day1-028

June 25, 2024

[1]:
```
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
```

[2]:
```
train_datagen = ImageDataGenerator(rescale=1./255,validation_split=0.2)

train_generator = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/archive',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
    )

val_generator = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/archive',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
    )
```

```
Found 1713 images belonging to 2 classes.
Found 427 images belonging to 2 classes.
```

[3]:
```
#defin model

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(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

[6]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

[7]:
```
model.fit(train_generator,validation_data=val_generator,epochs=5)
```

```
Epoch 1/5
54/54 [==============================] - 270s 5s/step - loss: 0.5550 - accuracy:
0.7017 - val_loss: 0.6019 - val_accuracy: 0.6698
Epoch 2/5
54/54 [==============================] - 279s 5s/step - loss: 0.3901 - accuracy:
0.8284 - val_loss: 0.4034 - val_accuracy: 0.7588
Epoch 3/5
54/54 [==============================] - 269s 5s/step - loss: 0.2868 - accuracy:
0.8762 - val_loss: 0.4560 - val_accuracy: 0.7892
Epoch 4/5
54/54 [==============================] - 258s 5s/step - loss: 0.2100 - accuracy:
0.9241 - val_loss: 0.6277 - val_accuracy: 0.7354
Epoch 5/5
54/54 [==============================] - 265s 5s/step - loss: 0.1690 - accuracy:
0.9370 - val_loss: 0.5725 - val_accuracy: 0.8080
```

[7]:
```
<keras.src.callbacks.History at 0x7f59f1aefbb0>
```

[8]:
```
model.save("model.h5","label.txt")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

[20]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

model = load_model('/content/model.h5')

test_image_path = '/content/drive/MyDrive/archive/1/2301.png'

img =image.load_img(test_image_path,target_size=(224,224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array,axis=0)
```

```
img_array /=225.0

prediction = model.predict(img_array)

print(prediction)
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f59b2d56b00>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has reduce_retracing=True option that can avoid unnecessary
retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.

1/1 [==============================] - 0s 196ms/step
[[0.86097455]]
```

[21]:
```
if prediction <0.5:
  print(" infected:",prediction[0][0])
else:
  print(" not infected",prediction[0][0])
```

```
 not infected 0.86097455
```

[ ]:
```

```