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

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

```
Dataset URL: https://www.kaggle.com/datasets/salader/dogs-vs-cats
License(s): unknown
Downloading dogs-vs-cats.zip to /content
100% 1.06G/1.06G [00:54<00:00, 22.1MB/s]
100% 1.06G/1.06G [00:54<00:00, 20.9MB/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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
# 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)
# )
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

train_ds = train_datagen.flow_from_directory(
    directory='/content/train',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary'
)

validation_ds = validation_datagen.flow_from_directory(
    directory='/content/test',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 20000 images belonging to 2 classes.
Found 5000 images 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)


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()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_1"

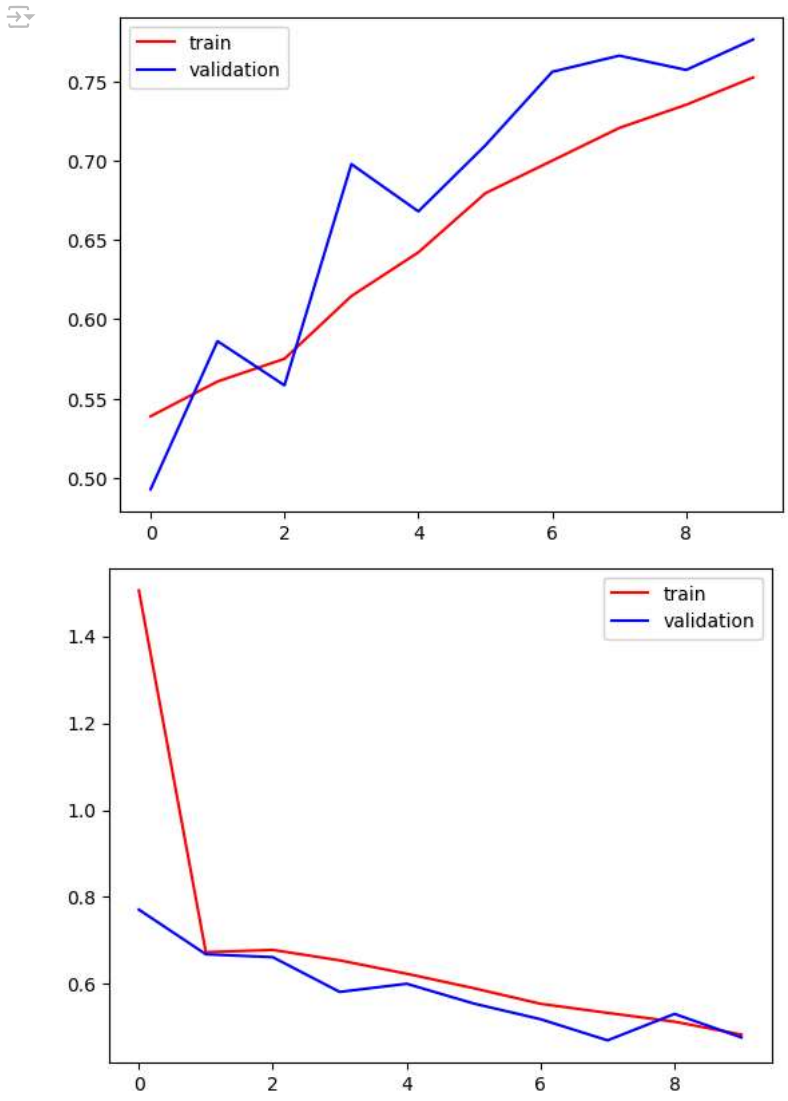
Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 254, 254, 32)	896
batch_normalization_3 (BatchNormalization)	(None, 254, 254, 32)	128
max_pooling2d_3 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_4 (Conv2D)	(None, 125, 125, 64)	18,496
batch_normalization_4 (BatchNormalization)	(None, 125, 125, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_5 (Conv2D)	(None, 60, 60, 128)	73,856
batch_normalization_5 (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)

```
import matplotlib.pyplot as plt

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

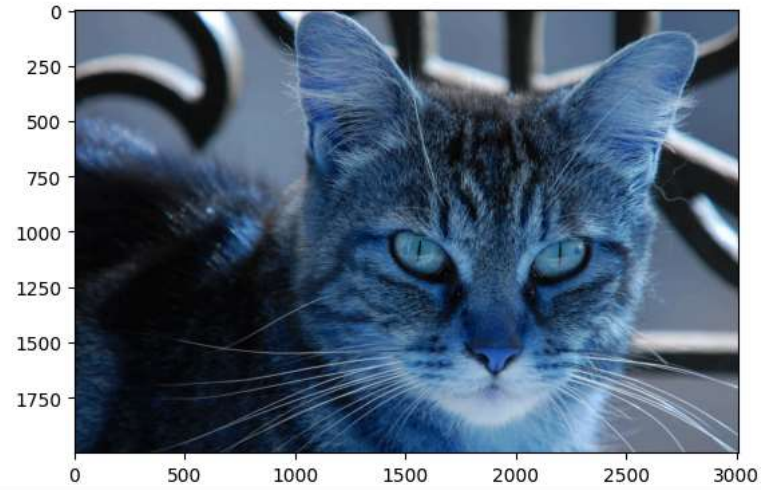
```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
plt.plot(history.history['loss'],color='red',label='train')
plt.plot(history.history['val_loss'],color='blue',label='validation')
plt.legend()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




```
import cv2
test_img = cv2.imread('/content/cat.jpeg')
plt.imshow(test_img)

test_img.shape
```

(2000, 3008, 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 33ms/step
array([[0.00327671]], dtype=float32)
```

```
test_img.shape
```

(256, 256, 3)

Start coding or generate with AI.