In [1]:

```
from PIL import Image
import tensorflow as tf
import tensorflow.keras.preprocessing as image
from sklearn.model_selection import train_test_split

from keras.layers import concatenate
from keras.layers import Input
import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D,GlobalAveragePooling2D, Flatten, Dense, Dropout, BatchNormaliz
from tensorflow.keras import layers
from tensorflow.keras.applications.densenet import DenseNet169
from matplotlib import pyplot as plt
import pandas as pd
import keras
inp_size=512
```

In [2]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Input, Flatten, GlobalAveragePooling2D, Concatenate, BatchNormalization, Dense, Dro


def create_model():
    input_layer = Input(shape = (inp_size, inp_size, 3))
    base_1 = keras.applications.EfficientNetB0(weights = 'imagenet', include_top = False, input_shape = (inp_size, inp_
    base_2 = keras.applications.ResNet152(weights = 'imagenet', include_top = False, input_shape = (inp_size, inp_size,
    base_3 = keras.applications.VGG19(weights = 'imagenet', include_top = False, input_shape = (inp_size, inp_size, 3))

    for layer in base_1.layers:
        layer.trainable =  False
    for layer in base_2.layers:
        layer.trainable = False
    for layer in base_3.layers:
        layer.trainable = False

    model_1 = base_1(input_layer)
    model_1 = GlobalAveragePooling2D()(model_1)
    output_1 = Flatten()(model_1)

    model_2 = base_2(input_layer)
    model_2 = GlobalAveragePooling2D()(model_2)
    output_2 = Flatten()(model_2)

    model_3 = base_3(input_layer)
    model_3 = GlobalAveragePooling2D()(model_3)
    output_3 = Flatten()(model_3)


    merged = tf.keras.layers.Concatenate()([output_1, output_2,output_3])
    x = BatchNormalization()(merged)
    x = Dense(128,activation = 'relu')(x)
    x = Dense(2,activation='softmax')(x)
    stacked_model = tf.keras.models.Model(inputs = input_layer, outputs = x)
    return stacked_model

# Generate sample images
# num_samples = 1000
# images = np.random.rand(num_samples, inp_size, inp_size, 3)

# # Generate sample labels
# labels = np.random.randint(0, 2, size=(num_samples, 2))

# # Define the model
# model = create_model()

# # Compile the model
# model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# # Fit the model to the sample data
# model.fit(images, labels, epochs=10, batch_size=32)
```

In [3]:

```
model = create_model()
```

In [29]:

```
model.summary()
```

```
Model: "model_7"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 input_19 (InputLayer)          [(None, 512, 512, 3  0           []                               
                                )]                                                                
                                                                                                  
 efficientnetb0 (Functional)    (None, 16, 16, 1280  4049571     ['input_19[0][0]']               
                                )                                                                 
                                                                                                  
 resnet152 (Functional)         (None, 16, 16, 2048  58370944    ['input_19[0][0]']               
                                )                                                                 
                                                                                                  
 vgg19 (Functional)             (None, 16, 16, 512)  20024384    ['input_19[0][0]']               
                                                                                                  
 global_average_pooling2d_11 (G  (None, 1280)        0           ['efficientnetb0[0][0]']         
 lobalAveragePooling2D)                                                                           
                                                                                                  
 global_average_pooling2d_12 (G  (None, 2048)        0           ['resnet152[0][0]']              
 lobalAveragePooling2D)                                                                           
                                                                                                  
 global_average_pooling2d_13 (G  (None, 512)         0           ['vgg19[0][0]']                  
 lobalAveragePooling2D)                                                                           
                                                                                                  
 flatten_11 (Flatten)           (None, 1280)         0           ['global_average_pooling2d_11[0][
                                                                 0]']                             
                                                                                                  
 flatten_12 (Flatten)           (None, 2048)         0           ['global_average_pooling2d_12[0][
                                                                 0]']                             
                                                                                                  
 flatten_13 (Flatten)           (None, 512)          0           ['global_average_pooling2d_13[0][
                                                                 0]']                             
                                                                                                  
 concatenate_4 (Concatenate)    (None, 3840)         0           ['flatten_11[0][0]',             
                                                                  'flatten_12[0][0]',             
                                                                  'flatten_13[0][0]']             
                                                                                                  
 batch_normalization_7 (BatchNo  (None, 3840)        15360       ['concatenate_4[0][0]']          
 rmalization)                                                                                     
                                                                                                  
 dense_15 (Dense)               (None, 128)          491648      ['batch_normalization_7[0][0]']  
                                                                                                  
 dense_16 (Dense)               (None, 2)            258         ['dense_15[0][0]']               
                                                                                                  
==================================================================================================
Total params: 82,952,165
Trainable params: 499,586
Non-trainable params: 82,452,579
__________________________________________________________________________________________________
```

In [16]:

```
model.save('final_res151_vgg19_stacked.h5')
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` wi
ll be empty until you train or evaluate the model.
```

In [24]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Input, Flatten, GlobalAveragePooling2D, Concatenate, BatchNormalization, Dense, Dro


def create_model():
    input_layer = Input(shape = (inp_size, inp_size, 3))
    base_1 = keras.applications.EfficientNetB0(weights = 'imagenet', include_top = False, input_shape = (inp_size, inp_
    for layer in base_1.layers:
        layer.trainable =  False
    model_1 = base_1(input_layer)
    model_1 = GlobalAveragePooling2D()(model_1)
    output_1 = Flatten()(model_1)
    x = BatchNormalization()(output_1)
    x = Dense(128,activation = 'relu')(x)
    x = Dense(2,activation='softmax')(x)
    stacked_model = tf.keras.models.Model(inputs = input_layer, outputs = x)
    return stacked_model

# # Generate sample images
# num_samples = 1000
# images = np.random.rand(num_samples, inp_size, inp_size, 3)

# # Generate sample labels
# labels = np.random.randint(0, 2, size=(num_samples, 2))

# # Define the model
# model = create_model()

# # Compile the model
# model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# # Fit the model to the sample data
# model.fit(images, labels, epochs=10, batch_size=32)
```

In [25]:

```
m = create_model()
m.summary()
```

```
Model: "model_6"

_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_17 (InputLayer)       [(None, 512, 512, 3)]     0         
                                                                 
 efficientnetb0 (Functional)  (None, 16, 16, 1280)     4049571   
                                                                 
 global_average_pooling2d_10  (None, 1280)             0         
  (GlobalAveragePooling2D)                                       
                                                                 
 flatten_10 (Flatten)        (None, 1280)              0         
                                                                 
 batch_normalization_6 (Batc  (None, 1280)             5120      
 hNormalization)                                                 
                                                                 
 dense_13 (Dense)            (None, 128)               163968    
                                                                 
 dense_14 (Dense)            (None, 2)                 258       
                                                                 
=================================================================
Total params: 4,218,917
Trainable params: 166,786
Non-trainable params: 4,052,131
_________________________________________________________________
```

In [26]:

```
m.save("final_effb0.h5")
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` wi
ll be empty until you train or evaluate the model.
```

In [ ]:

```

```