```
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,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/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

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'))
```

```
[9] 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
```

```
[9]  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)
_________________________________________________________________
```

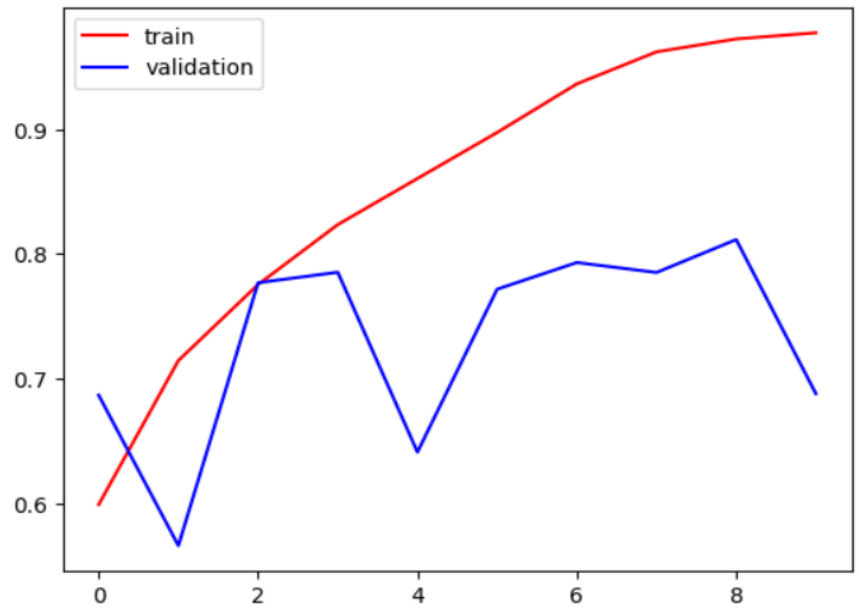
```
[10] model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

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

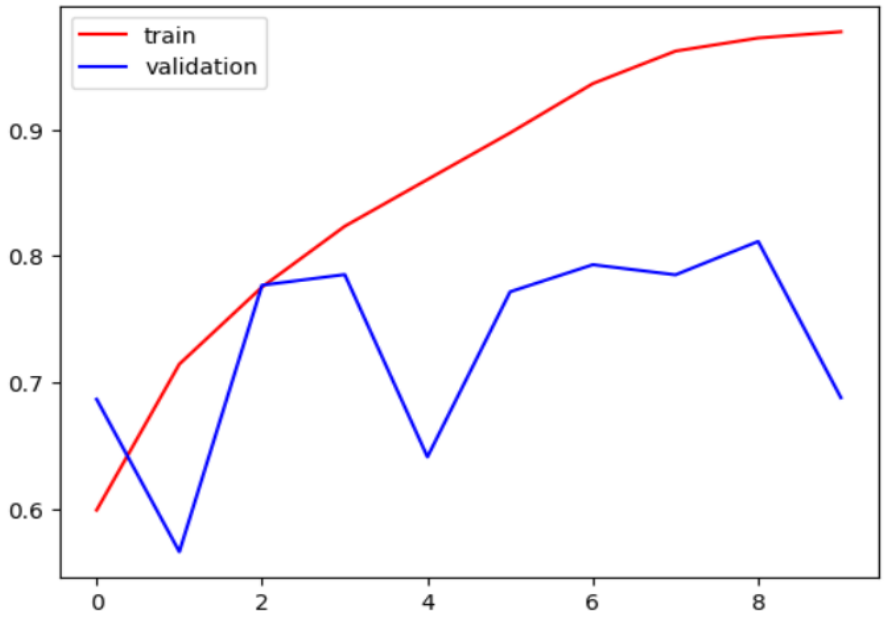
```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

```
Epoch 1/10
625/625 [==============================] - 75s 104ms/step - loss: 1.0438 - accuracy: 0.5992 - val_loss: 0.5837 - val_accuracy: 0.6868
Epoch 2/10
625/625 [==============================] - 64s 102ms/step - loss: 0.5629 - accuracy: 0.7146 - val_loss: 0.7096 - val_accuracy: 0.5662
Epoch 3/10
625/625 [==============================] - 65s 103ms/step - loss: 0.4755 - accuracy: 0.7756 - val_loss: 0.4646 - val_accuracy: 0.7770
Epoch 4/10
625/625 [==============================] - 66s 105ms/step - loss: 0.3955 - accuracy: 0.8236 - val_loss: 0.4488 - val_accuracy: 0.7854
Epoch 5/10
625/625 [==============================] - 64s 102ms/step - loss: 0.3202 - accuracy: 0.8605 - val_loss: 0.6793 - val_accuracy: 0.6412
Epoch 6/10
625/625 [==============================] - 64s 102ms/step - loss: 0.2448 - accuracy: 0.8974 - val_loss: 0.6348 - val_accuracy: 0.7718
Epoch 7/10
625/625 [==============================] - 64s 101ms/step - loss: 0.1623 - accuracy: 0.9364 - val_loss: 0.8133 - val_accuracy: 0.7932
Epoch 8/10
625/625 [==============================] - 67s 107ms/step - loss: 0.1043 - accuracy: 0.9621 - val_loss: 0.7203 - val_accuracy: 0.7852
Epoch 9/10
625/625 [==============================] - 64s 102ms/step - loss: 0.0821 - accuracy: 0.9725 - val_loss: 0.7915 - val_accuracy: 0.8116
Epoch 10/10
625/625 [==============================] - 66s 106ms/step - loss: 0.0658 - accuracy: 0.9774 - val_loss: 1.2414 - val_accuracy: 0.6880
```

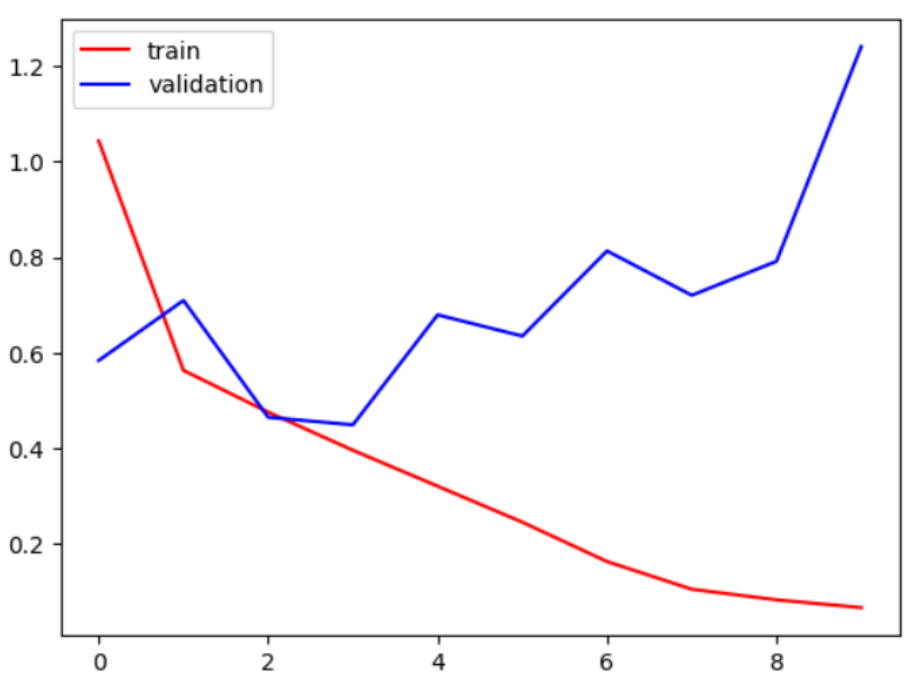
```
[12] import matplotlib.pyplot as plt

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



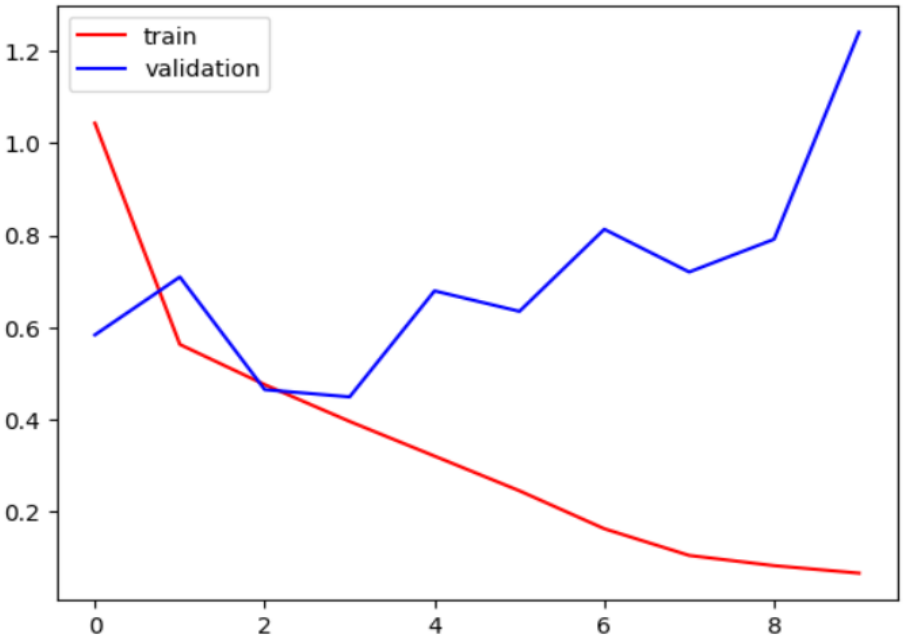
```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```




```
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()
```




```
[17] import cv2
```

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

```
[21] plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7d19d4150ca0>
```

```
[24] from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[25] test_img.shape
```

```
(463, 703, 3)
```

```
[26] test_img = cv2.resize(test_img,(256,256))
```

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

```
[28] model.predict(test_input)
```

```
1/1 [==============================] - 0s 382ms/step
array([[0.]], dtype=float32)
```

```
import cv2
```

```
[33] test_img = cv2.imread('/content/images.jpeg')
```

```
[34] plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7d19c0313f40>
```

```
[35] from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[36] test_img.shape
```

```
(205, 246, 3)
```

```
[37] test_img = cv2.resize(test_img,(256,256))
```

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

```
[40] model.predict(test_input)
```

```
1/1 [==============================] - 0s 18ms/step
array([[0.]], dtype=float32)
```