```
# Dataset - https://www.kaggle.com/datasets/salader/dogs-vs-cats
```

```
!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
Downloading dogs-vs-cats.zip to /content
100% 1.06G/1.06G [00:50<00:00, 23.9MB/s]
100% 1.06G/1.06G [00:50<00:00, 22.8MB/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 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)
```

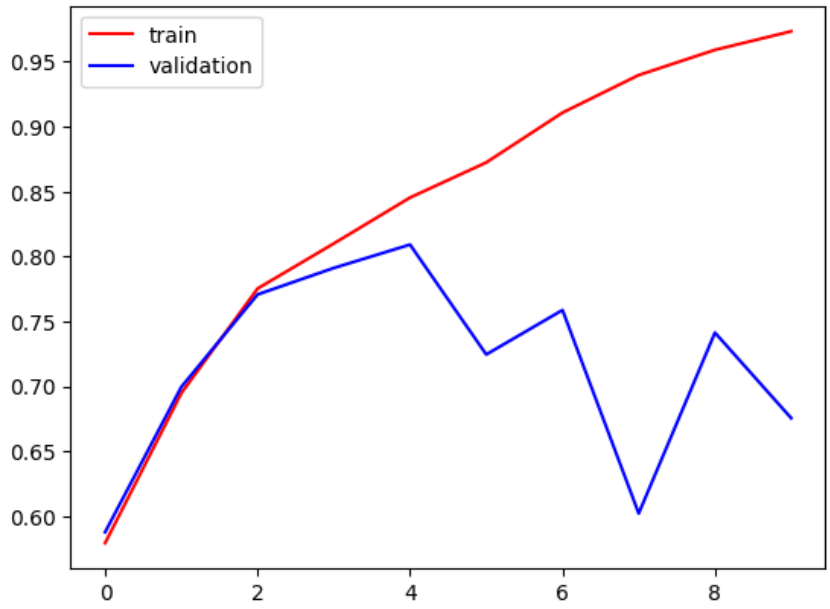
```
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

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

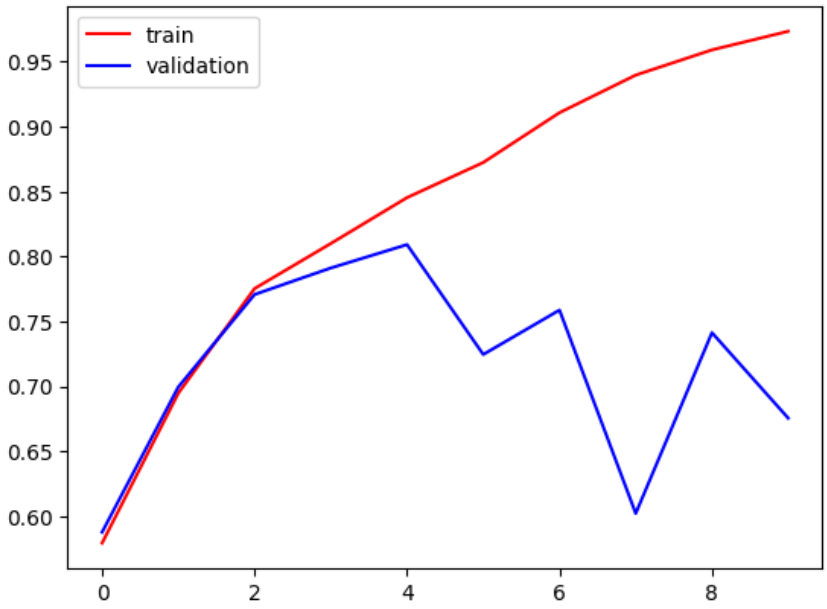
```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

```
Epoch 1/10
625/625 [==============================] - 135s 192ms/step - loss: 1.2646 - accuracy: 0.5794 - val_loss: 0.6
Epoch 2/10
625/625 [==============================] - 123s 196ms/step - loss: 0.5729 - accuracy: 0.6948 - val_loss: 0.5
Epoch 3/10
625/625 [==============================] - 115s 184ms/step - loss: 0.4759 - accuracy: 0.7753 - val_loss: 0.4
Epoch 4/10
625/625 [==============================] - 117s 186ms/step - loss: 0.4164 - accuracy: 0.8101 - val_loss: 0.4
Epoch 5/10
625/625 [==============================] - 112s 178ms/step - loss: 0.3524 - accuracy: 0.8454 - val_loss: 0.4
Epoch 6/10
625/625 [==============================] - 127s 203ms/step - loss: 0.2965 - accuracy: 0.8724 - val_loss: 0.5
Epoch 7/10
625/625 [==============================] - 128s 204ms/step - loss: 0.2139 - accuracy: 0.9108 - val_loss: 0.5
Epoch 8/10
625/625 [==============================] - 134s 215ms/step - loss: 0.1553 - accuracy: 0.9398 - val_loss: 2.7
Epoch 9/10
625/625 [==============================] - 119s 190ms/step - loss: 0.1137 - accuracy: 0.9592 - val_loss: 1.4
Epoch 10/10
625/625 [==============================] - 131s 209ms/step - loss: 0.0789 - accuracy: 0.9735 - val_loss: 2.7
```

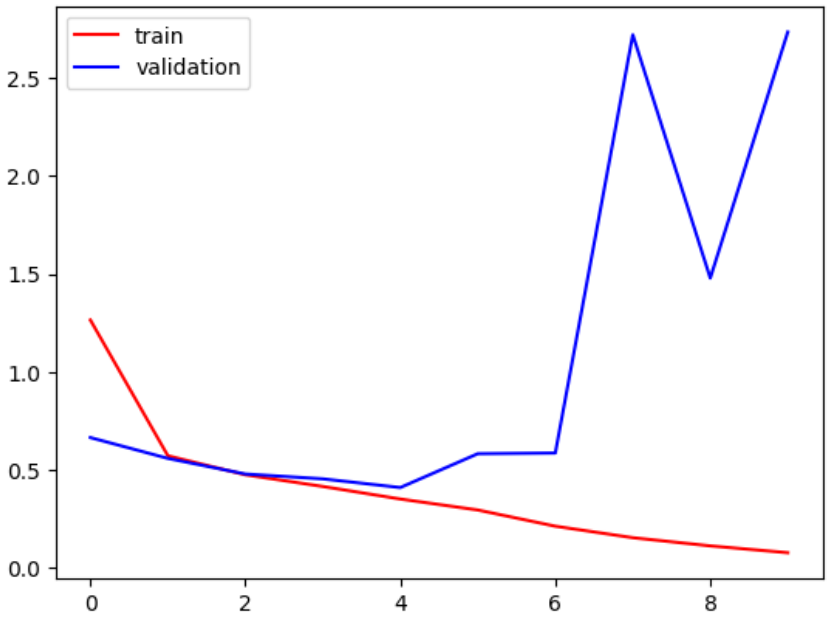
```
import matplotlib.pyplot as plt

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



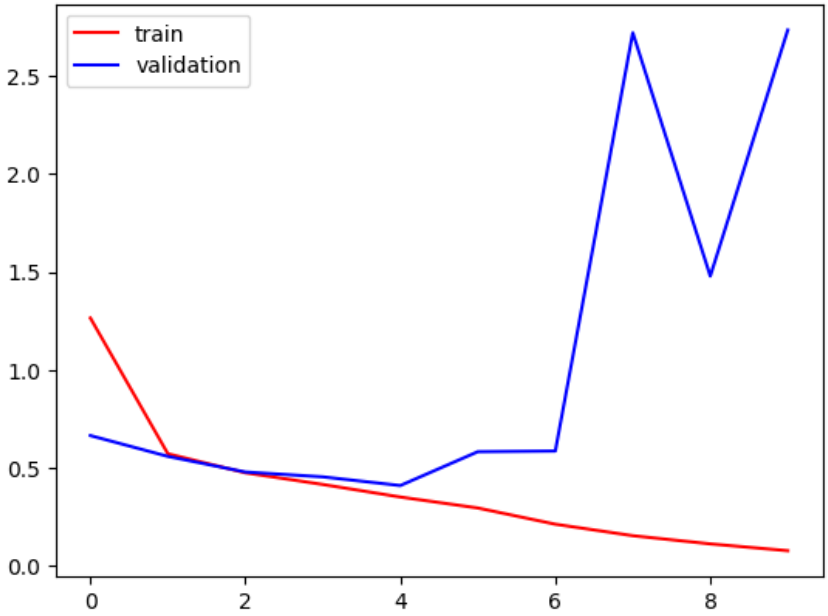

```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```




```
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()
```



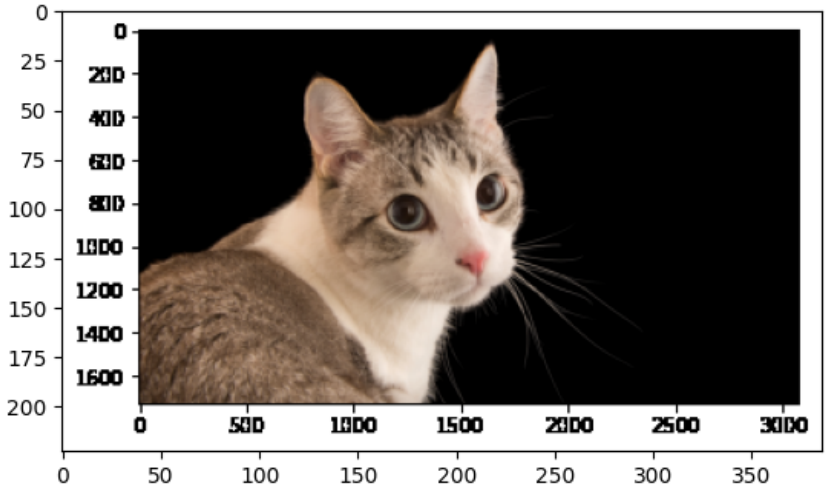

```
1 # ways to reduce overfitting
2
3 # Add more data
4 # Data Augmentation -> next video
5 # L1/L2 Regularizer
6 # Dropout
7 # Batch Norm
8 # Reduce complexity
```

```
1   import cv2
```

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

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

```
<matplotlib.image.AxesImage at 0x7de22a38a4a0>
```




```
1 test_img.shape
```

```
(223, 386, 3)
```

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

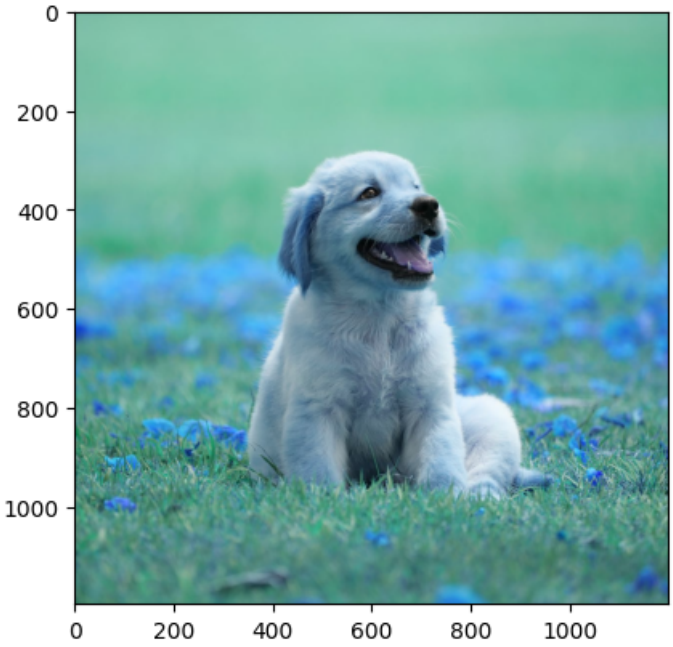
```
test_input = test_img.reshape((1,256,256,3))
```

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

```
1/1 [==============================] - 0s 388ms/step
array([[0.]], dtype=float32)
```

```
test_img2 = cv2.imread('/content/dog.jpg')
plt.imshow(test_img2)
```

```
<matplotlib.image.AxesImage at 0x7de1f7fd8d90>
```




```
test_img2 = cv2.resize(test_img2,(256,256))
```

```
test_input2 = test_img2.reshape((1,256,256,3))
```

```
model.predict(test_input2)
```

```
1/1 [==============================] - 0s 26ms/step
array([[1.]], dtype=float32)
```