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

```
!kaggle datasets download -d samuelcortinhas/cats-and-dogs-image-classification
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.js
Downloading cats-and-dogs-image-classification.zip to /content
 96% 62.0M/64.4M [00:02<00:00, 30.9MB/s]
100% 64.4M/64.4M [00:02<00:00, 23.8MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/cats-and-dogs-image-classification.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,BatchNormalization,Dropout
```

```
# 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/train',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)
```

```
Found 557 files belonging to 2 classes.
Found 557 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

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(BatchNormalization())
model.add(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(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(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"
_________________________________________________________________
```

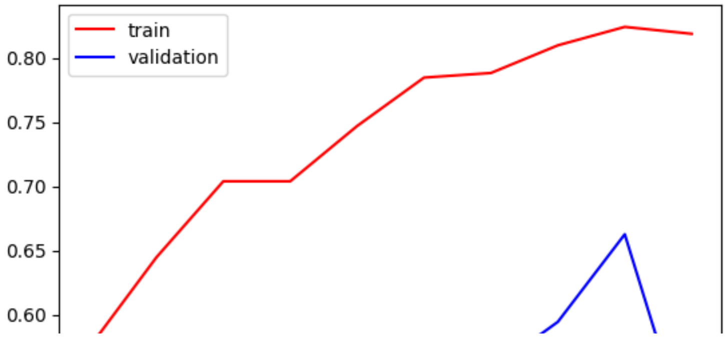
```
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896       
                                                                 
 batch_normalization (Batch  (None, 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'])
```

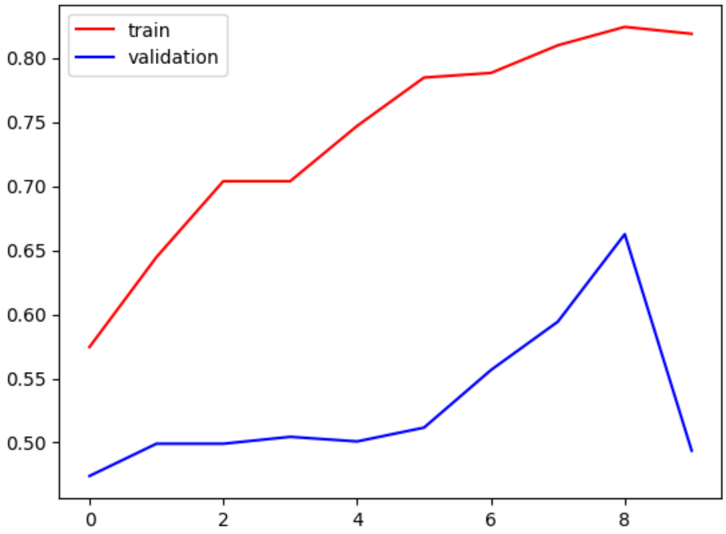
```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

```
Epoch 1/10
18/18 [==============================] - 20s 265ms/step - loss: 6.3124 - accuracy: 0.5745 - val_loss: 3.4313 - val_accuracy: 0.4740
Epoch 2/10
18/18 [==============================] - 7s 285ms/step - loss: 5.1244 - accuracy: 0.6445 - val_loss: 10.0341 - val_accuracy: 0.4991
Epoch 3/10
18/18 [==============================] - 6s 296ms/step - loss: 2.2824 - accuracy: 0.7038 - val_loss: 12.8366 - val_accuracy: 0.4991
Epoch 4/10
18/18 [==============================] - 6s 251ms/step - loss: 2.3799 - accuracy: 0.7038 - val_loss: 4.7074 - val_accuracy: 0.5045
Epoch 5/10
18/18 [==============================] - 6s 261ms/step - loss: 1.8279 - accuracy: 0.7469 - val_loss: 14.9632 - val_accuracy: 0.5009
Epoch 6/10
18/18 [==============================] - 5s 250ms/step - loss: 1.9379 - accuracy: 0.7846 - val_loss: 8.7747 - val_accuracy: 0.5117
Epoch 7/10
18/18 [==============================] - 8s 429ms/step - loss: 1.3015 - accuracy: 0.7882 - val_loss: 6.2203 - val_accuracy: 0.5566
Epoch 8/10
18/18 [==============================] - 5s 243ms/step - loss: 1.3880 - accuracy: 0.8097 - val_loss: 3.1116 - val_accuracy: 0.5943
Epoch 9/10
18/18 [==============================] - 6s 286ms/step - loss: 1.2661 - accuracy: 0.8241 - val_loss: 1.3596 - val_accuracy: 0.6625
Epoch 10/10
18/18 [==============================] - 5s 242ms/step - loss: 1.2965 - accuracy: 0.8187 - val_loss: 2.7764 - val_accuracy: 0.4937
```

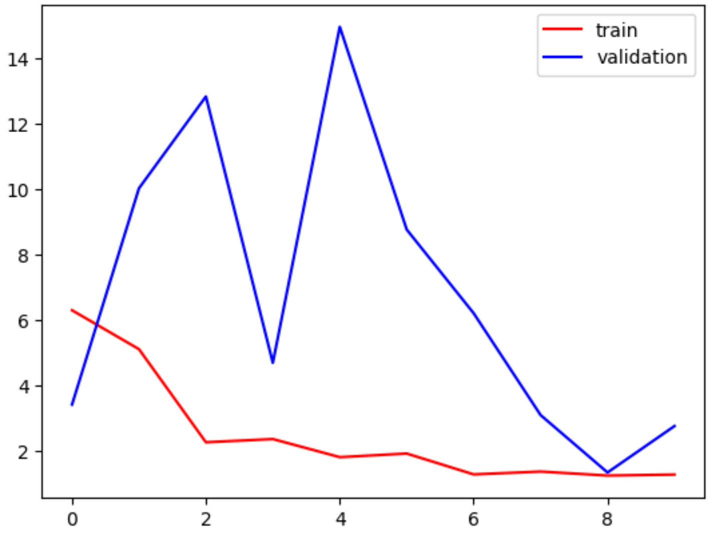
```
import matplotlib.pyplot as plt

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



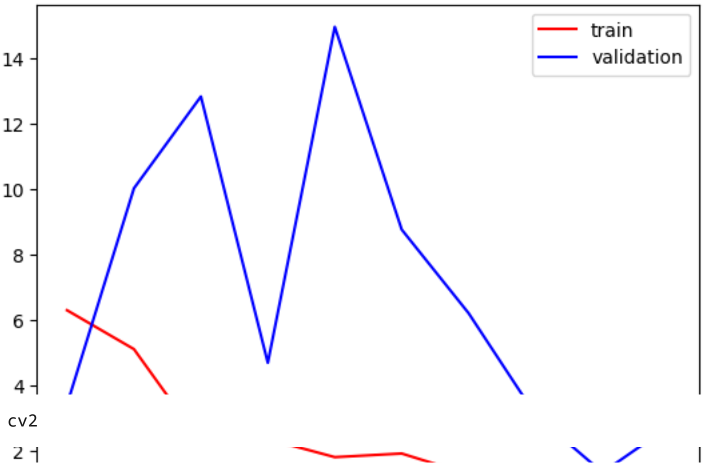
```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```



```
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()
```



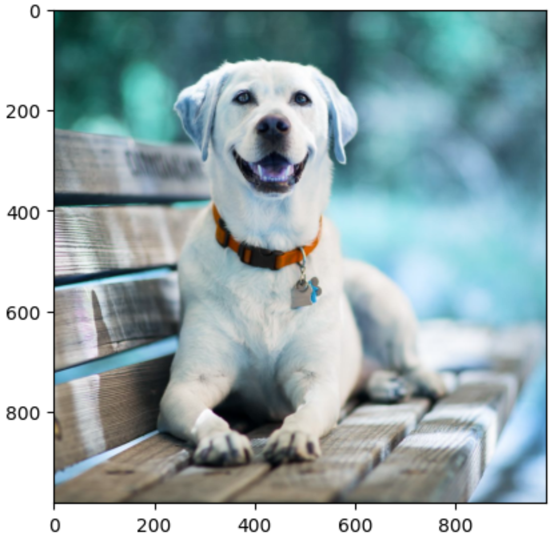
```
import cv2
```

```
test_img = cv2.imread('/content/cat.jpg')
plt.imshow(test_img)
test_img = cv2.resize(test_img, (256, 256))
test_input = test_img.reshape((1, 256, 256, 3)) / 255.  # Normalize the test image
```

```
prediction = model.predict(test_input)
print("Prediction:", prediction)
```

```
    1/1 [==============================] - 1s 625ms/step
    Prediction: [[0.20155276]]
```

```
test_img = cv2.imread('/content/dog.jpg')
plt.imshow(test_img)
test_img = cv2.resize(test_img, (256, 256))
test_input = test_img.reshape((1, 256, 256, 3)) / 255.  # Normalize the test image
```



```
prediction = model.predict(test_input)
print("Prediction:", prediction)
```

```
1/1 [==============================] - 0s 27ms/step
Prediction: [[0.08369035]]
```

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

history = model.fit(train_ds, epochs=20, validation_data=validation_ds)
```

```
Epoch 1/20
18/18 [==============================] - 10s 257ms/step - loss: 1.5615 - accuracy: 0.8276 - val_loss: 19.2181 - val_accuracy: 0.500
Epoch 2/20
18/18 [==============================] - 7s 326ms/step - loss: 1.4516 - accuracy: 0.8151 - val_loss: 19.1029 - val_accuracy: 0.5009
Epoch 3/20
18/18 [==============================] - 6s 264ms/step - loss: 1.2592 - accuracy: 0.8384 - val_loss: 23.4758 - val_accuracy: 0.5009
Epoch 4/20
18/18 [==============================] - 5s 262ms/step - loss: 0.9870 - accuracy: 0.8743 - val_loss: 4.8519 - val_accuracy: 0.4991
Epoch 5/20
18/18 [==============================] - 6s 264ms/step - loss: 0.8796 - accuracy: 0.8815 - val_loss: 16.9837 - val_accuracy: 0.4991
Epoch 6/20
18/18 [==============================] - 5s 244ms/step - loss: 0.4551 - accuracy: 0.9246 - val_loss: 7.7307 - val_accuracy: 0.4991
Epoch 7/20
18/18 [==============================] - 6s 264ms/step - loss: 0.3452 - accuracy: 0.9354 - val_loss: 9.1421 - val_accuracy: 0.4991
Epoch 8/20
18/18 [==============================] - 6s 264ms/step - loss: 0.2563 - accuracy: 0.9479 - val_loss: 14.6302 - val_accuracy: 0.4991
Epoch 9/20
18/18 [==============================] - 6s 273ms/step - loss: 0.3344 - accuracy: 0.9551 - val_loss: 2.0666 - val_accuracy: 0.5009
Epoch 10/20
18/18 [==============================] - 6s 314ms/step - loss: 0.3430 - accuracy: 0.9605 - val_loss: 7.7986 - val_accuracy: 0.4991
Epoch 11/20
18/18 [==============================] - 6s 264ms/step - loss: 0.1465 - accuracy: 0.9803 - val_loss: 19.6123 - val_accuracy: 0.5009
Epoch 12/20
18/18 [==============================] - 6s 264ms/step - loss: 0.1410 - accuracy: 0.9767 - val_loss: 16.3161 - val_accuracy: 0.5009
Epoch 13/20
18/18 [==============================] - 6s 319ms/step - loss: 0.1334 - accuracy: 0.9767 - val_loss: 19.5058 - val_accuracy: 0.5009
Epoch 14/20
18/18 [==============================] - 6s 265ms/step - loss: 0.0733 - accuracy: 0.9838 - val_loss: 32.4718 - val_accuracy: 0.5009
Epoch 15/20
18/18 [==============================] - 8s 413ms/step - loss: 0.2472 - accuracy: 0.9785 - val_loss: 25.5246 - val_accuracy: 0.5009
Epoch 16/20
18/18 [==============================] - 6s 266ms/step - loss: 0.3528 - accuracy: 0.9497 - val_loss: 19.0979 - val_accuracy: 0.5009
Epoch 17/20
18/18 [==============================] - 7s 322ms/step - loss: 0.2538 - accuracy: 0.9569 - val_loss: 33.0052 - val_accuracy: 0.5009
Epoch 18/20
18/18 [==============================] - 5s 247ms/step - loss: 0.1665 - accuracy: 0.9695 - val_loss: 28.6201 - val_accuracy: 0.5009
Epoch 19/20
18/18 [==============================] - 6s 264ms/step - loss: 0.2495 - accuracy: 0.9641 - val_loss: 31.7087 - val_accuracy: 0.5009
Epoch 20/20
18/18 [==============================] - 7s 291ms/step - loss: 0.2037 - accuracy: 0.9767 - val_loss: 31.0354 - val_accuracy: 0.5009
```

```
# Class mapping
class_mapping = {0: 'Cat', 1: 'Dog'}

# Load test images
test_img_cat = cv2.imread('/content/cat.jpg')
test_img_cat = cv2.resize(test_img_cat, (256, 256))
test_input_cat = test_img_cat.reshape((1, 256, 256, 3)) / 255.

test_img_dog = cv2.imread('/content/dog.jpg')
test_img_dog = cv2.resize(test_img_dog, (256, 256))
test_input_dog = test_img_dog.reshape((1, 256, 256, 3)) / 255.
```

```
# Model prediction on test images
prediction_cat = model.predict(test_input_cat)
prediction_dog = model.predict(test_input_dog)
```

```
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 125ms/step
```

```
# Interpret predictions using class mapping
class_prediction_cat = class_mapping[int(round(prediction_cat[0][0]))]
class_prediction_dog = class_mapping[int(round(prediction_dog[0][0]))]


print("Prediction for cat:", class_prediction_cat)
print("Prediction for dog:", class_prediction_dog)
```

```
Prediction for cat: Cat
Prediction for dog: Cat
```

```
# Evaluate the model on a test set (assuming you have a separate test directory)
test_ds = keras.utils.image_dataset_from_directory(
    directory='/content/test',
    labels='inferred',
    label_mode='int',
    batch_size=32,
    image_size=(256, 256)
)

test_ds = test_ds.map(process)

test_loss, test_accuracy = model.evaluate(test_ds)
print(f'Test Accuracy: {test_accuracy}')
print(f'Test Loss: {test_loss}')
```

```
Found 140 files belonging to 2 classes.
5/5 [==============================] - 1s 92ms/step - loss: 31.0931 - accuracy: 0.5000
Test Accuracy: 0.5
Test Loss: 31.093067169189453
```