```
!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,
Downloading dogs-vs-cats.zip to /content
100% 1.06G/1.06G [00:05<00:00, 209MB/s]
100% 1.06G/1.06G [00:05<00:00, 212MB/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)
```

```
#CNN Model
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), padding='valid', activation = 'relu', input_shape=
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_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 254, 254, 32)      896

 batch_normalization_3 (Batc  (None, 254, 254, 32)     128
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_4 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_4 (Batc  (None, 125, 125, 64)     256
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 60, 60, 128)       73856

 batch_normalization_5 (Batc  (None, 60, 60, 128)      512
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 115200)            0

 dense_1 (Dense)             (None, 128)               14745728
```

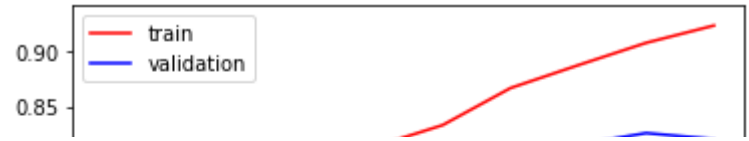
```
 dropout (Dropout)           (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (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)
```

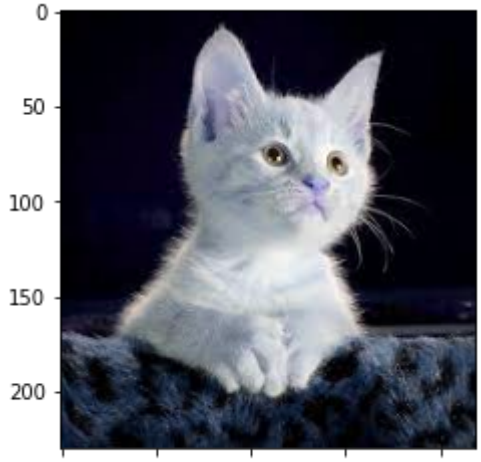
```
Epoch 1/10
625/625 [==============================] - 82s 117ms/step - loss: 1.4704 - accuracy:
Epoch 2/10
625/625 [==============================] - 73s 116ms/step - loss: 0.5806 - accuracy:
Epoch 3/10
625/625 [==============================] - 73s 116ms/step - loss: 0.5288 - accuracy:
Epoch 4/10
625/625 [==============================] - 75s 119ms/step - loss: 0.4494 - accuracy:
Epoch 5/10
625/625 [==============================] - 73s 116ms/step - loss: 0.3960 - accuracy:
Epoch 6/10
625/625 [==============================] - 73s 116ms/step - loss: 0.3592 - accuracy:
Epoch 7/10
625/625 [==============================] - 78s 123ms/step - loss: 0.2893 - accuracy:
Epoch 8/10
625/625 [==============================] - 75s 120ms/step - loss: 0.2406 - accuracy:
Epoch 9/10
625/625 [==============================] - 73s 116ms/step - loss: 0.1995 - accuracy:
Epoch 10/10
625/625 [==============================] - 72s 115ms/step - loss: 0.1638 - accuracy:
```

```
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()
```




```
import cv2
test1 = cv2.imread('/content/cat.jpeg')
test2 = cv2.imread('/content/dog.jpeg')
plt.imshow(test1)
```

```
<matplotlib.image.AxesImage at 0x7f10b1283710>
```




```
plt.imshow(test2)
```

```
<matplotlib.image.AxesImage at 0x7f10b0134a90>
```


```
test1.shape
```

```
(230, 219, 3)
```

```
test1 = cv2.resize(test1, (256,256))
```

```
test_input = test1.reshape((1, 256, 256, 3))
```

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

```
1/1 [==============================] - 0s 248ms/step
array([[0.]], dtype=float32)
```

```
test2.shape
```

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

```
test2 = cv2.resize(test2, (256,256))
test_input2 = test2.reshape((1,256,256,3))
```

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

```
1/1 [==============================] - 0s 32ms/step
array([[1.]], dtype=float32)
```