```
!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.js
Downloading dogs-vs-cats.zip to /content
 98% 1.04G/1.06G [00:05<00:00, 189MB/s]
100% 1.06G/1.06G [00:05<00:00, 209MB/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
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'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

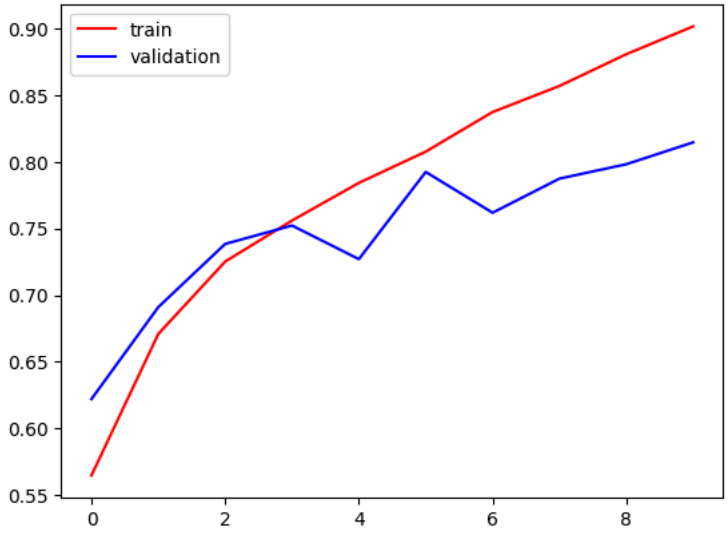
```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 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'])
```

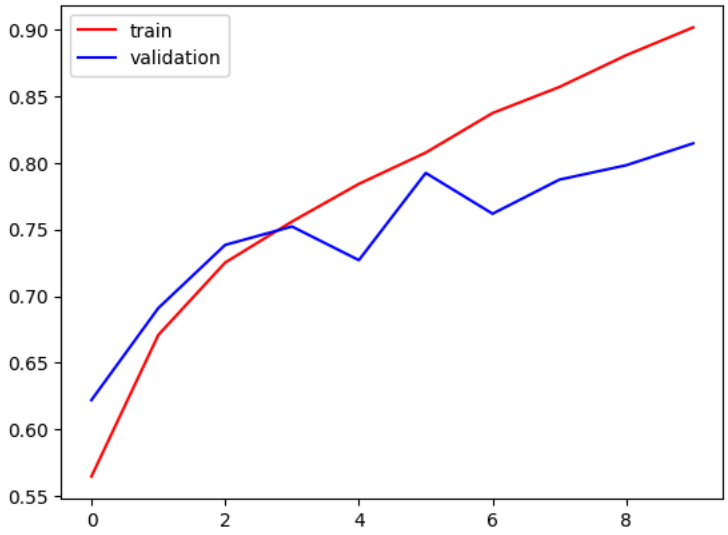
```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

```
Epoch 1/10
625/625 [==============================] - 76s 106ms/step - loss: 1.5541 - accuracy: 0.5647 - val_loss: 0.6517 - val_accuracy: 0.62
Epoch 2/10
625/625 [==============================] - 63s 101ms/step - loss: 0.6022 - accuracy: 0.6708 - val_loss: 0.6874 - val_accuracy: 0.69
Epoch 3/10
625/625 [==============================] - 66s 105ms/step - loss: 0.5365 - accuracy: 0.7251 - val_loss: 0.5369 - val_accuracy: 0.73
Epoch 4/10
625/625 [==============================] - 66s 106ms/step - loss: 0.4903 - accuracy: 0.7559 - val_loss: 0.5125 - val_accuracy: 0.75
Epoch 5/10
625/625 [==============================] - 64s 102ms/step - loss: 0.4457 - accuracy: 0.7842 - val_loss: 0.5626 - val_accuracy: 0.72
Epoch 6/10
625/625 [==============================] - 66s 106ms/step - loss: 0.3965 - accuracy: 0.8077 - val_loss: 0.4451 - val_accuracy: 0.79
Epoch 7/10
625/625 [==============================] - 64s 101ms/step - loss: 0.3479 - accuracy: 0.8374 - val_loss: 0.5433 - val_accuracy: 0.76
Epoch 8/10
625/625 [==============================] - 64s 102ms/step - loss: 0.3084 - accuracy: 0.8569 - val_loss: 0.4940 - val_accuracy: 0.78
Epoch 9/10
625/625 [==============================] - 64s 101ms/step - loss: 0.2588 - accuracy: 0.8808 - val_loss: 0.4975 - val_accuracy: 0.79
Epoch 10/10
625/625 [==============================] - 63s 100ms/step - loss: 0.2140 - accuracy: 0.9016 - val_loss: 0.5979 - val_accuracy: 0.81
```

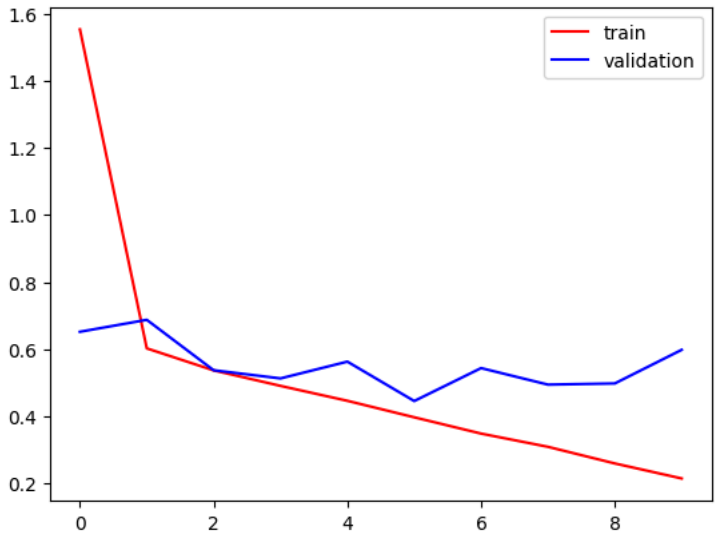
```
import matplotlib.pyplot as plt

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



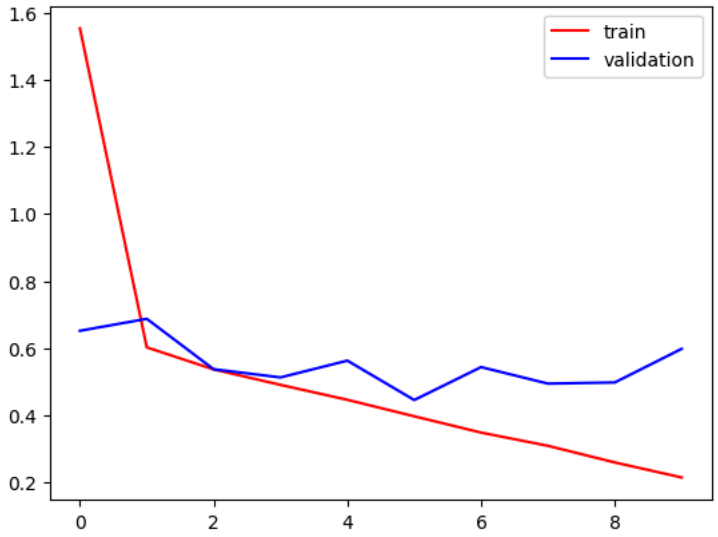

```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```




```
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()
```



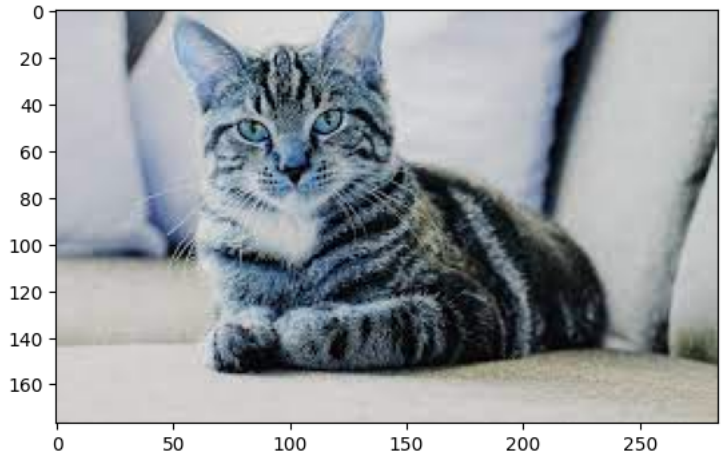

```
# ways to reduce overfitting

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

```
import cv2
```

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

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

```
<matplotlib.image.AxesImage at 0x7fcf40107640>
```




```
test_img.shape
```

```
(177, 284, 3)
```

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

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

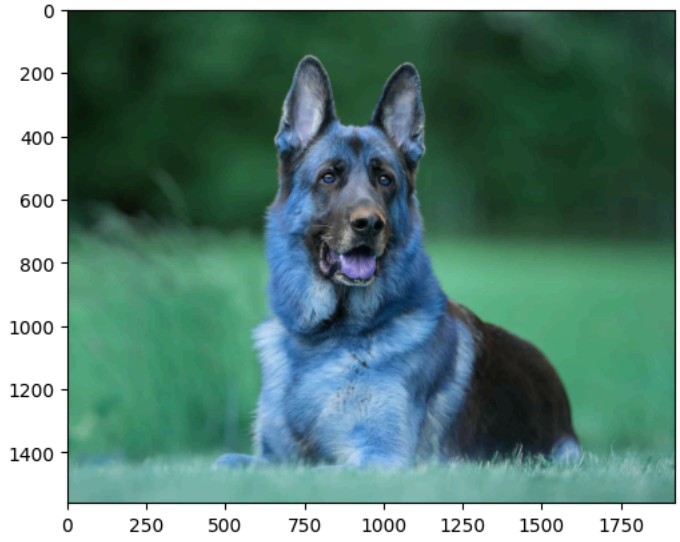
```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 396ms/step
array([[0.]], dtype=float32)
```

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

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

```
<matplotlib.image.AxesImage at 0x7fcf53f87a30>
```




```
test_img.shape
```

```
(1560, 1922, 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 20ms/step
array([[0.]], dtype=float32)
```