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

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

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this,
Downloading dogs-vs-cats.zip to /content
 99% 1.05G/1.06G [00:06<00:00, 337MB/s]
100% 1.06G/1.06G [00:06<00:00, 165MB/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, MaxPooling2D, Flatten
```

```
# generators
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.
```

```
# Normalize
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)
```

```
#create CNN model
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
from keras.models import Sequential

model = Sequential()

# First convolutional layer
model.add(Conv2D(32, kernel_size=(3, 3), padding='valid', activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))

# Second convolutional layer
model.add(Conv2D(64, kernel_size=(3, 3), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))

# Third convolutional layer
model.add(Conv2D(128, kernel_size=(3, 3), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))

model.add(Flatten())

# Fully connected layers
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))

# Output layer for binary classification
```

```
model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
```

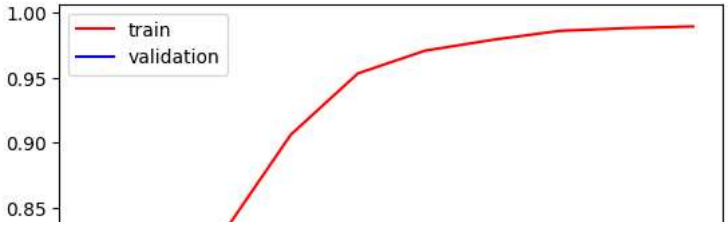
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

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

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

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

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

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

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 14847297 (56.64 MB)
Trainable params: 14847297 (56.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

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

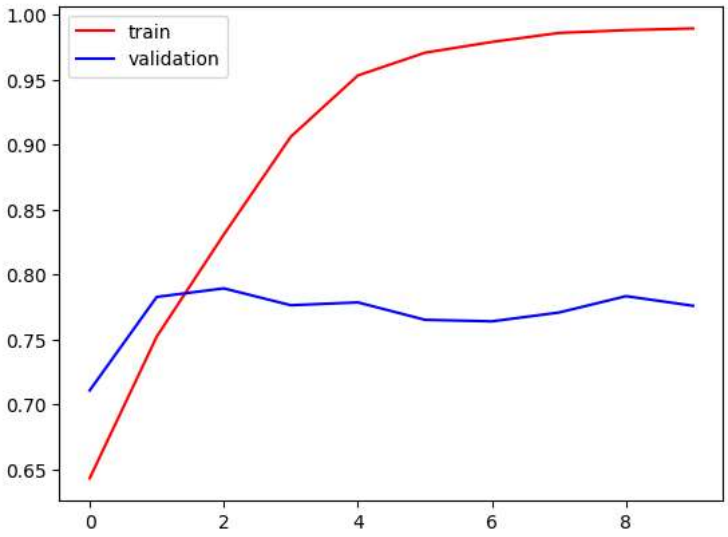
```
history = model.fit(train_ds,epochs=10,validation_data = validation_ds)
```

```
Epoch 1/10
625/625 [==============================] - 64s 79ms/step - loss: 0.6346 - accuracy: 0
Epoch 2/10
625/625 [==============================] - 54s 85ms/step - loss: 0.5023 - accuracy: 0
Epoch 3/10
625/625 [==============================] - 54s 87ms/step - loss: 0.3720 - accuracy: 0
Epoch 4/10
625/625 [==============================] - 51s 81ms/step - loss: 0.2259 - accuracy: 0
Epoch 5/10
625/625 [==============================] - 50s 80ms/step - loss: 0.1238 - accuracy: 0
Epoch 6/10
625/625 [==============================] - 57s 91ms/step - loss: 0.0797 - accuracy: 0
Epoch 7/10
625/625 [==============================] - 52s 83ms/step - loss: 0.0609 - accuracy: 0
Epoch 8/10
625/625 [==============================] - 51s 81ms/step - loss: 0.0440 - accuracy: 0.9861 - val_loss: 1.4157 - val_accuracy: 0.770
Epoch 9/10
625/625 [==============================] - 51s 81ms/step - loss: 0.0373 - accuracy: 0.9883 - val_loss: 1.5539 - val_accuracy: 0.783
Epoch 10/10
625/625 [==============================] - 53s 84ms/step - loss: 0.0329 - accuracy: 0.9895 - val_loss: 1.4770 - val_accuracy: 0.776
```

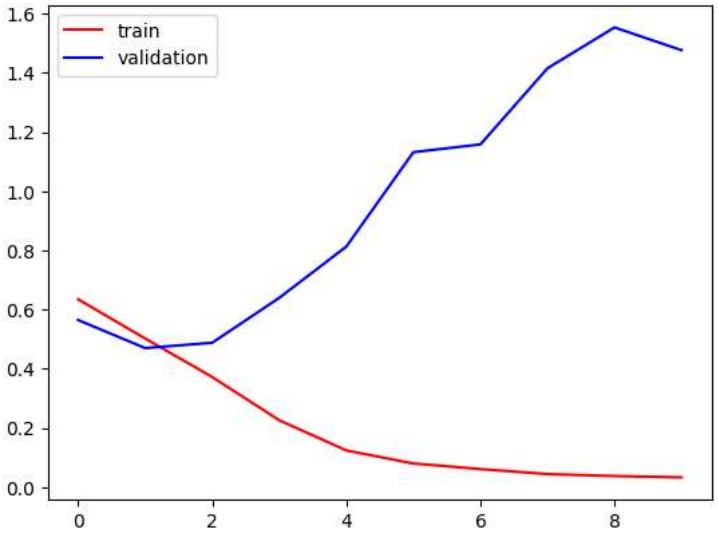
```
import matplotlib.pyplot as plt

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



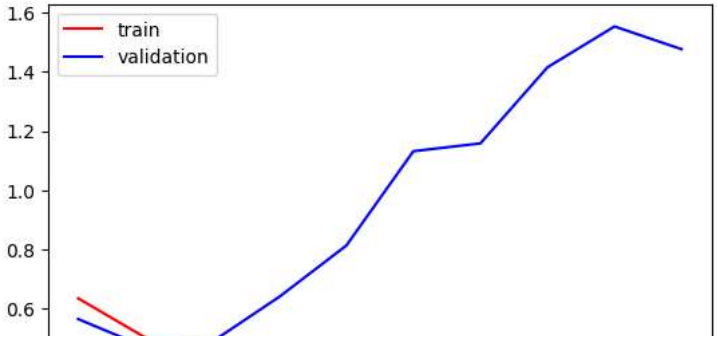

```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```




```
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
from PIL import Image
import matplotlib.pyplot as plt
```

```
test_img = cv2.imread('/content/cat.jpg')
```

```
# Display the image
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7d9d3a9dbf10>
```


```
test_img.shape
```

```
(1200, 1920, 3)
```

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

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

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

```
1/1 [==============================] - 0s 241ms/step
array([[0.]], dtype=float32)
```