

Cat and Dog Classification



VS



✓ cats vs dogs classifier

```
!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 can run 'chmod 600 /root/.kaggle/kaggle.json'
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 [00:10<00:00, 223MB/s]
100% 1.06G/1.06G [00:10<00:00, 111MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip')
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 - create batches
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,lable):
    image=tf.cast(image/255. ,tf.float32)
    return image,lable

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'))
```

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

```
model.summary()
```

⚡ Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (BatchNormalization)	(None, 254, 254, 32)	128
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 125, 125, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 60, 60, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 128)	14,745,728
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

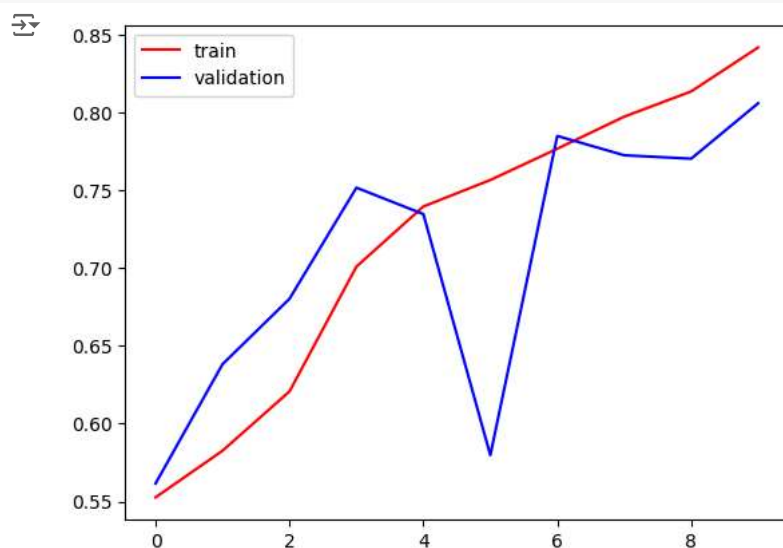
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

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

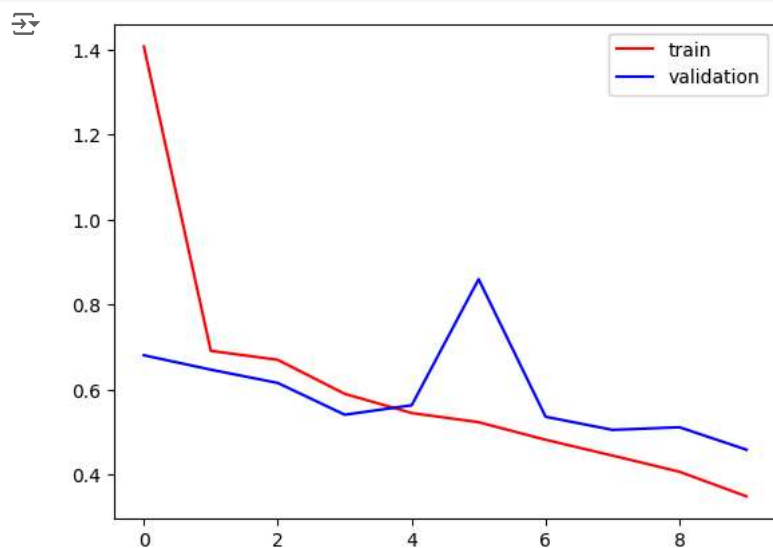
```
⚡ Epoch 1/10
625/625 ————— 83s 108ms/step - accuracy: 0.5519 - loss: 2.6128 - val_accuracy: 0.5616 - val_loss: 0.6810
Epoch 2/10
625/625 ————— 75s 120ms/step - accuracy: 0.5783 - loss: 0.6873 - val_accuracy: 0.6382 - val_loss: 0.6471
Epoch 3/10
625/625 ————— 60s 85ms/step - accuracy: 0.6118 - loss: 0.6658 - val_accuracy: 0.6802 - val_loss: 0.6159
Epoch 4/10
625/625 ————— 84s 88ms/step - accuracy: 0.6842 - loss: 0.6050 - val_accuracy: 0.7516 - val_loss: 0.5412
Epoch 5/10
625/625 ————— 56s 90ms/step - accuracy: 0.7332 - loss: 0.5594 - val_accuracy: 0.7346 - val_loss: 0.5634
```

```
Epoch 6/10
625/625 ————— 63s 101ms/step - accuracy: 0.7462 - loss: 0.5387 - val_accuracy: 0.5798 - val_loss: 0.8599
Epoch 7/10
625/625 ————— 84s 104ms/step - accuracy: 0.7657 - loss: 0.4995 - val_accuracy: 0.7848 - val_loss: 0.5364
Epoch 8/10
625/625 ————— 95s 124ms/step - accuracy: 0.7926 - loss: 0.4542 - val_accuracy: 0.7724 - val_loss: 0.5056
Epoch 9/10
625/625 ————— 62s 98ms/step - accuracy: 0.8055 - loss: 0.4253 - val_accuracy: 0.7702 - val_loss: 0.5117
Epoch 10/10
625/625 ————— 77s 90ms/step - accuracy: 0.8336 - loss: 0.3699 - val_accuracy: 0.8058 - val_loss: 0.4590
```

```
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()
```



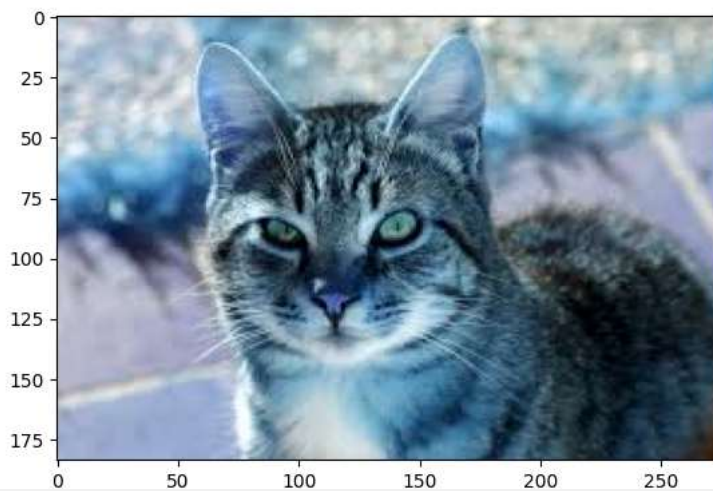
```
#need to reduce complexity
# add more data
#reducing complexity
```

```
import cv2
```


```
test_image = cv2.imread('/content/cat.jpeg')
```

```
plt.imshow(test_image)
```

 <matplotlib.image.AxesImage at 0x7afbc6a190f0>



```
test_image.shape
```

 (2500, 2392, 3)

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

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

For Cat Output Is 0

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

 1/1 ————— 0s 19ms/step
array([[0.]], dtype=float32)

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

```
test_image = cv2.imread('/content/dog.jpeg')
```

```
plt.imshow(test_image)
```

 <matplotlib.image.AxesImage at 0x7afbc6c56890>





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

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

For Dog Output Is 1.

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

 1/1  0s 20ms/step
array([[1.]], dtype=float32)