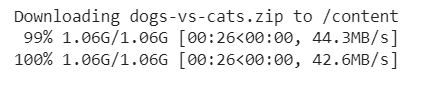
PROGRAM

!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/

!kaggle datasets download -d salader/dogs-vs-cats



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',

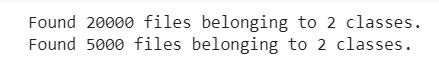
    labels='inferred',

    label\_mode = 'int',

    batch\_size=32,

    image\_size=(256,256)

)



# 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\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_4 (Conv2D) (None, 254, 254, 32) 896

batch\_normalization (BatchN (None, 254, 254, 32) 128

ormalization)

max\_pooling2d\_3 (MaxPooling (None, 127, 127, 32) 0

2D)

conv2d\_5 (Conv2D) (None, 125, 125, 64) 18496

batch\_normalization\_1 (Batc (None, 125, 125, 64) 256

hNormalization)

max\_pooling2d\_4 (MaxPooling (None, 62, 62, 64) 0

2D)

conv2d\_6 (Conv2D) (None, 60, 60, 128) 73856

batch\_normalization\_2 (BatchNormalization} (None, 60, 60, 128) 512

max\_pooling2d\_5 (MaxPooling (None, 30, 30, 128) 0

2D)

flatten\_1 (Flatten) (None, 115200) 0

dense\_3 (Dense) (None, 128) 14745728

dropout (Dropout) (None, 128) 0

dense\_4 (Dense) (None, 64) 8256

dropout\_1 (Dropout) (None, 64) 0

dense\_5 (Dense) (None, 1) 65

=================================================================

Total params: 14,848,193

Trainable params: 14,847,745

Non-trainable params: 448

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 [==============================] - 71s 112ms/step - loss: 1.2870 - accuracy: 0.6077 - val\_loss: 0.6443 - val\_accuracy: 0.6386

Epoch 2/10

625/625 [==============================] - 70s 112ms/step - loss: 0.5226 - accuracy: 0.7400 - val\_loss: 0.5353 - val\_accuracy: 0.7442

Epoch 3/10

625/625 [==============================] - 71s 113ms/step - loss: 0.4516 - accuracy: 0.7917 - val\_loss: 0.4704 - val\_accuracy: 0.7834

Epoch 4/10

625/625 [==============================] - 70s 111ms/step - loss: 0.3935 - accuracy: 0.8224 - val\_loss: 0.7385 - val\_accuracy: 0.6624

Epoch 5/10

625/625 [==============================] - 70s 111ms/step - loss: 0.3376 - accuracy: 0.8543 - val\_loss: 0.5054 - val\_accuracy: 0.7704

Epoch 6/10

625/625 [==============================] - 70s 112ms/step - loss: 0.2517 - accuracy: 0.8946 - val\_loss: 0.8225 - val\_accuracy: 0.7428

Epoch 7/10

625/625 [==============================] - 70s 112ms/step - loss: 0.1735 - accuracy: 0.9294 - val\_loss: 0.5472 - val\_accuracy: 0.7982

Epoch 8/10

625/625 [==============================] - 70s 111ms/step - loss: 0.1200 - accuracy: 0.9551 - val\_loss: 0.6331 - val\_accuracy: 0.8042

Epoch 9/10

625/625 [==============================] - 70s 111ms/step - loss: 0.0917 - accuracy: 0.9671 - val\_loss: 0.6862 - val\_accuracy: 0.8010

Epoch 10/10

625/625 [==============================] - 70s 111ms/step - loss: 0.0755 - accuracy: 0.9744 - val\_loss: 0.7166 - val\_accuracy: 0.8080

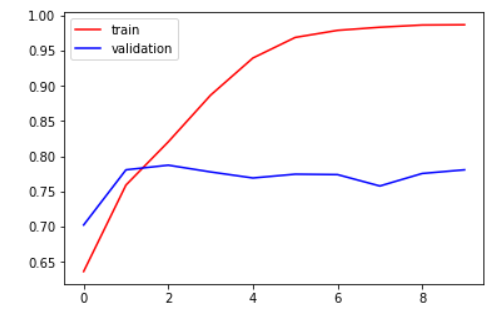
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'],color='red',label='train')

plt.plot(history.history['val\_accuracy'],color='blue',label='validation')

plt.legend()

plt.show()

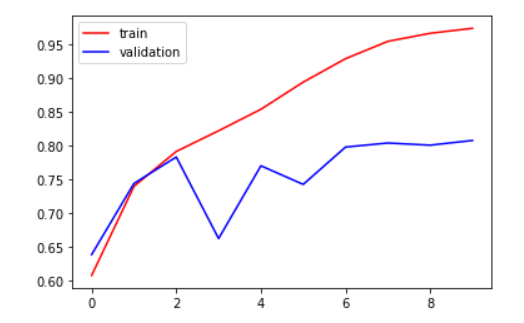


plt.plot(history.history['accuracy'],color='red',label='train')

plt.plot(history.history['val\_accuracy'],color='blue',label='validation')

plt.legend()

plt.show()

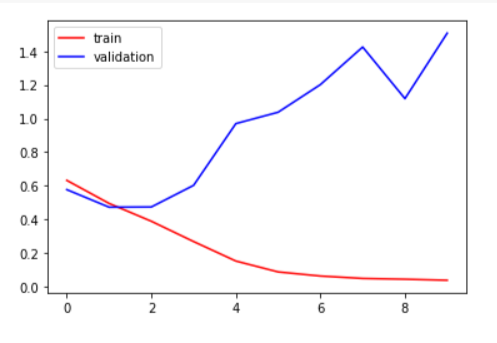


plt.plot(history.history['loss'],color='red',label='train')

plt.plot(history.history['val\_loss'],color='blue',label='validation')

plt.legend()

plt.show()

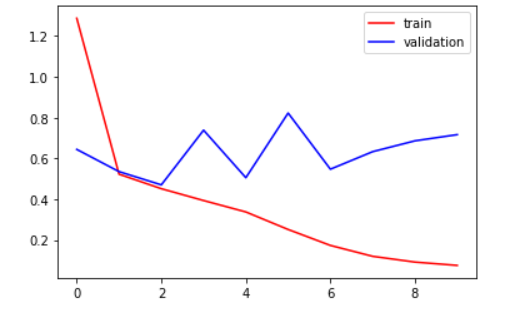


plt.plot(history.history['loss'],color='red',label='train')

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

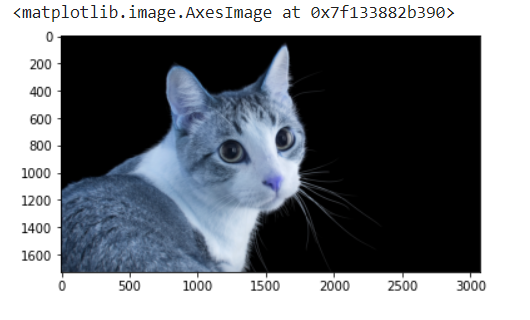
# Batch Norm

# Reduce complexity

import cv2

test\_img = cv2.imread('/content/cat.jpg')

plt.imshow(test\_img)



test\_img.shape

(1728, 3072, 3)

test\_img = cv2.resize(test\_img,(256,256))

test\_input = test\_img.reshape((1,256,256,3))

model.predict(test\_input)

array([[0.]], dtype=float32)

# Results

Include any relevant information about the model's performance, accuracy, and evaluation metrics.