817 - accuracy: 0.9291 - val_loss: 4.7219 - val_accu
racy: 0.5167
Epoch 10/100
50/50 [==============================] - 10s 206ms/step - loss: 0.1644 - accuracy: 0.9422 - val_loss: 4.5330 - val_accu
racy: 0.5417
Epoch 11/100
50/50 [==============================] - 10s 203ms/step - loss: 0.1525 - accuracy: 0.9453 - val_loss: 5.9270 - val_accu
racy: 0.5167
Epoch 12/100
50/50 [==============================] - 10s 201ms/step - loss: 0.1368 - accuracy: 0.9493 - val_loss: 4.7034 - val_accu
racy: 0.5417
Epoch 13/100
50/50 [==============================] - 10s 202ms/step - loss: 0.1421 - accuracy: 0.9493 - val_loss: 2.2010 - val_accu
racy: 0.6167
Epoch 14/100
50/50 [==============================] - 10s 203ms/step - loss: 0.1299 - accuracy: 0.9500 - val_loss: 3.9058 - val_accu
racy: 0.5417
Epoch 15/100
50/50 [==============================] - 10s 205ms/step - loss: 0.0930 - accuracy: 0.9666 - val_loss: 3.9982 - val_accu
racy: 0.5917
```

```
Epoch 16/100
50/50 [==============================] - 10s 205ms/step - loss: 0.1167 - accuracy: 0.9605 - val_loss: 3.0971 - val_accu
racy: 0.6583
Epoch 17/100
50/50 [==============================] - 10s 206ms/step - loss: 0.1011 - accuracy: 0.9686 - val_loss: 3.3670 - val_accu
racy: 0.6000
Epoch 18/100
50/50 [==============================] - 10s 208ms/step - loss: 0.0749 - accuracy: 0.9716 - val_loss: 5.9293 - val_accu
racy: 0.5750
Epoch 19/100
50/50 [==============================] - 10s 202ms/step - loss: 0.1327 - accuracy: 0.9645 - val_loss: 6.8743 - val_accu
racy: 0.5667
Epoch 20/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0837 - accuracy: 0.9676 - val_loss: 1.4438 - val_accu
racy: 0.7583
Epoch 21/100
50/50 [==============================] - 10s 200ms/step - loss: 0.1106 - accuracy: 0.9696 - val_loss: 3.4531 - val_accu
racy: 0.5917
Epoch 22/100
50/50 [==============================] - 10s 200ms/step - loss: 0.0825 - accuracy: 0.9706 - val_loss: 4.8307 - val_accu
racy: 0.6000
Epoch 23/100
50/50 [==============================] - 10s 205ms/step - loss: 0.0971 - accuracy: 0.9656 - val_loss: 5.0928 - val_accu
racy: 0.6333
Epoch 24/100
50/50 [==============================] - 10s 201ms/step - loss: 0.0871 - accuracy: 0.9676 - val_loss: 3.3813 - val_accu
racy: 0.7167
Epoch 25/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0658 - accuracy: 0.9777 - val_loss: 10.8243 - val_acc
uracy: 0.5500
Epoch 26/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0978 - accuracy: 0.9726 - val_loss: 7.9092 - val_accu
racy: 0.4917
Epoch 27/100
50/50 [==============================] - 10s 201ms/step - loss: 0.0732 - accuracy: 0.9777 - val_loss: 4.7966 - val_accu
racy: 0.6667
Epoch 28/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0579 - accuracy: 0.9777 - val_loss: 4.0475 - val_accu
racy: 0.6750
Epoch 29/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0668 - accuracy: 0.9777 - val_loss: 7.2993 - val_accu
racy: 0.6417
Epoch 30/100
50/50 [==============================] - 10s 206ms/step - loss: 0.0541 - accuracy: 0.9828 - val_loss: 7.0440 - val_accu
racy: 0.6000
```

```
Epoch 31/100
50/50 [==============================] - 10s 202ms/step - loss: 0.1283 - accuracy: 0.9716 - val_loss: 9.5202 - val_accu
racy: 0.5167
Epoch 32/100
50/50 [==============================] - 10s 203ms/step - loss: 0.1104 - accuracy: 0.9767 - val_loss: 9.4203 - val_accu
racy: 0.5167
Epoch 33/100
50/50 [==============================] - 10s 205ms/step - loss: 0.0842 - accuracy: 0.9716 - val_loss: 5.7366 - val_accu
racy: 0.6000
Epoch 34/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0878 - accuracy: 0.9787 - val_loss: 2.8607 - val_accu
racy: 0.7167
Epoch 35/100
50/50 [==============================] - 10s 200ms/step - loss: 0.0567 - accuracy: 0.9777 - val_loss: 3.9766 - val_accu
racy: 0.6750
Epoch 36/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0668 - accuracy: 0.9818 - val_loss: 7.6885 - val_accu
racy: 0.5917
Epoch 37/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0531 - accuracy: 0.9848 - val_loss: 6.7034 - val_accu
racy: 0.6000
Epoch 38/100
50/50 [==============================] - 10s 201ms/step - loss: 0.0686 - accuracy: 0.9767 - val_loss: 6.1137 - val_accu
racy: 0.6250
Epoch 39/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0687 - accuracy: 0.9878 - val_loss: 3.1021 - val_accu
racy: 0.7250
Epoch 40/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0613 - accuracy: 0.9848 - val_loss: 1.6946 - val_accu
racy: 0.8250
Epoch 41/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0679 - accuracy: 0.9787 - val_loss: 5.6471 - val_accu
racy: 0.6167
Epoch 42/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0649 - accuracy: 0.9777 - val_loss: 2.5001 - val_accu
racy: 0.7250
Epoch 43/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0403 - accuracy: 0.9858 - val_loss: 6.8650 - val_accu
racy: 0.6500
Epoch 44/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0822 - accuracy: 0.9868 - val_loss: 3.8030 - val_accu
racy: 0.6667
Epoch 45/100
50/50 [==============================] - 10s 204ms/step - loss: 0.0621 - accuracy: 0.9770 - val_loss: 5.0307 - val_accu
racy: 0.6500
```

```
Epoch 46/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0266 - accuracy: 0.9909 - val_loss: 8.9077 - val_accu
racy: 0.5750
Epoch 47/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0367 - accuracy: 0.9909 - val_loss: 11.1510 - val_acc
uracy: 0.5750
Epoch 48/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0703 - accuracy: 0.9767 - val_loss: 2.5511 - val_accu
racy: 0.7333
Epoch 49/100
50/50 [==============================] - 10s 207ms/step - loss: 0.0833 - accuracy: 0.9830 - val_loss: 4.1969 - val_accu
racy: 0.6167
Epoch 50/100
50/50 [==============================] - 10s 201ms/step - loss: 0.0686 - accuracy: 0.9848 - val_loss: 1.8435 - val_accu
racy: 0.7333
Epoch 51/100
50/50 [==============================] - 10s 202ms/step - loss: 0.0303 - accuracy: 0.9838 - val_loss: 5.2668 - val_accu
racy: 0.7000
Epoch 52/100
50/50 [==============================] - 10s 201ms/step - loss: 0.0655 - accuracy: 0.9858 - val_loss: 5.1701 - val_accu
racy: 0.6750
Epoch 53/100
50/50 [==============================] - 10s 205ms/step - loss: 0.0496 - accuracy: 0.9889 - val_loss: 5.8049 - val_accu
racy: 0.6333
Epoch 54/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0309 - accuracy: 0.9919 - val_loss: 9.5020 - val_accu
racy: 0.6750
Epoch 55/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0823 - accuracy: 0.9757 - val_loss: 9.1325 - val_accu
racy: 0.5750
Epoch 56/100
50/50 [==============================] - 10s 203ms/step - loss: 0.0353 - accuracy: 0.9889 - val_loss: 9.3248 - val_accu
racy: 0.5750
```

In [ ]:
```
plot_graph(cnn_history)
```




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

The training stopped after 54th epoch. There is no improvement in the training loss.

Lets try to repeat the experiment with Transfer Learning

InceptionV3

In [ ]:
```
# Import the inception model
from tensorflow.keras.applications.inception_v3 import InceptionV3

pre_trained_inception_model = InceptionV3(
    input_shape=(150, 150, 3),
    include_top=False,
    weights='imagenet'
)
```

Freeze top layers and build the model

In [ ]:
```
# we will use Functional API to merge both pretrained model and our custom model
def build_model(pretrained_model):
  # Make all the layers in the pre-trained model non-trainable
  for layer in pretrained_model.layers:
    layer.trainable = False
  last_layer = pretrained_model.get_layer('block5_pool') # block5_pool mixed7
```

```
  print('last layer output shape: ', last_layer.output_shape)
  last_output = last_layer.output
  # Flatten the output layer to 1 dimension
  x = layers.Flatten()(last_output)
  # Add a fully connected layer with 1,024 hidden units and ReLU activation
  x = layers.Dense(1024, activation='relu')(x)
  # Add a dropout rate of 0.2
  x = layers.Dropout(0.2)(x)
  # Add a final sigmoid layer for classification
  x = layers.Dense  (1, activation='sigmoid')(x)

  model = Model( pretrained_model.input, x)

  model.compile(optimizer=RMSprop(learning_rate=0.001),
                loss = 'binary_crossentropy',
                metrics = ['accuracy'])
  return model
```

Model Training

In [ ]:

```
# Run this and see how many epochs it should take before the callback
# fires, and stops training at 99.9% accuracy
# (It should take less than 100 epochs)
BATCH_SIZE = 20
TRAIN_STEPS_PER_EPOCH = len(train_horses_fnames)//BATCH_SIZE
VAL_STEPS_PER_EPOCH = len(validation_horses_fnames)//BATCH_SIZE
callbacks = myCallback()
inception_model = build_model(pre_trained_inception_model)
inception_history = inception_model.fit(
                    train_generator,
                    validation_data = validation_generator,
                    steps_per_epoch = TRAIN_STEPS_PER_EPOCH * 2,
                    epochs = 100,
                    validation_steps = VAL_STEPS_PER_EPOCH,
                    verbose = 1,
                    callbacks=[callbacks, es_callback]
                    )
```

```
last layer output shape:  (None, 7, 7, 768)
Epoch 1/100
50/50 [==============================] - 15s 233ms/step - loss: 1.2128 - accuracy: 0.9230 - val_loss: 0.0134 - val_accu
racy: 0.9917
Epoch 2/100
50/50 [==============================] - 11s 214ms/step - loss: 0.1825 - accuracy: 0.9737 - val_loss: 6.9436e-08 - val_
accuracy: 1.0000
Epoch 3/100
50/50 [==============================] - 11s 214ms/step - loss: 0.1069 - accuracy: 0.9838 - val_loss: 0.3799 - val_accu
racy: 0.9667
Epoch 4/100
50/50 [==============================] - 11s 216ms/step - loss: 0.1054 - accuracy: 0.9889 - val_loss: 0.0062 - val_accu
racy: 0.9917
Epoch 5/100
50/50 [==============================] - 11s 217ms/step - loss: 0.1261 - accuracy: 0.9919 - val_loss: 0.0268 - val_accu
racy: 0.9917
Epoch 6/100
50/50 [==============================] - 11s 214ms/step - loss: 0.0918 - accuracy: 0.9868 - val_loss: 0.2998 - val_accu
racy: 0.9667
Epoch 7/100
50/50 [==============================] - 11s 214ms/step - loss: 0.0745 - accuracy: 0.9919 - val_loss: 0.0967 - val_accu
racy: 0.9917
Epoch 8/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0283 - accuracy: 0.9960 - val_loss: 0.1188 - val_accu
racy: 0.9750
Epoch 9/100
50/50 [==============================] - 11s 216ms/step - loss: 0.1070 - accuracy: 0.9909 - val_loss: 0.0642 - val_accu
racy: 0.9917
Epoch 10/100
50/50 [==============================] - 11s 214ms/step - loss: 0.0296 - accuracy: 0.9949 - val_loss: 0.1392 - val_accu
racy: 0.9833
Epoch 11/100
50/50 [==============================] - 11s 214ms/step - loss: 0.0174 - accuracy: 0.9980 - val_loss: 0.0131 - val_accu
racy: 0.9917
Epoch 12/100
50/50 [==============================] - 11s 217ms/step - loss: 0.0492 - accuracy: 0.9880 - val_loss: 0.0604 - val_accu
racy: 0.9917
Epoch 13/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0933 - accuracy: 0.9920 - val_loss: 4.1139e-07 - val_
accuracy: 1.0000
Epoch 14/100
50/50 [==============================] - 11s 215ms/step - loss: 0.0929 - accuracy: 0.9909 - val_loss: 0.1997 - val_accu
racy: 0.9833
Epoch 15/100
50/50 [==============================] - 11s 215ms/step - loss: 0.0418 - accuracy: 0.9939 - val_loss: 0.0018 - val_accu
```

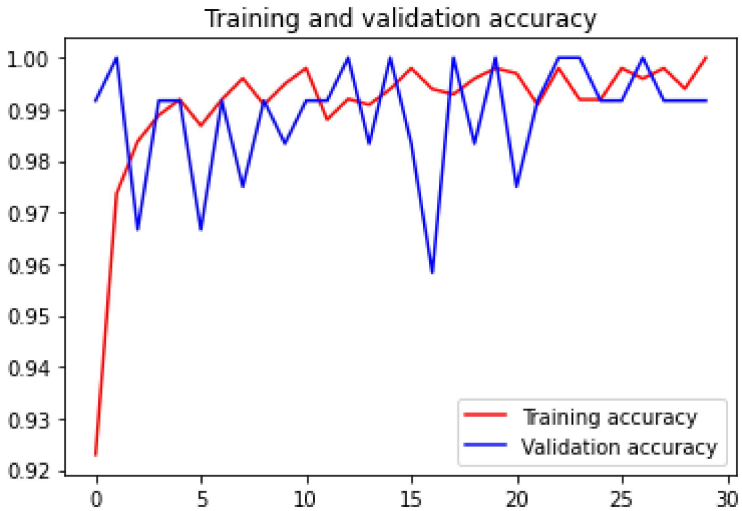
```
racy: 1.0000
Epoch 16/100
50/50 [==============================] - 11s 218ms/step - loss: 0.0442 - accuracy: 0.9980 - val_loss: 0.2171 - val_accu
racy: 0.9833
Epoch 17/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0982 - accuracy: 0.9939 - val_loss: 0.4318 - val_accu
racy: 0.9583
Epoch 18/100
50/50 [==============================] - 11s 214ms/step - loss: 0.1003 - accuracy: 0.9929 - val_loss: 4.1362e-08 - val_
accuracy: 1.0000
Epoch 19/100
50/50 [==============================] - 11s 215ms/step - loss: 0.0180 - accuracy: 0.9959 - val_loss: 0.0294 - val_accu
racy: 0.9833
Epoch 20/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0164 - accuracy: 0.9980 - val_loss: 8.8795e-11 - val_
accuracy: 1.0000
Epoch 21/100
50/50 [==============================] - 11s 219ms/step - loss: 0.0228 - accuracy: 0.9970 - val_loss: 0.2519 - val_accu
racy: 0.9750
Epoch 22/100
50/50 [==============================] - 11s 215ms/step - loss: 0.0483 - accuracy: 0.9909 - val_loss: 0.0338 - val_accu
racy: 0.9917
Epoch 23/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0038 - accuracy: 0.9980 - val_loss: 2.5101e-08 - val_
accuracy: 1.0000
Epoch 24/100
50/50 [==============================] - 11s 217ms/step - loss: 0.0368 - accuracy: 0.9919 - val_loss: 0.0024 - val_accu
racy: 1.0000
Epoch 25/100
50/50 [==============================] - 11s 215ms/step - loss: 0.1004 - accuracy: 0.9919 - val_loss: 0.0538 - val_accu
racy: 0.9917
Epoch 26/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0187 - accuracy: 0.9980 - val_loss: 0.0784 - val_accu
racy: 0.9917
Epoch 27/100
50/50 [==============================] - 11s 217ms/step - loss: 0.0640 - accuracy: 0.9959 - val_loss: 2.3130e-16 - val_
accuracy: 1.0000
Epoch 28/100
50/50 [==============================] - 11s 216ms/step - loss: 0.0336 - accuracy: 0.9980 - val_loss: 0.1603 - val_accu
racy: 0.9917
Epoch 29/100
50/50 [==============================] - 11s 219ms/step - loss: 0.0548 - accuracy: 0.9940 - val_loss: 0.0492 - val_accu
racy: 0.9917
Epoch 30/100
50/50 [==============================] - ETA: 0s - loss: 1.6942e-05 - accuracy: 1.0000
```

```
Reached 99.9% accuracy so cancelling training!
50/50 [==============================] - 11s 218ms/step - loss: 1.6942e-05 - accuracy: 1.0000 - val_loss: 0.2509 - val_
accuracy: 0.9917
```

We reched 99.9% accuracy with in 30th Epoch itself

Model Accuracy

In [ ]:
```
plot_graph(inception_history)
```




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

VGG16

In [ ]:
```
# Import the inception model
from tensorflow.keras.applications.vgg16 import VGG16

pre_trained_VGG16_model = VGG16(
    input_shape=(150, 150, 3),
    include_top=False,
    weights='imagenet' # we will use the same imagenet weights
)
```

In [ ]:
```
vgg16_model = build_model(pre_trained_VGG16_model)
vgg16_model_history = vgg16_model.fit(
                    train_generator,
```

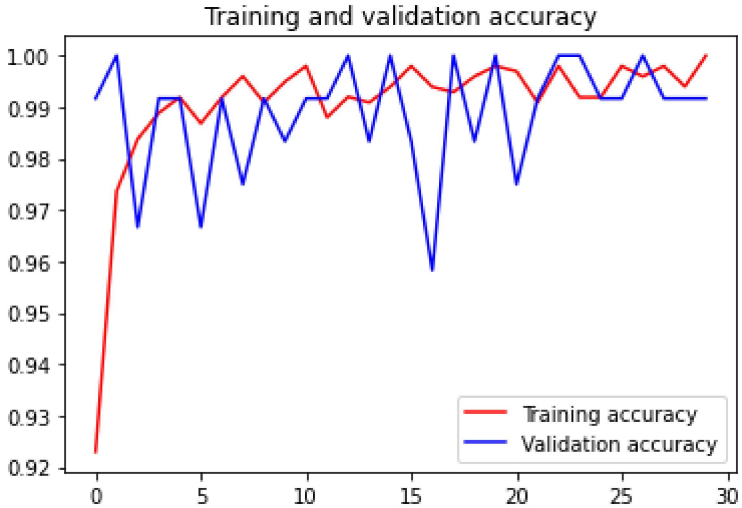
```
                    validation_data = validation_generator,
                    steps_per_epoch = TRAIN_STEPS_PER_EPOCH * 2,
                    epochs = 100,
                    validation_steps = VAL_STEPS_PER_EPOCH,
                    verbose = 1,
                    callbacks=[callbacks, es_callback]
                    )
plot_graph(inception_history)
```

```
last layer output shape:  (None, 4, 4, 512)
Epoch 1/100
50/50 [==============================] - 14s 240ms/step - loss: 1.2749 - accuracy: 0.7822 - val_loss: 0.5556 - val_accu
racy: 0.8667
Epoch 2/100
50/50 [==============================] - 11s 225ms/step - loss: 0.3983 - accuracy: 0.8734 - val_loss: 0.0634 - val_accu
racy: 0.9750
Epoch 3/100
50/50 [==============================] - 12s 237ms/step - loss: 0.2314 - accuracy: 0.9129 - val_loss: 0.0550 - val_accu
racy: 0.9667
Epoch 4/100
50/50 [==============================] - 11s 225ms/step - loss: 0.2287 - accuracy: 0.9210 - val_loss: 1.0257 - val_accu
racy: 0.8167
Epoch 5/100
50/50 [==============================] - 11s 227ms/step - loss: 0.1488 - accuracy: 0.9440 - val_loss: 0.3148 - val_accu
racy: 0.9500
Epoch 6/100
50/50 [==============================] - 11s 225ms/step - loss: 0.1544 - accuracy: 0.9605 - val_loss: 0.1608 - val_accu
racy: 0.9583
Epoch 7/100
50/50 [==============================] - 11s 222ms/step - loss: 0.1938 - accuracy: 0.9473 - val_loss: 0.4215 - val_accu
racy: 0.9417
Epoch 8/100
50/50 [==============================] - 11s 225ms/step - loss: 0.1385 - accuracy: 0.9585 - val_loss: 0.7486 - val_accu
racy: 0.8667
Epoch 9/100
50/50 [==============================] - 11s 227ms/step - loss: 0.0752 - accuracy: 0.9666 - val_loss: 0.3480 - val_accu
racy: 0.9500
Epoch 10/100
50/50 [==============================] - 11s 224ms/step - loss: 0.1371 - accuracy: 0.9544 - val_loss: 0.8658 - val_accu
racy: 0.8667
Epoch 11/100
50/50 [==============================] - 11s 227ms/step - loss: 0.0834 - accuracy: 0.9696 - val_loss: 0.2708 - val_accu
racy: 0.9583
Epoch 12/100
50/50 [==============================] - 11s 227ms/step - loss: 0.0744 - accuracy: 0.9797 - val_loss: 0.3169 - val_accu
racy: 0.9500
Epoch 13/100
50/50 [==============================] - 11s 225ms/step - loss: 0.1261 - accuracy: 0.9676 - val_loss: 0.2844 - val_accu
racy: 0.9583
Epoch 14/100
50/50 [==============================] - 11s 223ms/step - loss: 0.1004 - accuracy: 0.9696 - val_loss: 0.6732 - val_accu
racy: 0.9167
Epoch 15/100
50/50 [==============================] - 11s 225ms/step - loss: 0.0855 - accuracy: 0.9777 - val_loss: 0.5091 - val_accu
```

```
racy: 0.9500
Epoch 16/100
50/50 [==============================] - 11s 224ms/step - loss: 0.0528 - accuracy: 0.9797 - val_loss: 0.7018 - val_accu
racy: 0.9333
Epoch 17/100
50/50 [==============================] - 11s 222ms/step - loss: 0.0752 - accuracy: 0.9757 - val_loss: 0.7217 - val_accu
racy: 0.9167
Epoch 18/100
50/50 [==============================] - 11s 219ms/step - loss: 0.1200 - accuracy: 0.9686 - val_loss: 0.1383 - val_accu
racy: 0.9750
Epoch 19/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0572 - accuracy: 0.9818 - val_loss: 0.6150 - val_accu
racy: 0.9083
Epoch 20/100
50/50 [==============================] - 11s 220ms/step - loss: 0.0783 - accuracy: 0.9797 - val_loss: 0.6357 - val_accu
racy: 0.9250
Epoch 21/100
50/50 [==============================] - 11s 221ms/step - loss: 0.0634 - accuracy: 0.9828 - val_loss: 0.0510 - val_accu
racy: 0.9833
Epoch 22/100
50/50 [==============================] - 11s 221ms/step - loss: 0.0907 - accuracy: 0.9747 - val_loss: 0.8432 - val_accu
racy: 0.9167
Epoch 23/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0810 - accuracy: 0.9757 - val_loss: 0.4934 - val_accu
racy: 0.9417
Epoch 24/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0500 - accuracy: 0.9828 - val_loss: 1.0256 - val_accu
racy: 0.8667
Epoch 25/100
50/50 [==============================] - 11s 222ms/step - loss: 0.0767 - accuracy: 0.9807 - val_loss: 0.8515 - val_accu
racy: 0.8833
Epoch 26/100
50/50 [==============================] - 11s 222ms/step - loss: 0.1015 - accuracy: 0.9757 - val_loss: 0.6270 - val_accu
racy: 0.9500
Epoch 27/100
50/50 [==============================] - 11s 224ms/step - loss: 0.0870 - accuracy: 0.9726 - val_loss: 0.8797 - val_accu
racy: 0.9333
Epoch 28/100
50/50 [==============================] - 11s 220ms/step - loss: 0.0312 - accuracy: 0.9878 - val_loss: 0.7856 - val_accu
racy: 0.9250
Epoch 29/100
50/50 [==============================] - 12s 243ms/step - loss: 0.0606 - accuracy: 0.9868 - val_loss: 0.6505 - val_accu
racy: 0.9333
Epoch 30/100
50/50 [==============================] - 12s 230ms/step - loss: 0.0631 - accuracy: 0.9828 - val_loss: 0.3990 - val_accu
```

```
racy: 0.9500
Epoch 31/100
50/50 [==============================] - 11s 222ms/step - loss: 0.0491 - accuracy: 0.9858 - val_loss: 0.2587 - val_accu
racy: 0.9583
Epoch 32/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0438 - accuracy: 0.9848 - val_loss: 0.2788 - val_accu
racy: 0.9750
Epoch 33/100
50/50 [==============================] - 11s 220ms/step - loss: 0.0367 - accuracy: 0.9899 - val_loss: 0.0419 - val_accu
racy: 0.9750
Epoch 34/100
50/50 [==============================] - 11s 219ms/step - loss: 0.0566 - accuracy: 0.9807 - val_loss: 0.4132 - val_accu
racy: 0.9583
Epoch 35/100
50/50 [==============================] - 11s 224ms/step - loss: 0.0245 - accuracy: 0.9899 - val_loss: 1.5951 - val_accu
racy: 0.8667
Epoch 36/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0527 - accuracy: 0.9899 - val_loss: 0.7659 - val_accu
racy: 0.9500
Epoch 37/100
50/50 [==============================] - 11s 227ms/step - loss: 0.0384 - accuracy: 0.9919 - val_loss: 0.5664 - val_accu
racy: 0.9417
Epoch 38/100
50/50 [==============================] - 11s 224ms/step - loss: 0.0508 - accuracy: 0.9848 - val_loss: 0.5401 - val_accu
racy: 0.9583
Epoch 39/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0394 - accuracy: 0.9889 - val_loss: 0.4226 - val_accu
racy: 0.9667
Epoch 40/100
50/50 [==============================] - 11s 222ms/step - loss: 0.0535 - accuracy: 0.9878 - val_loss: 0.1885 - val_accu
racy: 0.9750
Epoch 41/100
50/50 [==============================] - 11s 225ms/step - loss: 0.0426 - accuracy: 0.9868 - val_loss: 0.6575 - val_accu
racy: 0.9583
Epoch 42/100
50/50 [==============================] - 11s 225ms/step - loss: 0.0712 - accuracy: 0.9818 - val_loss: 0.9648 - val_accu
racy: 0.9167
Epoch 43/100
50/50 [==============================] - 11s 222ms/step - loss: 0.0347 - accuracy: 0.9909 - val_loss: 0.5998 - val_accu
racy: 0.9500
Epoch 44/100
50/50 [==============================] - 11s 224ms/step - loss: 0.0208 - accuracy: 0.9949 - val_loss: 0.4313 - val_accu
racy: 0.9667
Epoch 45/100
50/50 [==============================] - 11s 230ms/step - loss: 0.0502 - accuracy: 0.9878 - val_loss: 2.1595 - val_accu
```

```
racy: 0.8500
Epoch 46/100
50/50 [==============================] - 11s 225ms/step - loss: 0.0999 - accuracy: 0.9828 - val_loss: 0.4565 - val_accu
racy: 0.9583
Epoch 47/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0305 - accuracy: 0.9929 - val_loss: 0.9215 - val_accu
racy: 0.9500
Epoch 48/100
50/50 [==============================] - 11s 225ms/step - loss: 0.0601 - accuracy: 0.9848 - val_loss: 0.2996 - val_accu
racy: 0.9750
Epoch 49/100
50/50 [==============================] - 11s 221ms/step - loss: 0.0545 - accuracy: 0.9889 - val_loss: 0.2718 - val_accu
racy: 0.9583
Epoch 50/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0508 - accuracy: 0.9868 - val_loss: 0.7381 - val_accu
racy: 0.9583
Epoch 51/100
50/50 [==============================] - 11s 224ms/step - loss: 0.0771 - accuracy: 0.9807 - val_loss: 0.7050 - val_accu
racy: 0.9583
Epoch 52/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0410 - accuracy: 0.9899 - val_loss: 0.2595 - val_accu
racy: 0.9667
Epoch 53/100
50/50 [==============================] - 11s 222ms/step - loss: 0.0599 - accuracy: 0.9848 - val_loss: 0.9326 - val_accu
racy: 0.9417
Epoch 54/100
50/50 [==============================] - 11s 223ms/step - loss: 0.0374 - accuracy: 0.9929 - val_loss: 0.7116 - val_accu
racy: 0.9500
```




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

Model stopped training after 54th iteration once there is no improvement in loss

Final Model Selection - Key findings

- With 3 different model we exerimented, Inception baed model outperformed other 2 models.
- VGG16 seems very light and faster to train compared to InceptionV3 model
- InceptionV3 converged fatster compared to VGG16
- Our plain custom model didnt converge

Final recomendation is to use Inception Model

Possible Flaws and Potential improvement

- We have used very less training document and Image augmentation to train the network. This can cause overfitting at somepoint.
- To mitigate it, we can increse sample size.
- Apart from Inception model, we can also try YOLO or RestNet to improve the model convergence