

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

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

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


Downloading dogs-vs-cats.zip to /content
99% 1.06G/1.06G [00:26<00:00, 44.3MB/s]
100% 1.06G/1.06G [00:26<00:00, 42.6MB/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_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (Batch Normalization)	(None, 254, 254, 32)	128
max_pooling2d_3 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_5 (Conv2D)	(None, 125, 125, 64)	18496
batch_normalization_1 (Batch Normalization)	(None, 125, 125, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_6 (Conv2D)	(None, 60, 60, 128)	73856
batch_normalization_2 (Batch Normalization)	(None, 60, 60, 128)	512
max_pooling2d_5 (MaxPooling2D)	(None, 30, 30, 128)	0
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

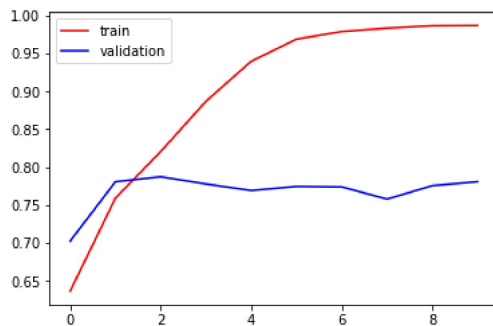
```

```
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()

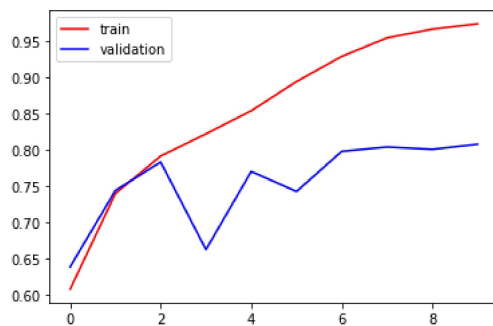
```



```

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

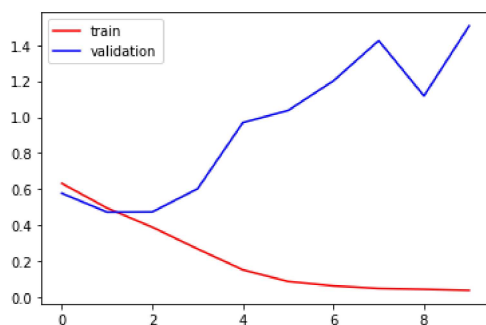
```



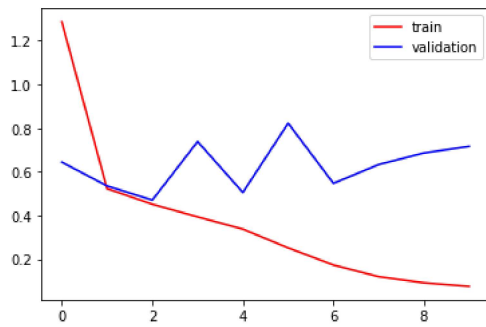
```

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()
```



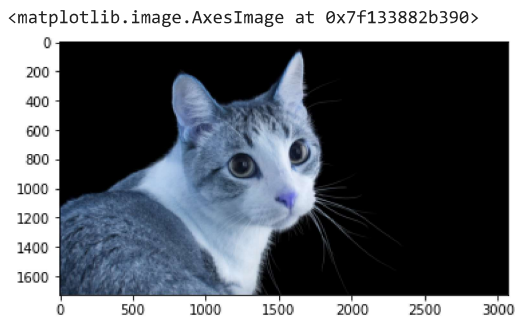
```
# ways to reduce overfitting
```

```
# 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)
```