```
!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, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading dogs-vs-cats.zip to /content
 99% 1.05G/1.06G [00:06<00:00, 233MB/s]
100% 1.06G/1.06G [00:06<00:00, 171MB/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,BatchNormalization,Dropout
```

Generators are used to process large amount of data

```
# 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.
```

Normalization -To convert pixel values to 0 to 1

```
# 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)
```

Creating cnn model

```
# 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)
_________________________________________________________________
```

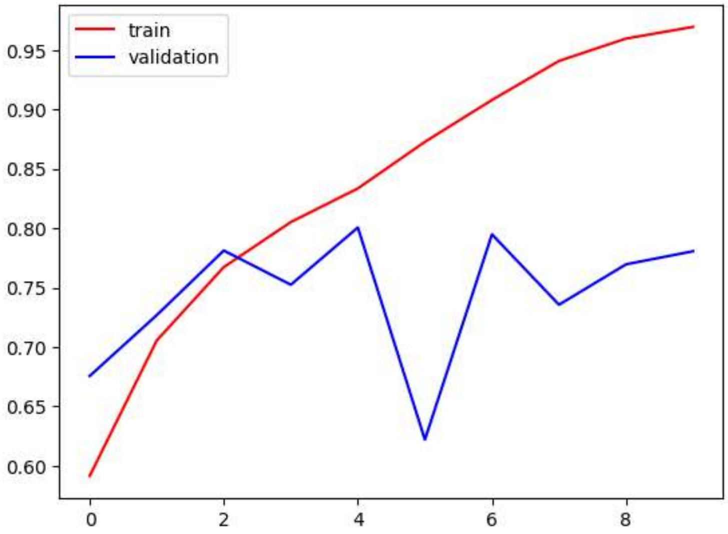
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

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

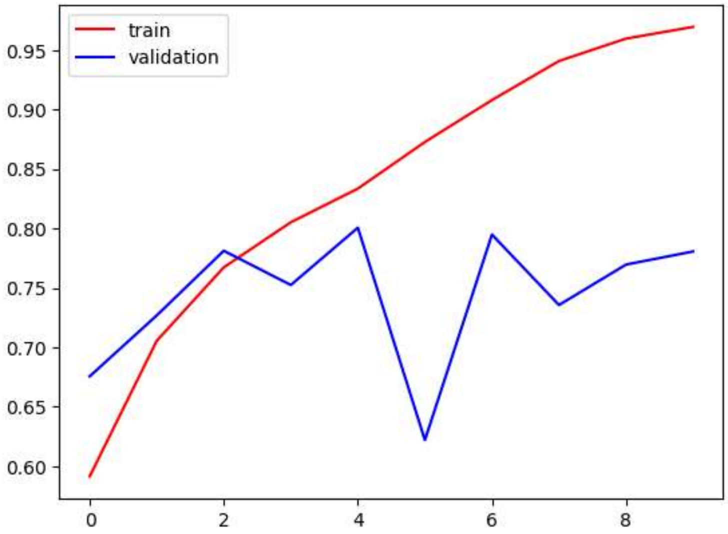
```
Epoch 1/10
625/625 [==============================] - 81s 108ms/step - loss: 1.4467 - accuracy: 0.5914 - val_loss: 0.6136 - val_accuracy: 0.6756
Epoch 2/10
625/625 [==============================] - 63s 101ms/step - loss: 0.5810 - accuracy: 0.7054 - val_loss: 0.5403 - val_accuracy: 0.7268
Epoch 3/10
625/625 [==============================] - 64s 102ms/step - loss: 0.4778 - accuracy: 0.7673 - val_loss: 0.4698 - val_accuracy: 0.7812
Epoch 4/10
625/625 [==============================] - 64s 103ms/step - loss: 0.4212 - accuracy: 0.8051 - val_loss: 0.5325 - val_accuracy: 0.7524
Epoch 5/10
625/625 [==============================] - 67s 106ms/step - loss: 0.3733 - accuracy: 0.8335 - val_loss: 0.4521 - val_accuracy: 0.8006
Epoch 6/10
625/625 [==============================] - 70s 112ms/step - loss: 0.2989 - accuracy: 0.8727 - val_loss: 1.4874 - val_accuracy: 0.6220
Epoch 7/10
625/625 [==============================] - 68s 108ms/step - loss: 0.2227 - accuracy: 0.9079 - val_loss: 0.5673 - val_accuracy: 0.7948
Epoch 8/10
625/625 [==============================] - 68s 109ms/step - loss: 0.1496 - accuracy: 0.9408 - val_loss: 1.1132 - val_accuracy: 0.7356
```

```
Epoch 9/10
625/625 [==============================] - 65s 103ms/step - loss: 0.1129 - accuracy: 0.9596 - val_loss: 1.1465 - val_accuracy: 0.7696
Epoch 10/10
625/625 [==============================] - 66s 105ms/step - loss: 0.0895 - accuracy: 0.9694 - val_loss: 0.7557 - val_accuracy: 0.7806
```

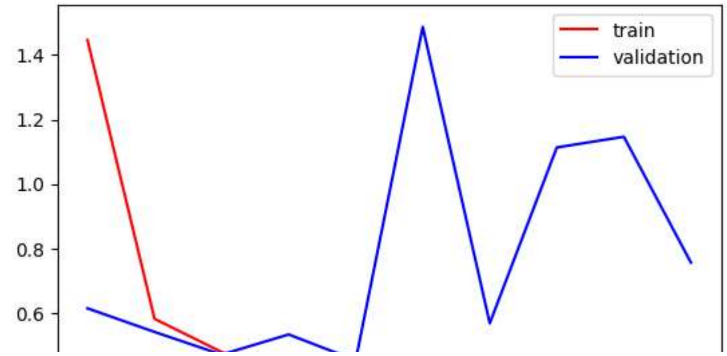
```
import matplotlib.pyplot as plt

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



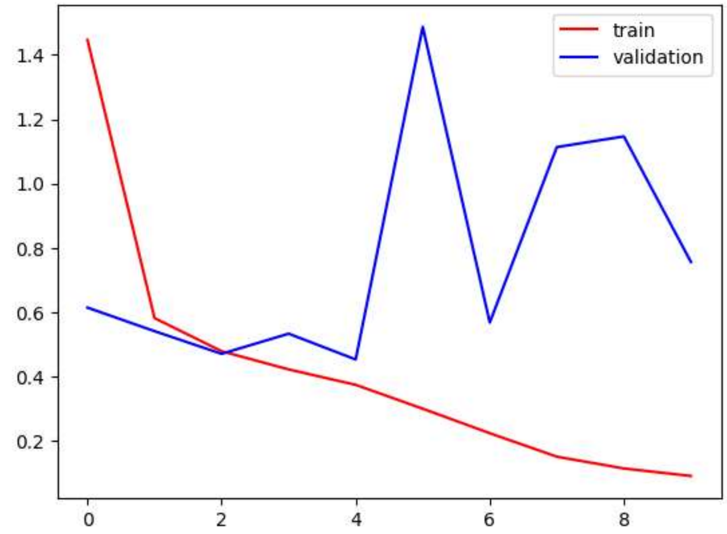
```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```



```
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()
```



```
# ways to reduce overfitting

# Add more data
# Data Augmentation -> next video
# L1/L2 Regularizer
# Dropout
# Batch Norm
# Reduce complexity
```

```
import cv2
```

Testing on Cat

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

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7c75913fef20>
```

```
test_img.shape
```

```
(275, 183, 3)
```

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

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

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

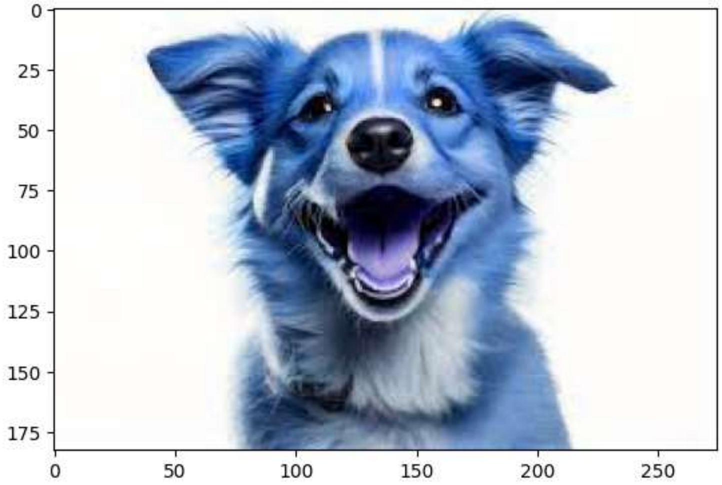
```
1/1 [==============================] - 0s 276ms/step
array([[0.]], dtype=float32)
```

Testing on Dog

```
test_img=cv2.imread('/content/dog.jpeg')
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7c759138fe80>
```



```
test_img.shape
```

```
(183, 275, 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 19ms/step
array([[1.]], dtype=float32)
```