task7-malariya

June 16, 2025

[1]:
```
import os

# Set your Kaggle credentials
os.environ['KAGGLE_USERNAME'] = "godwinsamraj"
os.environ['KAGGLE_KEY'] = "8e9616edcd9c72a9033c54d0982046a1"

# Download the dataset
!kaggle datasets download -d meetnagadia/malaria-dataset

# Unzip into folder
!unzip -q malaria-dataset.zip -d malaria_dataset
```

```
Dataset URL: https://www.kaggle.com/datasets/meetnagadia/malaria-dataset
License(s): DbCL-1.0
Downloading malaria-dataset.zip to /content
  0% 0.00/6.18M [00:00<?, ?B/s]
100% 6.18M/6.18M [00:00<00:00, 753MB/s]
```

[3]:
```
!ls
```

```
malaria_dataset  malaria-dataset.zip  sample_data
```

[4]:
```
!ls malaria_dataset
```

```
Dataset
```

[5]:
```
base_path = "malaria_dataset/Dataset"
```

[6]:
```
import os

classes = [folder for folder in os.listdir(base_path) if os.path.isdir(os.path.
 ↪join(base_path, folder))]
print("Classes found in dataset:", classes)
```

```
Classes found in dataset: ['Test', 'Train']
```

[7]:
```
train_dir = "malaria_dataset/Dataset/Train"
test_dir = "malaria_dataset/Dataset/Test"
```

[8]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255)

train_data = datagen.flow_from_directory(
    train_dir,
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical'  # because we have Parasitized & Uninfected
)

val_data = datagen.flow_from_directory(
    test_dir,
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 416 images belonging to 2 classes.
Found 134 images belonging to 2 classes.
```

[9]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
 ↪Dropout

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(64,64,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(train_data.num_classes, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
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)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1,605,760
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 2)	258

Total params: 1,625,410 (6.20 MB)

Trainable params: 1,625,410 (6.20 MB)

Non-trainable params: 0 (0.00 B)

[10]:
```
model.fit(train_data, validation_data=val_data, epochs=10)
```

```
Epoch 1/10
/usr/local/lib/python3.11/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
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            0s 117ms/step -
accuracy: 0.4871 - loss: 1.0279
/usr/local/lib/python3.11/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
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             4s 192ms/step -
accuracy: 0.4877 - loss: 1.0190 - val_accuracy: 0.6493 - val_loss: 0.6831
Epoch 2/10
13/13             4s 126ms/step -
accuracy: 0.5743 - loss: 0.6846 - val_accuracy: 0.5821 - val_loss: 0.6859
Epoch 3/10
13/13             2s 126ms/step -
accuracy: 0.7086 - loss: 0.6446 - val_accuracy: 0.6791 - val_loss: 0.6432
Epoch 4/10
13/13             2s 124ms/step -
accuracy: 0.6154 - loss: 0.6380 - val_accuracy: 0.6716 - val_loss: 0.6355
Epoch 5/10
13/13             2s 123ms/step -
accuracy: 0.7113 - loss: 0.5582 - val_accuracy: 0.5522 - val_loss: 0.6900
Epoch 6/10
13/13             2s 158ms/step -
accuracy: 0.7222 - loss: 0.5254 - val_accuracy: 0.4552 - val_loss: 0.7747
Epoch 7/10
13/13             2s 125ms/step -
accuracy: 0.8059 - loss: 0.4433 - val_accuracy: 0.5448 - val_loss: 0.7339
Epoch 8/10
13/13             2s 123ms/step -
accuracy: 0.8653 - loss: 0.3363 - val_accuracy: 0.6343 - val_loss: 0.8046
Epoch 9/10
13/13             2s 121ms/step -
accuracy: 0.8070 - loss: 0.4032 - val_accuracy: 0.5224 - val_loss: 0.7706
Epoch 10/10
13/13             2s 123ms/step -
accuracy: 0.9115 - loss: 0.2715 - val_accuracy: 0.6119 - val_loss: 0.6868
```

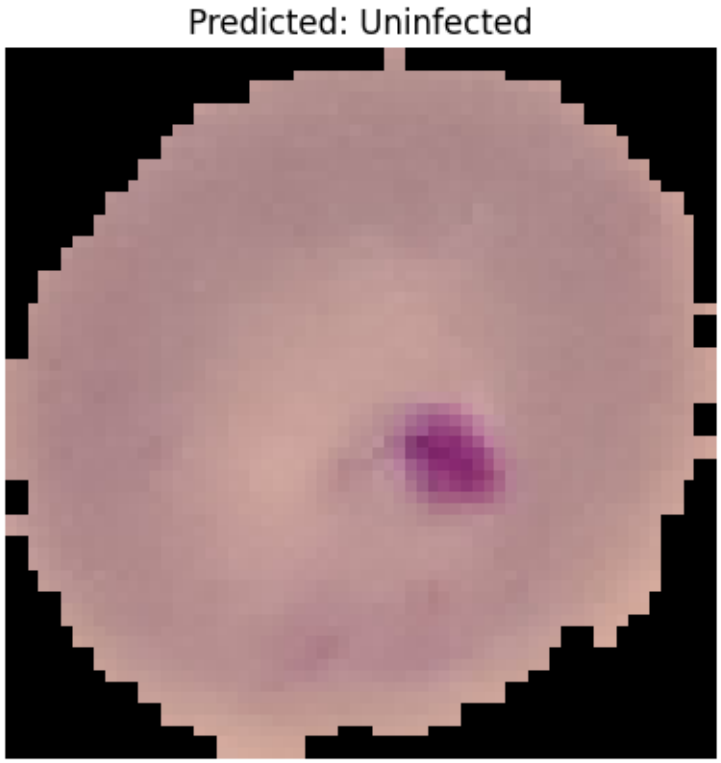
[10]:
```
<keras.src.callbacks.history.History at 0x7a517ff2bc50>
```

[14]:
```
img_path = "malaria_dataset/Dataset/Test/Parasite/
 ↪C39P4thinF_original_IMG_20150622_105554_cell_10.png"
```

[15]:
```
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt

#  Correct image path
img_path = "malaria_dataset/Dataset/Test/Parasite/
 ↪C39P4thinF_original_IMG_20150622_105554_cell_10.png"

# Load and preprocess
```

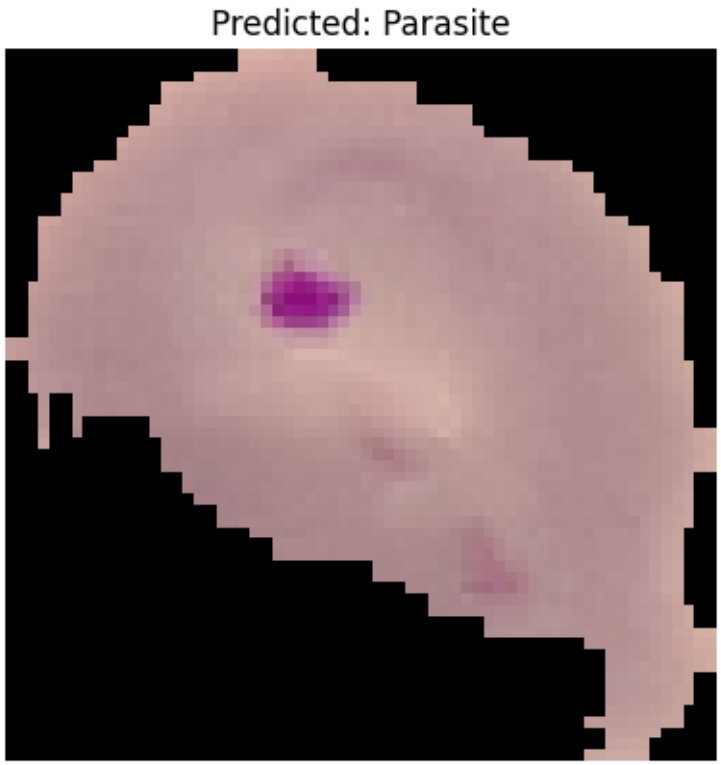
```
img = image.load_img(img_path, target_size=(64, 64))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.

# Predict
prediction = model.predict(img_array)
predicted_class = np.argmax(prediction)

# Class labels
class_labels = list(train_data.class_indices.keys())
print(f"Predicted Class: {class_labels[predicted_class]}")

# Display image
plt.imshow(img)
plt.title(f"Predicted: {class_labels[predicted_class]}")
plt.axis('off')
plt.show()
```

```
1/1             0s 76ms/step
Predicted Class: Uninfected
```



[16]:
```
import os
import random
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt

# Define test folders
test_dir = "malaria_dataset/Dataset/Test"
categories = ["Parasite", "Uninfected"]

#  Randomly choose a category and an image from it
chosen_category = random.choice(categories)
category_path = os.path.join(test_dir, chosen_category)
chosen_image = random.choice(os.listdir(category_path))
img_path = os.path.join(category_path, chosen_image)

print(f" Selected Image: {img_path}")

# Load and preprocess
img = image.load_img(img_path, target_size=(64, 64))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.

# Predict
prediction = model.predict(img_array)
predicted_class = np.argmax(prediction)

# Class labels
class_labels = list(train_data.class_indices.keys())
print(f" Predicted Class: {class_labels[predicted_class]}")

# Display image
plt.imshow(img)
plt.title(f"Predicted: {class_labels[predicted_class]}")
plt.axis('off')
plt.show()
```

```
 Selected Image: malaria_dataset/Dataset/Test/Parasite/C39P4thinF_original_IMG_
20150622_105803_cell_78.png
1/1             0s 28ms/step
 Predicted Class: Parasite
```


Predicted: Parasite