

```

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE=224
BATCH_SIZE=32

train_datagen=ImageDataGenerator(rescale=1./255,validation_split=0.2)

train_generator=train_datagen.flow_from_directory('/content/drive/
MyDrive/deep',

target_size=(IMG_SIZE,IMG_SIZE),

batch_size=BATCH_SIZE,
class_mode='binary',
subset='training')

Found 400 images belonging to 1 classes.

val_generator=train_datagen.flow_from_directory('/content/drive/
MyDrive/deep',

target_size=(IMG_SIZE,IMG_SIZE),

batch_size=BATCH_SIZE,
class_mode='binary',
subset='validation')

Found 100 images belonging to 1 classes.

model=keras.Sequential([layers.Conv2D(32,
(3,3),activation='relu',input_shape=(IMG_SIZE,IMG_SIZE,3)),
layers.MaxPooling2D((2,2)),
layers.Conv2D(64,(3,3),activation='relu'),
layers.MaxPooling2D((2,2)),
layers.Conv2D(128,(3,3),activation='relu'),
layers.MaxPooling2D((2,2)),
layers.Flatten(),
layers.Dense(128,activation='relu'),
layers.Dense(1,activation='sigmoid')])
model.summary()

/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

```

Model: "sequential"

Layer (type) Param #	Output Shape
conv2d (Conv2D) 896	(None, 222, 222, 32)
max_pooling2d (MaxPooling2D) 0	(None, 111, 111, 32)
conv2d_1 (Conv2D) 18,496	(None, 109, 109, 64)
max_pooling2d_1 (MaxPooling2D) 0	(None, 54, 54, 64)
conv2d_2 (Conv2D) 73,856	(None, 52, 52, 128)
max_pooling2d_2 (MaxPooling2D) 0	(None, 26, 26, 128)
flatten (Flatten) 0	(None, 86528)
dense (Dense) 11,075,712	(None, 128)
dense_1 (Dense) 129	(None, 1)

Total params: 11,169,089 (42.61 MB)

Trainable params: 11,169,089 (42.61 MB)

Non-trainable params: 0 (0.00 B)

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
model.fit(train_generator,epochs=2,validation_data=val_generator,batch_size=BATCH_SIZE)
```

Epoch 1/2

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/
data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
```

```
13/13 _____ 160s 8s/step - accuracy: 0.7649 - loss:
0.1819 - val_accuracy: 1.0000 - val_loss: 6.4323e-38
```

Epoch 2/2

```
13/13 _____ 104s 4s/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
```

```
<keras.src.callbacks.history.History at 0x7fca5d308890>
```

```
model.save('/content/drive/MyDrive/deep/DEEP.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

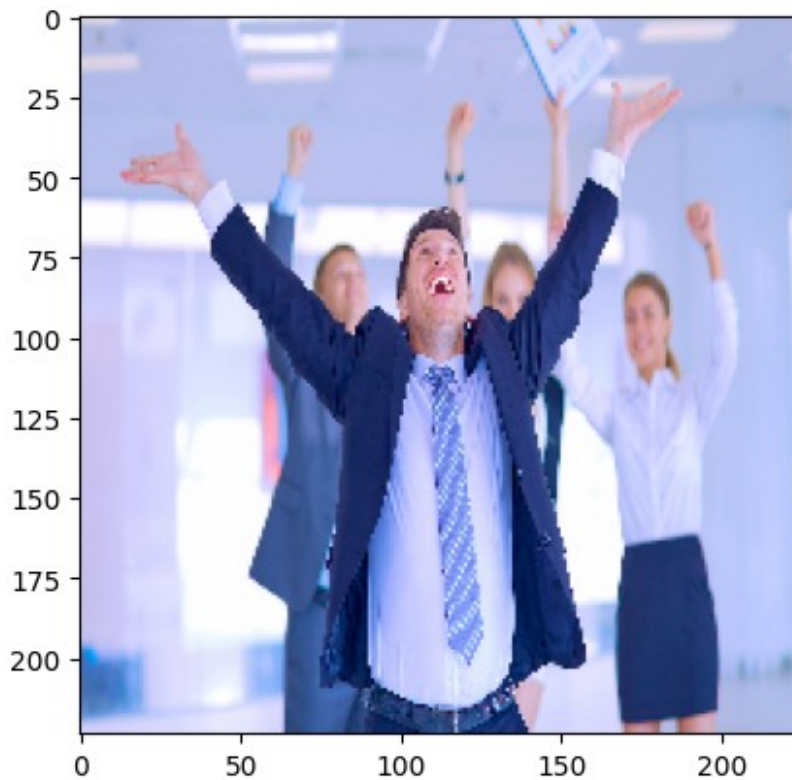
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
```

```
model=load_model('/content/drive/MyDrive/deep/DEEP.h5')
print("model loaded")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

model loaded

```
test='/content/drive/MyDrive/deep/prepole face dataset/happy/170404-
happy-workers-feature.jpg'
img=image.load_img(test,target_size=(224,224))
plt.imshow(img)
plt.axis()
plt.show()
```



```
img_array=image.img_to_array(img)
img_array = np.expand_dims(img_array,axis=0)
img_array/=255
```

```
prediction=model.predict(img_array)
print(prediction)
```

```
1/1 ————— 0s 133ms/step
[[0.]]
```

```
if prediction >0.5:
    print("sad")
else:
    print("happy")
```

```
happy
```