```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt
```

```
from google.colab import files
files.upload()
! rm -rf ~/.kaggle/
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 62 bytes, last modified: 4/3/2020 - 100% done

```
Saving kaggle.json to kaggle (1).json
```

```
! pip install -q kaggle
```

```
!pip uninstall -y kaggle
!pip install --upgrade pip
!pip install kaggle==1.5.6
!kaggle -v
```

```
Uninstalling kaggle-1.5.6:
  Successfully uninstalled kaggle-1.5.6
Collecting pip
  Downloading https://files.pythonhosted.org/packages/54/0c/d01aa759fdc501a58f431eb594a1
     |████████████████████████████████| 1.4MB 2.8MB/s
Installing collected packages: pip
  Found existing installation: pip 19.3.1
    Uninstalling pip-19.3.1:
      Successfully uninstalled pip-19.3.1
Successfully installed pip-20.0.2
Collecting kaggle==1.5.6
  Downloading kaggle-1.5.6.tar.gz (58 kB)
     |████████████████████████████████| 58 kB 1.8 MB/s
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from k
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kagg
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-pack
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) ... done
  Created wheel for kaggle: filename=kaggle-1.5.6-py3-none-any.whl size=72859 sha256=b2e
  Stored in directory: /root/.cache/pip/wheels/01/3e/ff/77407ebac3ef71a79b9166a8382aecf8
Successfully built kaggle
Installing collected packages: kaggle
Successfully installed kaggle-1.5.6
Kaggle API 1.5.6
```

```
! kaggle competitions download -c dogs-vs-cats
```

```
Downloading dogs-vs-cats.zip to /content
100% 809M/812M [00:16<00:00, 72.3MB/s]
100% 812M/812M [00:16<00:00, 52.9MB/s]
```

```
! mkdir dogs-vs-cats
! unzip dogs-vs-cats -d dogs-vs-cats
```

```
Archive:  dogs-vs-cats.zip
  inflating: dogs-vs-cats/sampleSubmission.csv
  inflating: dogs-vs-cats/test1.zip
  inflating: dogs-vs-cats/train.zip
```

```
! mkdir train
! unzip dogs-vs-cats/train -d train
! mkdir test
! unzip dogs-vs-cats/test1 -d test
```

```
! mkdir train/train/dog/
```

```
! mkdir train/train/cat/
```

```
!mv train/train/cat.* train/train/cat
!mv train/train/dog.* train/train/dog
```

```
!mkdir validation/
!mkdir validation/cat/
!mkdir validation/dog/
!mv train/train/cat/cat.1????.jpg validation/cat/
!mv train/train/dog/dog.1????.jpg validation/dog/
```

```
train_dir = 'train/train/'
validation_dir = 'validation'
train_cats_dir = 'train/train/cat/'
train_dogs_dir = 'train/train/dog/'
validation_cats_dir = 'validation/cat/'
validation_dogs_dir = 'validation/dog/'

num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))

num_cats_val = len(os.listdir(validation_cats_dir))
num_dogs_val = len(os.listdir(validation_dogs_dir))

total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val
```

Задание 1. Загрузите данные. Разделите исходный набор данных на обучающую, валидацион

```
print('total training cat images:', num_cats_tr)
print('total training dog images:', num_dogs_tr)

print('total validation cat images:', num_cats_val)
print('total validation dog images:', num_dogs_val)
print("--")
print("Total training images:", total_train)
print("Total validation images:", total_val)
```

```
total training cat images: 10000
total training dog images: 10000
total validation cat images: 2500
total validation dog images: 2500
--
Total training images: 20000
Total validation images: 5000
```

```
batch_size = 128
epochs = 15
IMG_HEIGHT = 150
IMG_WIDTH = 150
```

```
IMG_WIDTH = 150
```

```
train_image_generator = ImageDataGenerator(rescale=1./255)
validation_image_generator = ImageDataGenerator(rescale=1./255)
```

```
train_data_gen = train_image_generator.flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(IMG_HEIGHT, IMG_
                                                           class_mode='binary')
```

```
Found 20000 images belonging to 2 classes.
```

```
val_data_gen = validation_image_generator.flow_from_directory(batch_size=batch_size,
                                                              directory=validation_dir,
                                                              target_size=(IMG_HEIGHT, I
                                                              class_mode='binary')
```

```
Found 5000 images belonging to 2 classes.
```

```
sample_training_images, _ = next(train_data_gen)
```

```
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

```
plotImages(sample_training_images[:5])
```

Задание 2. Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями.
получено?

```
model = Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 20736)             0
_________________________________________________________________
dense (Dense)                (None, 512)               10617344
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 10,641,441
Trainable params: 10,641,441
Non-trainable params: 0
_________________________________________________________________
```

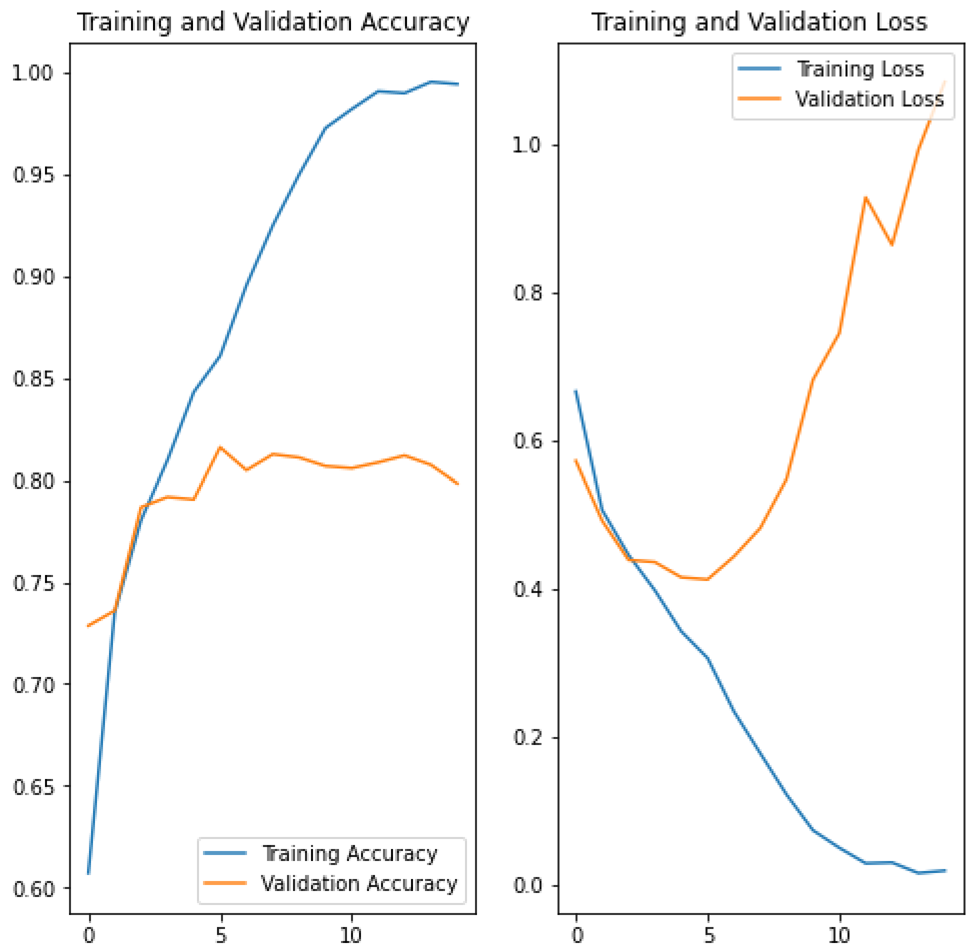
```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
```

```
    validation_steps=total_val // batch_size
)
```

```
WARNING:tensorflow:From <ipython-input-23-01c6f78f4d4f>:6: Model.fit_generator (from ten
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/15
156/156 [==============================] - 68s 433ms/step - loss: 0.6665 - accuracy: 0.6
Epoch 2/15
156/156 [==============================] - 65s 420ms/step - loss: 0.5068 - accuracy: 0.7
Epoch 3/15
156/156 [==============================] - 69s 440ms/step - loss: 0.4463 - accuracy: 0.7
Epoch 4/15
156/156 [==============================] - 66s 424ms/step - loss: 0.3983 - accuracy: 0.8
Epoch 5/15
156/156 [==============================] - 65s 419ms/step - loss: 0.3430 - accuracy: 0.8
Epoch 6/15
156/156 [==============================] - 65s 417ms/step - loss: 0.3068 - accuracy: 0.8
Epoch 7/15
156/156 [==============================] - 66s 423ms/step - loss: 0.2346 - accuracy: 0.8
Epoch 8/15
156/156 [==============================] - 66s 420ms/step - loss: 0.1781 - accuracy: 0.9
Epoch 9/15
156/156 [==============================] - 66s 421ms/step - loss: 0.1221 - accuracy: 0.9
Epoch 10/15
156/156 [==============================] - 66s 421ms/step - loss: 0.0740 - accuracy: 0.9
Epoch 11/15
156/156 [==============================] - 65s 416ms/step - loss: 0.0503 - accuracy: 0.9
Epoch 12/15
156/156 [==============================] - 65s 419ms/step - loss: 0.0293 - accuracy: 0.9
Epoch 13/15
156/156 [==============================] - 65s 414ms/step - loss: 0.0304 - accuracy: 0.9
Epoch 14/15
156/156 [==============================] - 64s 413ms/step - loss: 0.0161 - accuracy: 0.9
Epoch 15/15
156/156 [==============================] - 65s 415ms/step - loss: 0.0194 - accuracy: 0.9
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
```

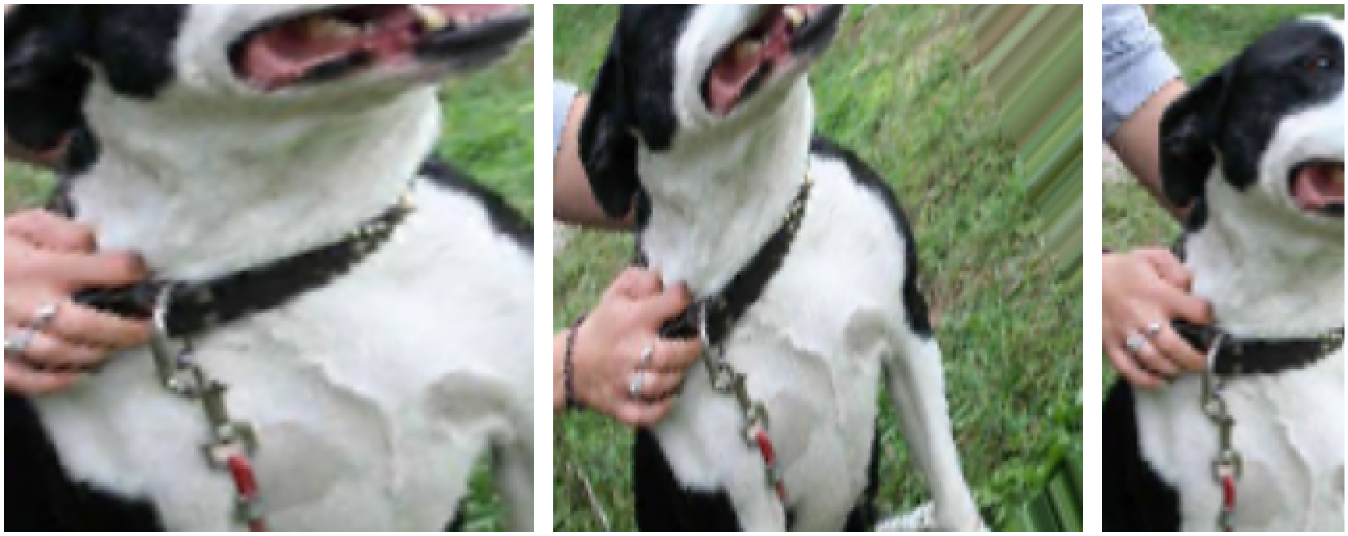
```
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Задание 3. Примените дополнение данных (data augmentation). Как это повлияло на качество

```
image_gen_train = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )
train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH)
                                                     class_mode='binary')
augmented_images = [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmented_images)
```

```
Found 20000 images belonging to 2 classes.
```



```python
image_gen_val = ImageDataGenerator(rescale=1./255)
```

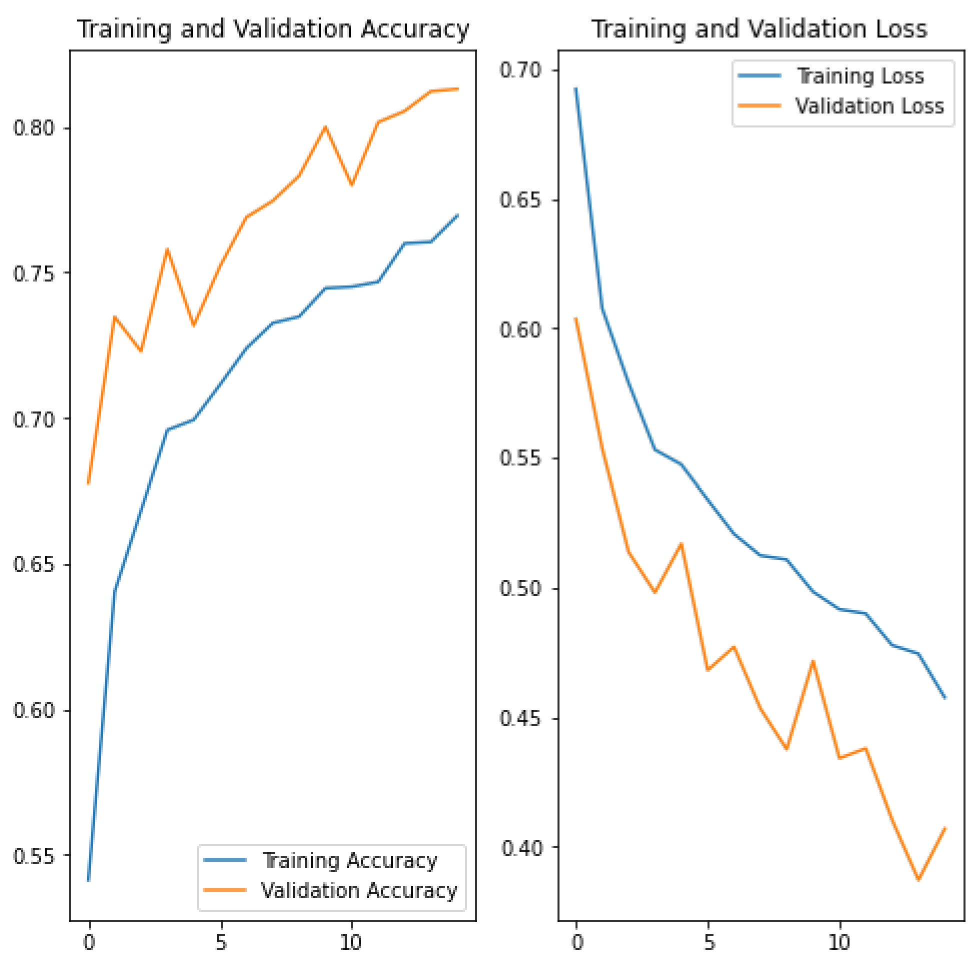
```python
val_data_gen = image_gen_val.flow_from_directory(batch_size=batch_size,
                                                 directory=validation_dir,
                                                 target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                 class_mode='binary')
```

```
Found 5000 images belonging to 2 classes.
```

```python
model = Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.summary()
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)

```

```
Model: "sequential_1"
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```




```
156/156 [==============================] - 140s 899ms/step - loss: 0.4576 - accuracy: 0
```

Качество улучшилось

Задание 4. Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, передаточное обучение. Как это повлияло на качество классификатора? Какой максимальны Kaggle? Почему?

```
model_new = Sequential([
    Conv2D(16, 3, padding='same', activation='relu',
           input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(),
    Dropout(0.2),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1)
])
```

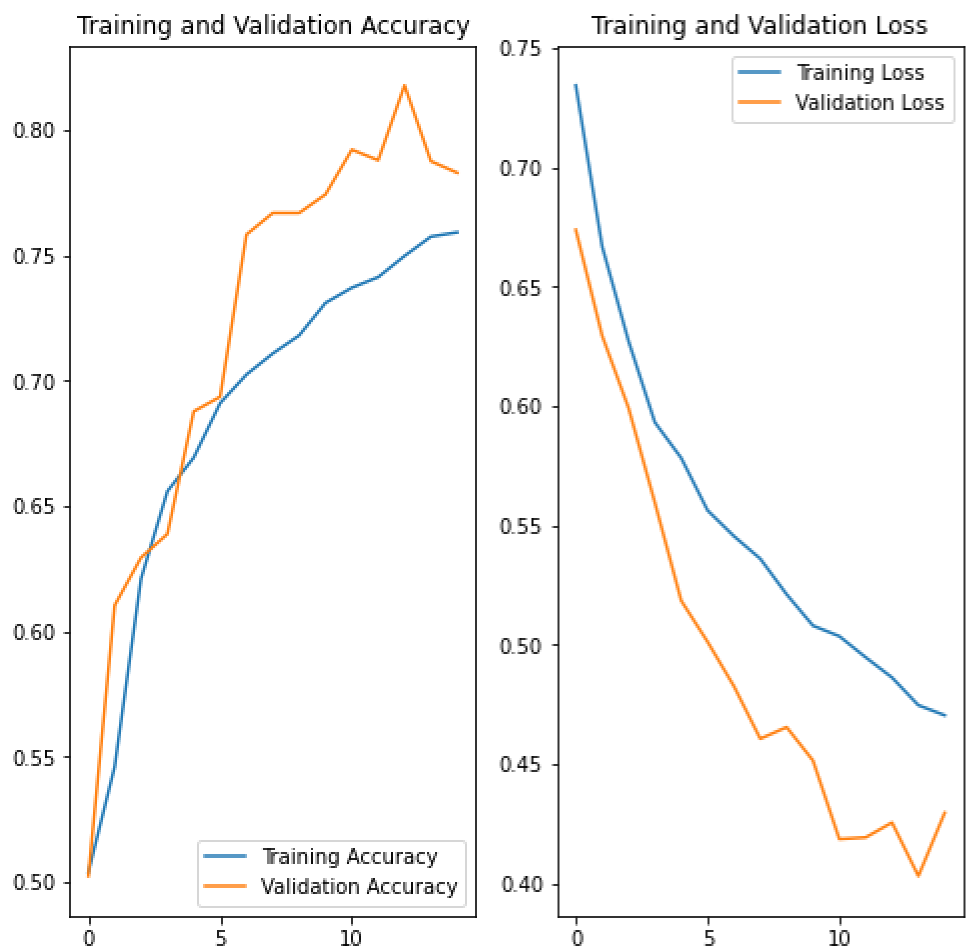
```
model_new.compile(optimizer='adam',
                  loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                  metrics=['accuracy'])

model_new.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 150, 150, 16)      448       
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 75, 75, 16)        0         
_________________________________________________________________
dropout (Dropout)            (None, 75, 75, 16)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 75, 75, 32)        4640      
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 37, 37, 32)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 37, 37, 64)        18496     
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 18, 18, 64)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 18, 64)        0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 20736)             0         
_________________________________________________________________
dense_4 (Dense)              (None, 512)               10617344  
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513       
=================================================================
Total params: 10,641,441
Trainable params: 10,641,441
Non-trainable params: 0
_________________________________________________________________
```

```
history = model_new.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)
```

```
Epoch 1/15
156/156 [==============================] - 142s 913ms/step - loss: 0.7345 - accuracy: 0.
Epoch 2/15
156/156 [==============================] - 141s 904ms/step - loss: 0.6669 - accuracy: 0.
Epoch 3/15
156/156 [==============================] - 142s 908ms/step - loss: 0.6273 - accuracy: 0.
Epoch 4/15
156/156 [==============================] - 142s 910ms/step - loss: 0.5932 - accuracy: 0.
Epoch 5/15
156/156 [==============================] - 140s 900ms/step - loss: 0.5783 - accuracy: 0.
Epoch 6/15
156/156 [==============================] - 141s 905ms/step - loss: 0.5563 - accuracy: 0.
Epoch 7/15
156/156 [==============================] - 141s 906ms/step - loss: 0.5455 - accuracy: 0.
Epoch 8/15
156/156 [==============================] - 142s 911ms/step - loss: 0.5360 - accuracy: 0.
Epoch 9/15
156/156 [==============================] - 142s 910ms/step - loss: 0.5211 - accuracy: 0.
Epoch 10/15
156/156 [==============================] - 143s 916ms/step - loss: 0.5080 - accuracy: 0.
Epoch 11/15
156/156 [==============================] - 143s 915ms/step - loss: 0.5036 - accuracy: 0.
Epoch 12/15
156/156 [==============================] - 145s 930ms/step - loss: 0.4947 - accuracy: 0.
Epoch 13/15
156/156 [==============================] - 145s 931ms/step - loss: 0.4862 - accuracy: 0.
Epoch 14/15
156/156 [==============================] - 145s 928ms/step - loss: 0.4746 - accuracy: 0.
Epoch 15/15
156/156 [==============================] - 144s 924ms/step - loss: 0.4704 - accuracy: 0.
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```




```
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model

base_model=MobileNet(weights='imagenet',include_top=False)

x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x)
x=Dense(1024,activation='relu')(x)
x=Dense(512,activation='relu')(x)
preds=Dense(1,activation='softmax')(x)
model=Model(inputs=base_model.input,outputs=preds)
model.summary()
for layer in model.layers:
    layer.trainable=False
model.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['accuracy'])

history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
```

```
    validation_steps=total_val // batch_size
)
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [128, 1
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_4 (InputLayer)         [(None, None, None, 3)]   0         
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, None, None, 3)     0         
_________________________________________________________________
conv1 (Conv2D)               (None, None, None, 32)    864       
_________________________________________________________________
conv1_bn (BatchNormalization (None, None, None, 32)    128       
_________________________________________________________________
conv1_relu (ReLU)            (None, None, None, 32)    0         
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, None, None, 32)    288       
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, None, None, 32)    128       
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, None, None, 32)    0         
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, None, None, 64)    2048      
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, None, None, 64)    256       
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, None, None, 64)    0         
_________________________________________________________________
conv_pad_2 (ZeroPadding2D)   (None, None, None, 64)    0         
_________________________________________________________________
conv_dw_2 (DepthwiseConv2D)  (None, None, None, 64)    576       
_________________________________________________________________
conv_dw_2_bn (BatchNormaliza (None, None, None, 64)    256       
_________________________________________________________________
conv_dw_2_relu (ReLU)        (None, None, None, 64)    0         
_________________________________________________________________
conv_pw_2 (Conv2D)           (None, None, None, 128)   8192      
_________________________________________________________________
conv_pw_2_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_pw_2_relu (ReLU)        (None, None, None, 128)   0         
_________________________________________________________________
conv_dw_3 (DepthwiseConv2D)  (None, None, None, 128)   1152      
_________________________________________________________________
conv_dw_3_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_dw_3_relu (ReLU)        (None, None, None, 128)   0         
_________________________________________________________________
conv_pw_3 (Conv2D)           (None, None, None, 128)   16384     
_________________________________________________________________
conv_pw_3_bn (BatchNormaliza (None, None, None, 128)   512       
_________________________________________________________________
conv_pw_3_relu (ReLU)        (None, None, None, 128)   0         
_________________________________________________________________
conv_pad_4 (ZeroPadding2D)   (None, None, None, 128)   0         
_________________________________________________________________
conv_dw_4 (DepthwiseConv2D)  (None, None, None, 128)   1152      
_________________________________________________________________
```

conv_dw_4_bn (BatchNormaliza	(None, None, None, 128)	512
conv_dw_4_relu (ReLU)	(None, None, None, 128)	0
conv_pw_4 (Conv2D)	(None, None, None, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_pw_4_relu (ReLU)	(None, None, None, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, None, None, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_dw_5_relu (ReLU)	(None, None, None, 256)	0
conv_pw_5 (Conv2D)	(None, None, None, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_pw_5_relu (ReLU)	(None, None, None, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, None, None, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, None, None, 256)	2304
conv_dw_6_bn (BatchNormaliza	(None, None, None, 256)	1024
conv_dw_6_relu (ReLU)	(None, None, None, 256)	0
conv_pw_6 (Conv2D)	(None, None, None, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_6_relu (ReLU)	(None, None, None, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_7_relu (ReLU)	(None, None, None, 512)	0
conv_pw_7 (Conv2D)	(None, None, None, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_7_relu (ReLU)	(None, None, None, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_8_relu (ReLU)	(None, None, None, 512)	0
conv_pw_8 (Conv2D)	(None, None, None, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, None, None, 512)	2048

conv_pw_8_relu (ReLU)	(None, None, None, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_dw_9_relu (ReLU)	(None, None, None, 512)	0
conv_pw_9 (Conv2D)	(None, None, None, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, None, None, 512)	2048
conv_pw_9_relu (ReLU)	(None, None, None, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_dw_10_relu (ReLU)	(None, None, None, 512)	0
conv_pw_10 (Conv2D)	(None, None, None, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_pw_10_relu (ReLU)	(None, None, None, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_dw_11_relu (ReLU)	(None, None, None, 512)	0
conv_pw_11 (Conv2D)	(None, None, None, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_pw_11_relu (ReLU)	(None, None, None, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, None, None, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, None, None, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, None, None, 512)	2048
conv_dw_12_relu (ReLU)	(None, None, None, 512)	0
conv_pw_12 (Conv2D)	(None, None, None, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, None, None, 1024)	4096
conv_pw_12_relu (ReLU)	(None, None, None, 1024)	0
conv_dw_13 (DepthwiseConv2D)	(None, None, None, 1024)	9216
conv_dw_13_bn (BatchNormaliz	(None, None, None, 1024)	4096
conv_dw_13_relu (ReLU)	(None, None, None, 1024)	0

```
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, None, None, 1024)  1048576   
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, None, None, 1024)  4096      
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, None, None, 1024)  0         
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1024)              0         
_________________________________________________________________
dense_14 (Dense)             (None, 1024)              1049600   
_________________________________________________________________
dense_15 (Dense)             (None, 1024)              1049600   
_________________________________________________________________
dense_16 (Dense)             (None, 512)               524800    
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 513       
=================================================================
Total params: 5,853,377
Trainable params: 5,831,489
Non-trainable params: 21,888
_________________________________________________________________
Epoch 1/15
156/156 [==============================] - 148s 949ms/step - loss: 7.5925 - accuracy: 0.
Epoch 2/15
156/156 [==============================] - 146s 934ms/step - loss: 7.6391 - accuracy: 0.
Epoch 3/15
156/156 [==============================] - 144s 925ms/step - loss: 7.5925 - accuracy: 0.
Epoch 4/15
156/156 [==============================] - 144s 926ms/step - loss: 7.6513 - accuracy: 0.
Epoch 5/15
156/156 [==============================] - 146s 933ms/step - loss: 7.6628 - accuracy: 0.
Epoch 6/15
156/156 [==============================] - 146s 935ms/step - loss: 7.5987 - accuracy: 0.
Epoch 7/15
156/156 [==============================] - 147s 941ms/step - loss: 7.6582 - accuracy: 0.
Epoch 8/15
156/156 [==============================] - 146s 936ms/step - loss: 7.5987 - accuracy: 0.
Epoch 9/15
156/156 [==============================] - 146s 934ms/step - loss: 7.6086 - accuracy: 0.
Epoch 10/15
156/156 [==============================] - 144s 925ms/step - loss: 7.6391 - accuracy: 0.
Epoch 11/15
156/156 [==============================] - 144s 921ms/step - loss: 7.6071 - accuracy: 0.
Epoch 12/15
156/156 [==============================] - 145s 927ms/step - loss: 7.6559 - accuracy: 0.
Epoch 13/15
156/156 [==============================] - 145s 932ms/step - loss: 7.6598 - accuracy: 0.
Epoch 14/15
156/156 [==============================] - 146s 937ms/step - loss: 7.6032 - accuracy: 0.
Epoch 15/15
156/156 [==============================] - 145s 932ms/step - loss: 7.6399 - accuracy: 0.
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
```

```
val_acc = history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```