```
!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
Dataset URL: https://www.kaggle.com/datasets/salader/dogs-vs-cats
License(s): unknown
Downloading dogs-vs-cats.zip to /content
 96% 1.02G/1.06G [00:07<00:00, 182MB/s]
100% 1.06G/1.06G [00:07<00:00, 147MB/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
```

```
#generator(useful for large amount of data)
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
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'))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (BatchNormalization)	(None, 254, 254, 32)	128
max_pooling2d_3 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_4 (Conv2D)	(None, 125, 125, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 125, 125, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_5 (Conv2D)	(None, 60, 60, 128)	73,856
batch_normalization_2 (BatchNormalization)	(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)	14,745,728
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 1)	65

Total params: 14,848,193 (56.64 MB)

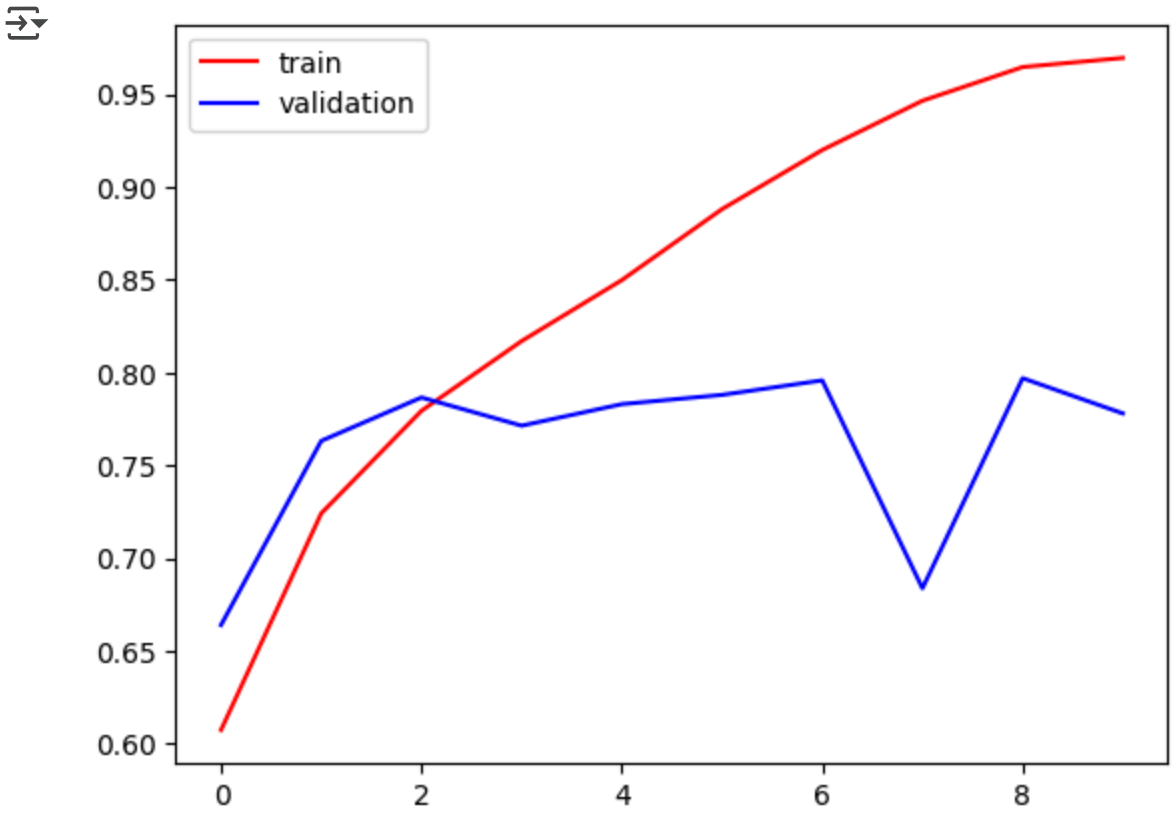
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

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

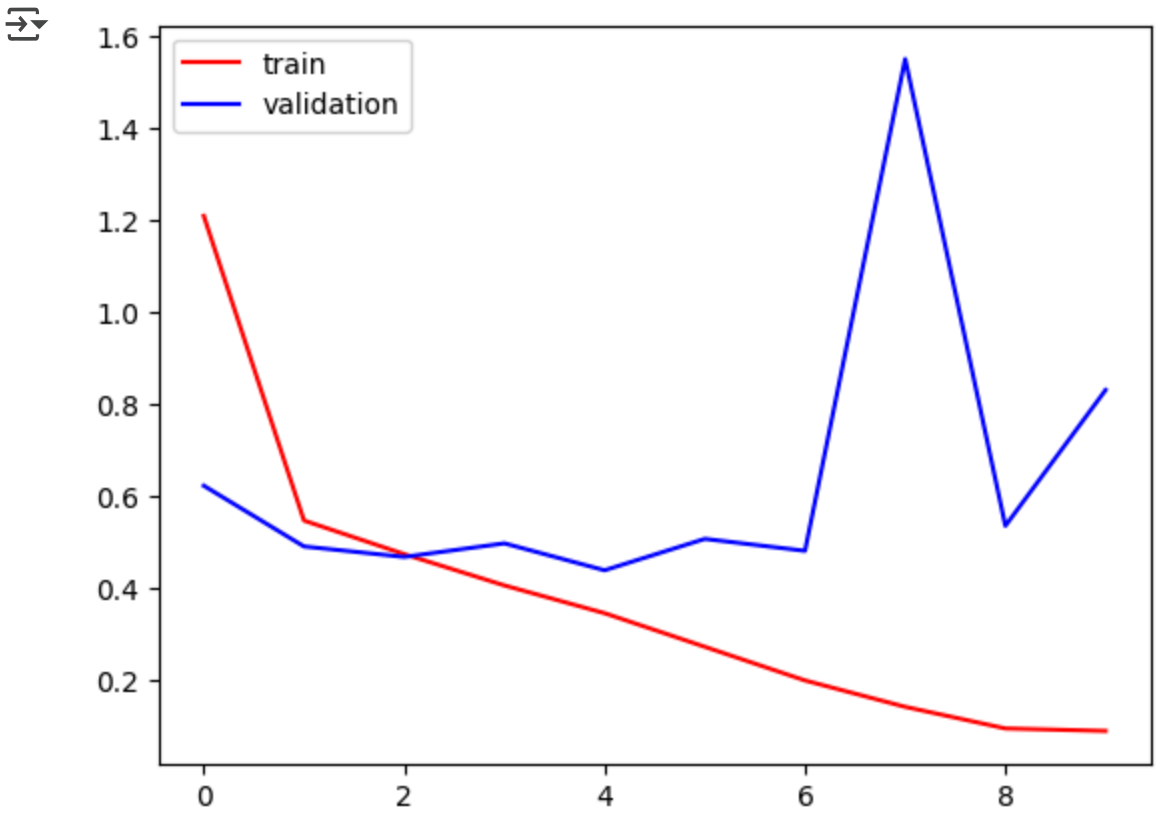
```
Epoch 1/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 60s 86ms/step - accuracy: 0.5756 - loss: 2.1789 - val_accur
Epoch 2/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.7019 - loss: 0.5758 - val_accur
Epoch 3/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.7690 - loss: 0.4888 - val_accur
Epoch 4/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 86s 90ms/step - accuracy: 0.8092 - loss: 0.4218 - val_accur
Epoch 5/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 84ms/step - accuracy: 0.8408 - loss: 0.3617 - val_accur
Epoch 6/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.8784 - loss: 0.2964 - val_accur
```

```
Epoch 7/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 83ms/step - accuracy: 0.9089 - loss: 0.2221 - val_accur
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 82ms/step - accuracy: 0.9420 - loss: 0.1520 - val_accur
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 56s 90ms/step - accuracy: 0.9620 - loss: 0.1026 - val_accur
Epoch 10/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.9666 - loss: 0.0931 - val_accur
```

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




```
# way to reduce overfitting

# Add more data
# L1/L2 Regulization
# Dropout
# Batch Normalization
# Reduce complexity


import cv2


# test_img = cv2.imread('/content/Dog.jpeg')


test_img = cv2.imread('/content/cat.jpeg')


plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x79cd7666b3b0>
```


```
test_img.shape
```

```
(183, 276, 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 29ms/step
array([[1.4531483e-32]], dtype=float32)
```

Start coding or generate with AI.