```
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.python.keras import layers, models
from tensorflow.python.keras.applications import vgg16
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

#the path to the dataset
data_dir = 'chest_xray/train'
data_dir_val = 'chest_xray/val'
data_dir_test = 'chest_xray/test'


category_names = sorted(os.listdir('chest_xray/train'))
nb_categories = len(category_names)

#defining that the transfer learning are using 3 layers.
img_height, img_width = 224, 224
conv_base = vgg16.VGG16(weights='imagenet', include_top=False,
pooling='max', input_shape = (img_width, img_height, 3))

model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(nb_categories, activation='softmax'))
model.summary()

# Number of images to load at each iteration
batch_size = 32

# only rescaling
train_datagen = ImageDataGenerator(
  rescale=1. / 255
)
test_datagen = ImageDataGenerator(
  rescale=1. / 255
)

# these are generators for train/test data that will read pictures in the
subfolders

print('Total number of images for "training":')
train = train_datagen.flow_from_directory(
  data_dir,
  target_size=(img_height, img_width),
  batch_size=batch_size,
  class_mode="categorical")

print('Total number of images for "validation":')
val = test_datagen.flow_from_directory(
  data_dir_val,
  target_size=(img_height, img_width),
  batch_size=batch_size,
  class_mode="categorical",
  shuffle=False)

print('Total number of images for "testing":')
test = test_datagen.flow_from_directory(
```

```
  data_dir_test,
  target_size=(img_height, img_width),
  batch_size=batch_size,
  class_mode="categorical",
  shuffle=False)

#setting the learing rate and what optimizer to use
base_learning_rate = 0.0003
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])

#Put how many runs it shall take.
history = model.fit(train,
                    epochs=10,
                    shuffle=True,
                    validation_data=val,
                    )

model.save('my_model_transfer.h5')

#printing graphs for loss and accurazy
def print_graph(item):
  plt.plot(history.history[item])
  plt.plot(history.history['val_' + item])
  plt.legend(['training', 'validation'])
  plt.title('Training and validation ' + item)
  plt.xlabel('epoch')
  plt.show()


print_graph('accuracy')
print_graph('loss')
```