```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/


!kaggle datasets download -d salader/dogs-vs-cats
```

```
Dataset URL: https://www.kaggle.com/datasets/salader/dogs-vs-cats
License(s): unknown
Downloading dogs-vs-cats.zip to /content
 99% 1.06G/1.06G [00:11<00:00, 90.9MB/s]
100% 1.06G/1.06G [00:11<00:00, 101MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip','r')
zip_ref.extractall('/content')
zip_ref.close()


import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPool2D,Flatten,BatchNormalization,Dropout


#generators -create batches
train_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/train',
    labels = 'inferred',
    label_mode = 'int',
    batch_size = 32,
    image_size = (256,256)
)

validation_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/test',
    labels = 'inferred',
    label_mode = 'int',
    batch_size = 32,
    image_size = (256,256)
)
```

```
Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.
```

```
def process(image,label):
    image = tf.cast(image/255.,tf.float32)
    return image,label

    train_ds = train_ds.map(process)
    validation_ds = validation_ds.map(process)


model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(keras.layers.MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(keras.layers.MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 batch_normalization (Batch  (None, 254, 254, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_1 (Bat  (None, 125, 125, 64)      256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 batch_normalization_2 (Bat  (None, 60, 60, 128)       512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 14848193 (56.64 MB)
Trainable params: 14847745 (56.64 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

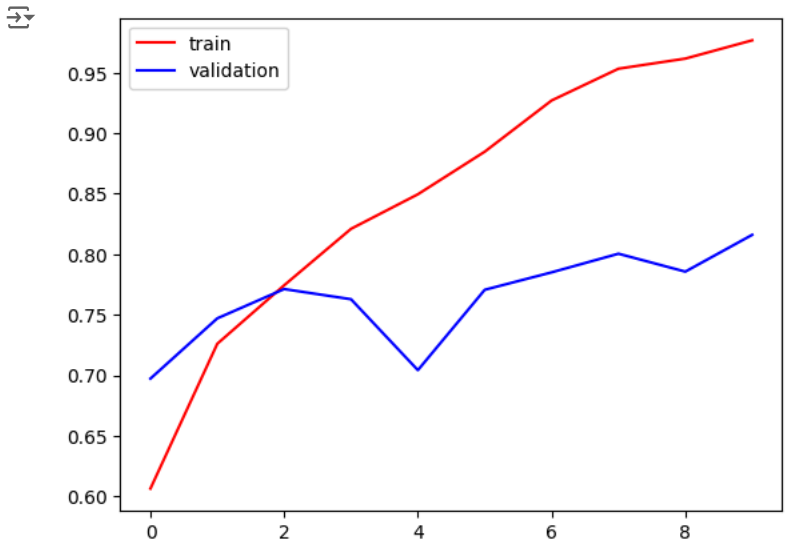
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

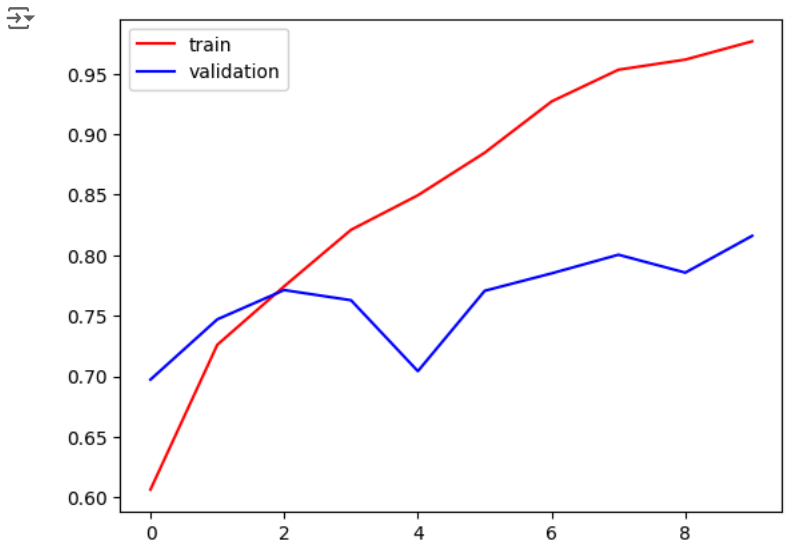
```
Epoch 1/10
625/625 [==============================] - 76s 106ms/step - loss: 1.2805 - accuracy: 0.6064 - val_loss: 0.5644 - val_accuracy
Epoch 2/10
625/625 [==============================] - 64s 102ms/step - loss: 0.5450 - accuracy: 0.7260 - val_loss: 0.5160 - val_accuracy
Epoch 3/10
625/625 [==============================] - 67s 107ms/step - loss: 0.4705 - accuracy: 0.7743 - val_loss: 0.4793 - val_accuracy
Epoch 4/10
625/625 [==============================] - 64s 102ms/step - loss: 0.4030 - accuracy: 0.8209 - val_loss: 0.5812 - val_accuracy
Epoch 5/10
625/625 [==============================] - 64s 102ms/step - loss: 0.3480 - accuracy: 0.8495 - val_loss: 1.1537 - val_accuracy
Epoch 6/10
625/625 [==============================] - 64s 102ms/step - loss: 0.2797 - accuracy: 0.8847 - val_loss: 0.5869 - val_accuracy
Epoch 7/10
625/625 [==============================] - 64s 101ms/step - loss: 0.1901 - accuracy: 0.9270 - val_loss: 0.5883 - val_accuracy
Epoch 8/10
625/625 [==============================] - 63s 100ms/step - loss: 0.1253 - accuracy: 0.9532 - val_loss: 0.6223 - val_accuracy
Epoch 9/10
625/625 [==============================] - 64s 102ms/step - loss: 0.1011 - accuracy: 0.9616 - val_loss: 0.7425 - val_accuracy
Epoch 10/10
625/625 [==============================] - 64s 102ms/step - loss: 0.0698 - accuracy: 0.9767 - val_loss: 0.6736 - val_accuracy
```

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```

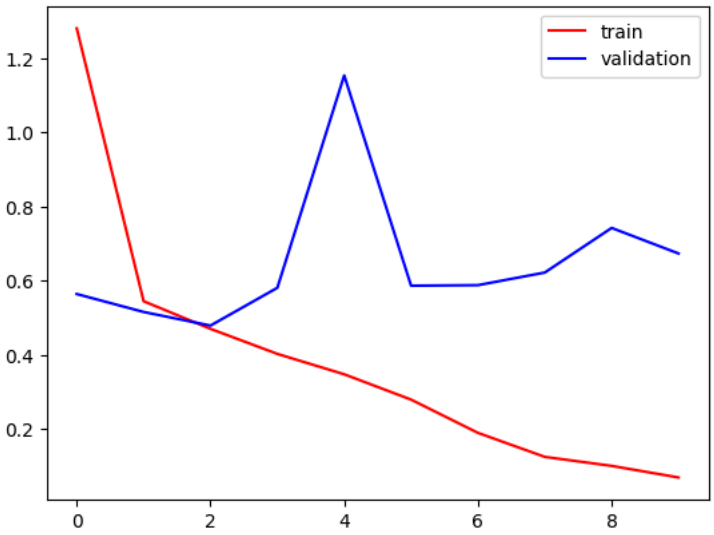



```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```



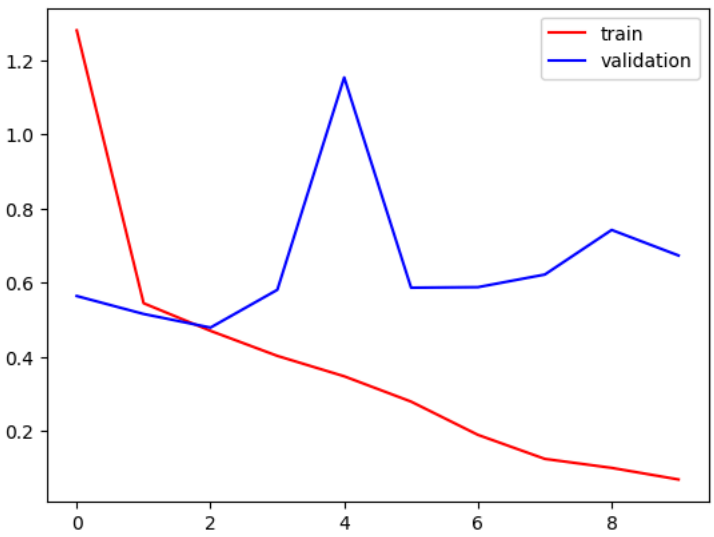

```
plt.plot(history.history['loss'],color='red',label="train")
plt.plot(history.history['val_loss'],color='blue',label='validation')
plt.legend()
plt.show()
```

[→]




```
plt.plot(history.history['loss'],color='red',label="train")
plt.plot(history.history['val_loss'],color='blue',label='validation')
plt.legend()
plt.show()
```

[→]




```
import cv2

test_image = cv2.imread('/content/test/cats/cat.10007.jpg')

plt.imshow(test_image)
```

```
<matplotlib.image.AxesImage at 0x7acc4c1044f0>
```

```
test_image.shape
```

```
(280, 300, 3)
```

```
test_image = cv2.resize(test_image,(256,256))
```

```
test_input = test_image.reshape(1,256,256,3)
```

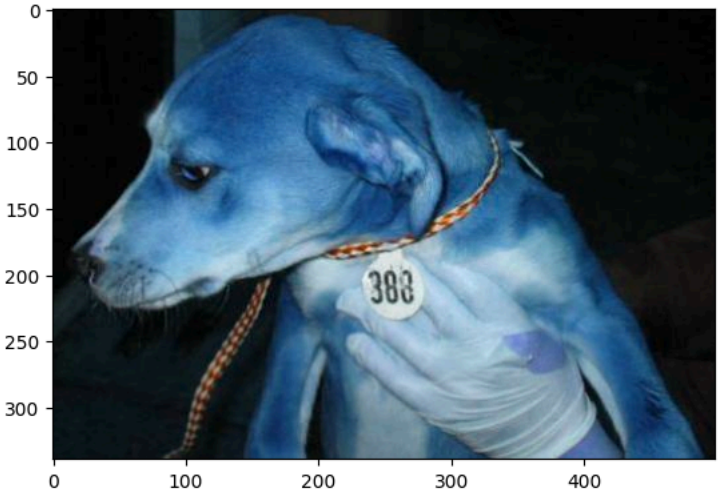
```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 432ms/step
array([[5.1001283e-05]], dtype=float32)
```

```
test_image = cv2.imread('/content/test/dogs/dog.10006.jpg')
```

```
plt.imshow(test_image)
```

```
<matplotlib.image.AxesImage at 0x7acd48221ea0>
```




```
test_image.shape
```

```
(339, 499, 3)
```

```
test_image = cv2.resize(test_image,(256,256))
```

```
test_input = test_image.reshape(1,256,256,3)
```

```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 108ms/step
array([[0.111521]], dtype=float32)
```