projectml-2024

November 17, 2024

[1]:
```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import PIL
import matplotlib.pyplot as plt
```

[2]:
```
IMAGE_SHAPE = (250, 250)
batch_size = 30
image_generator = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)
train_d = 'train'
training_image_data = image_generator.flow_from_directory(train_d,␣
 ↪target_size=IMAGE_SHAPE)

test_d = 'test'
testing_image_data = image_generator.flow_from_directory(test_d,␣
 ↪target_size=IMAGE_SHAPE)
```

```
Found 1156 images belonging to 9 classes.
Found 502 images belonging to 9 classes.
```

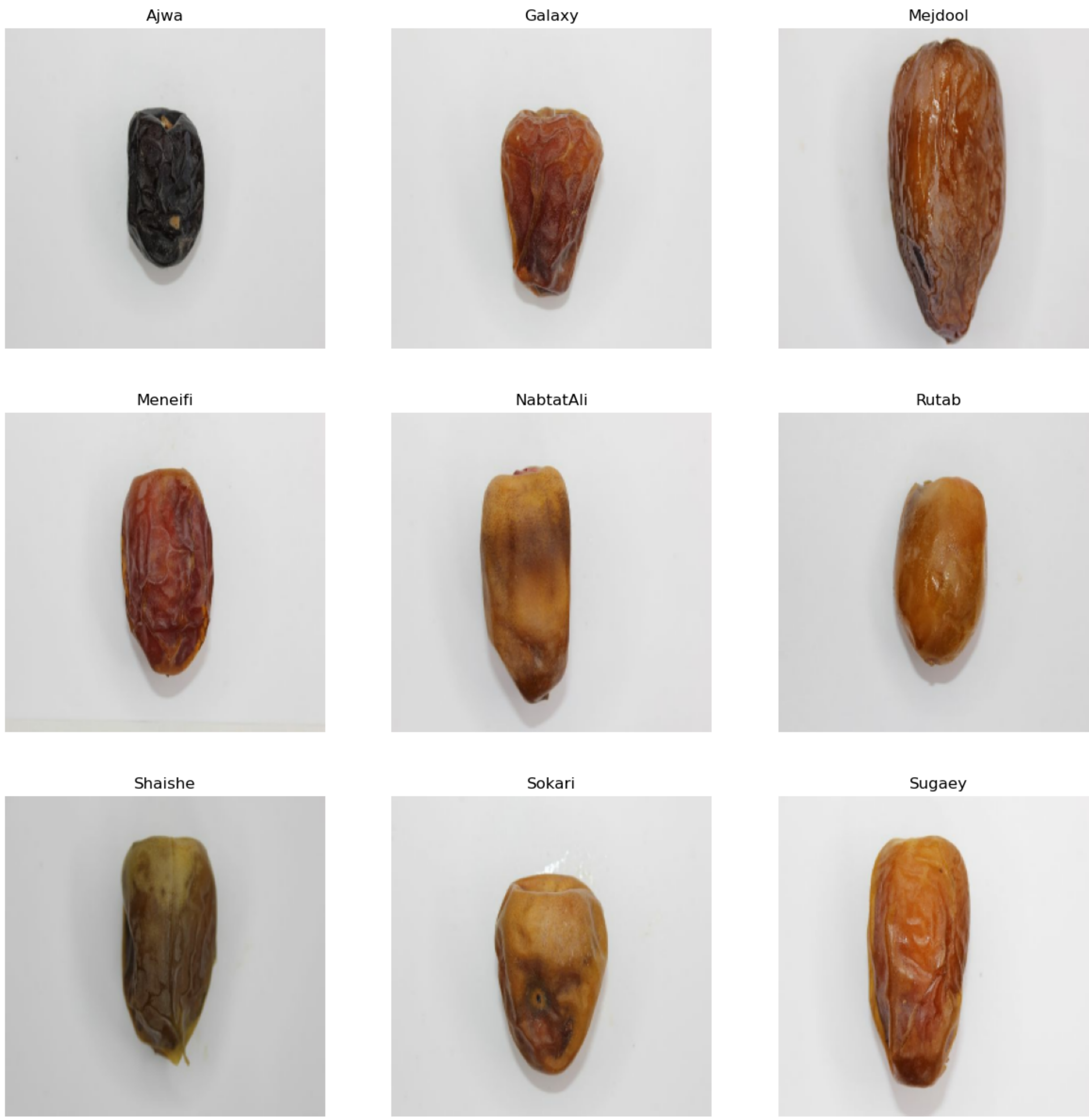
[3]:
```
train_ds = tf.keras.utils.image_dataset_from_directory(
    train_d,
    image_size=IMAGE_SHAPE,
    batch_size=batch_size
)

test_ds = tf.keras.utils.image_dataset_from_directory(
    test_d,
    image_size=IMAGE_SHAPE,
    batch_size=batch_size
)
```

```
Found 1156 files belonging to 9 classes.
Found 502 files belonging to 9 classes.
```

[4]:

```python
class_names = train_ds.class_names
print("Class Names:", class_names)


images_per_class = {}
for images, labels in train_ds:
    for image, label in zip(images, labels):
        class_name = class_names[label.numpy()]
        if class_name not in images_per_class:
            images_per_class[class_name] = image

        if len(images_per_class) == len(class_names):
            break
    if len(images_per_class) == len(class_names):
        break


plt.figure(figsize=(15, 15))
for i, class_name in enumerate(class_names):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images_per_class[class_name].numpy().astype("uint8"))
    plt.title(class_name)
    plt.axis("off")

plt.show()

AUTOTUNE = tf.data.AUTOTUNE
```

```
Class Names: ['Ajwa', 'Galaxy', 'Mejdool', 'Meneifi', 'NabtatAli', 'Rutab',
'Shaishe', 'Sokari', 'Sugaey']
```



[5]:
```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

normalization_layer = layers.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
```

[6]:
```
num_classes = len(class_names)

model = Sequential([
```

```
    layers.Rescaling(1./255, input_shape=(IMAGE_SHAPE[0], IMAGE_SHAPE[1], 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])



model.compile(optimizer='adam', loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 250, 250, 3)       0

 conv2d (Conv2D)             (None, 250, 250, 16)      448

 max_pooling2d (MaxPooling2D  (None, 125, 125, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 62, 62, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 31, 31, 64)       0
 2D)

 flatten (Flatten)           (None, 61504)             0

 dense (Dense)               (None, 128)               7872640

 dense_1 (Dense)             (None, 9)                 1161
```

```
=================================================================
Total params: 7,897,385
Trainable params: 7,897,385
Non-trainable params: 0
_________________________________________________________________
```

[7]:

```
epochs = 17
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs
)
```

```
Epoch 1/17
39/39 [==============================] - 51s 1s/step - loss: 1.9212 - accuracy:
0.3322 - val_loss: 1.3392 - val_accuracy: 0.5558
Epoch 2/17
39/39 [==============================] - 92s 2s/step - loss: 0.8878 - accuracy:
0.6877 - val_loss: 0.7521 - val_accuracy: 0.7191
Epoch 3/17
39/39 [==============================] - 90s 2s/step - loss: 0.5083 - accuracy:
0.8452 - val_loss: 0.4704 - val_accuracy: 0.8247
Epoch 4/17
39/39 [==============================] - 83s 2s/step - loss: 0.3452 - accuracy:
0.8884 - val_loss: 0.4857 - val_accuracy: 0.8207
Epoch 5/17
39/39 [==============================] - 85s 2s/step - loss: 0.3853 - accuracy:
0.8668 - val_loss: 0.4051 - val_accuracy: 0.8586
Epoch 6/17
39/39 [==============================] - 86s 2s/step - loss: 0.2918 - accuracy:
0.9005 - val_loss: 0.4147 - val_accuracy: 0.8486
Epoch 7/17
39/39 [==============================] - 90s 2s/step - loss: 0.2960 - accuracy:
0.9005 - val_loss: 0.4009 - val_accuracy: 0.8526
Epoch 8/17
39/39 [==============================] - 86s 2s/step - loss: 0.1881 - accuracy:
0.9412 - val_loss: 0.3418 - val_accuracy: 0.8785
Epoch 9/17
39/39 [==============================] - 88s 2s/step - loss: 0.1121 - accuracy:
0.9732 - val_loss: 0.4792 - val_accuracy: 0.8586
Epoch 10/17
39/39 [==============================] - 91s 2s/step - loss: 0.1447 - accuracy:
0.9542 - val_loss: 0.2771 - val_accuracy: 0.8944
Epoch 11/17
39/39 [==============================] - 93s 2s/step - loss: 0.0807 - accuracy:
0.9766 - val_loss: 0.3104 - val_accuracy: 0.8944
Epoch 12/17
39/39 [==============================] - 87s 2s/step - loss: 0.1016 - accuracy:
```

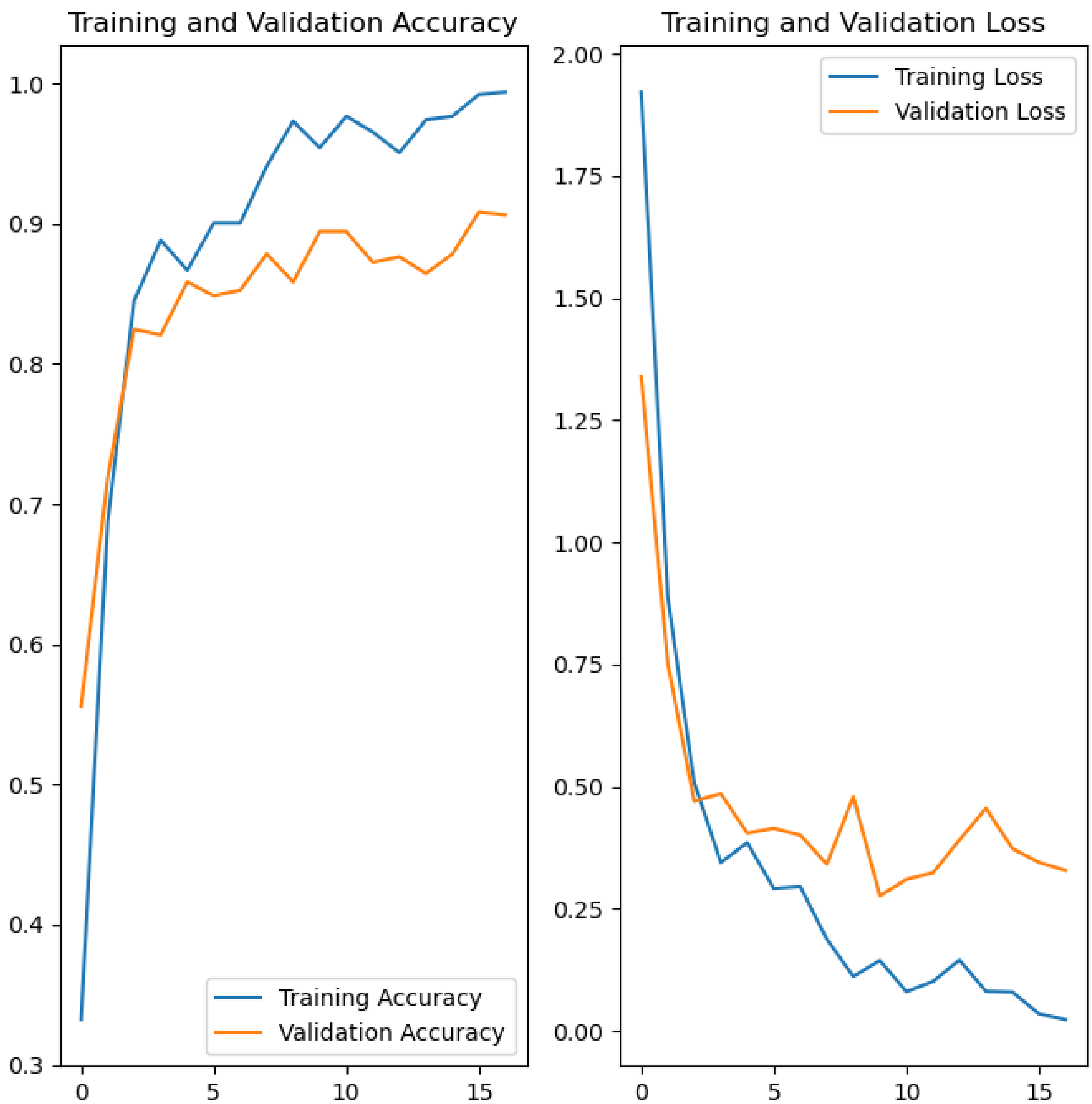
```
0.9654 - val_loss: 0.3241 - val_accuracy: 0.8725
Epoch 13/17
39/39 [==============================] - 89s 2s/step - loss: 0.1455 - accuracy:
0.9507 - val_loss: 0.3909 - val_accuracy: 0.8765
Epoch 14/17
39/39 [==============================] - 91s 2s/step - loss: 0.0814 - accuracy:
0.9740 - val_loss: 0.4560 - val_accuracy: 0.8645
Epoch 15/17
39/39 [==============================] - 92s 2s/step - loss: 0.0799 - accuracy:
0.9766 - val_loss: 0.3732 - val_accuracy: 0.8785
Epoch 16/17
39/39 [==============================] - 86s 2s/step - loss: 0.0354 - accuracy:
0.9922 - val_loss: 0.3454 - val_accuracy: 0.9084
Epoch 17/17
39/39 [==============================] - 87s 2s/step - loss: 0.0236 - accuracy:
0.9939 - val_loss: 0.3289 - val_accuracy: 0.9064
```

[8]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)
```

[9]:

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



[10]:
```
test_images, test_labels = [], []
for images, labels in test_ds:
    test_images.extend(images.numpy())
    test_labels.extend(labels.numpy())
```

[11]:
```
test_images = np.array(test_images)
test_labels = np.array(test_labels)
```

[12]:
```
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)
```

```
16/16 [==============================] - 7s 439ms/step
```

[13]:
```
print("Classification Report:")
print(classification_report(test_labels, predicted_labels,
 ↪target_names=class_names))

print("Confusion Matrix:")
cm = confusion_matrix(test_labels, predicted_labels)
print(cm)
```

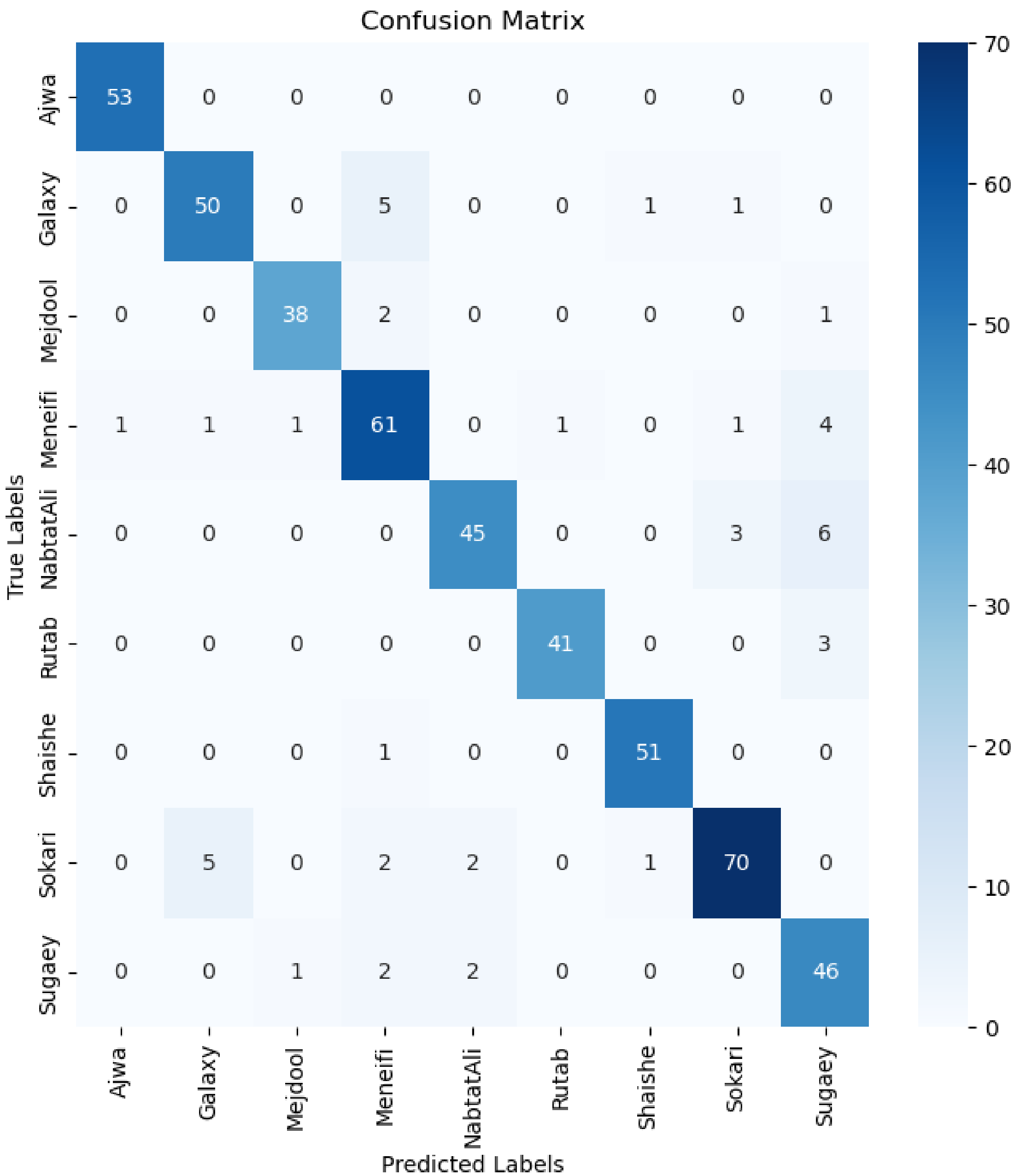
```
Classification Report:
              precision    recall  f1-score   support

        Ajwa       0.98      1.00      0.99        53
      Galaxy       0.89      0.88      0.88        57
     Mejdool       0.95      0.93      0.94        41
     Meneifi       0.84      0.87      0.85        70
   NabtatAli       0.92      0.83      0.87        54
       Rutab       0.98      0.93      0.95        44
     Shaishe       0.96      0.98      0.97        52
      Sokari       0.93      0.88      0.90        80
      Sugaey       0.77      0.90      0.83        51

    accuracy                           0.91       502
   macro avg       0.91      0.91      0.91       502
weighted avg       0.91      0.91      0.91       502

Confusion Matrix:
[[53  0  0  0  0  0  0  0  0]
 [ 0 50  0  5  0  0  1  1  0]
 [ 0  0 38  2  0  0  0  0  1]
 [ 1  1  1 61  0  1  0  1  4]
 [ 0  0  0  0 45  0  0  3  6]
 [ 0  0  0  0  0 41  0  0  3]
 [ 0  0  0  1  0  0 51  0  0]
 [ 0  5  0  2  2  0  1 70  0]
 [ 0  0  1  2  2  0  0  0 46]]
```

[14]:
```
accuracy = np.sum(test_labels == predicted_labels) / len(test_labels)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9063745019920318
```

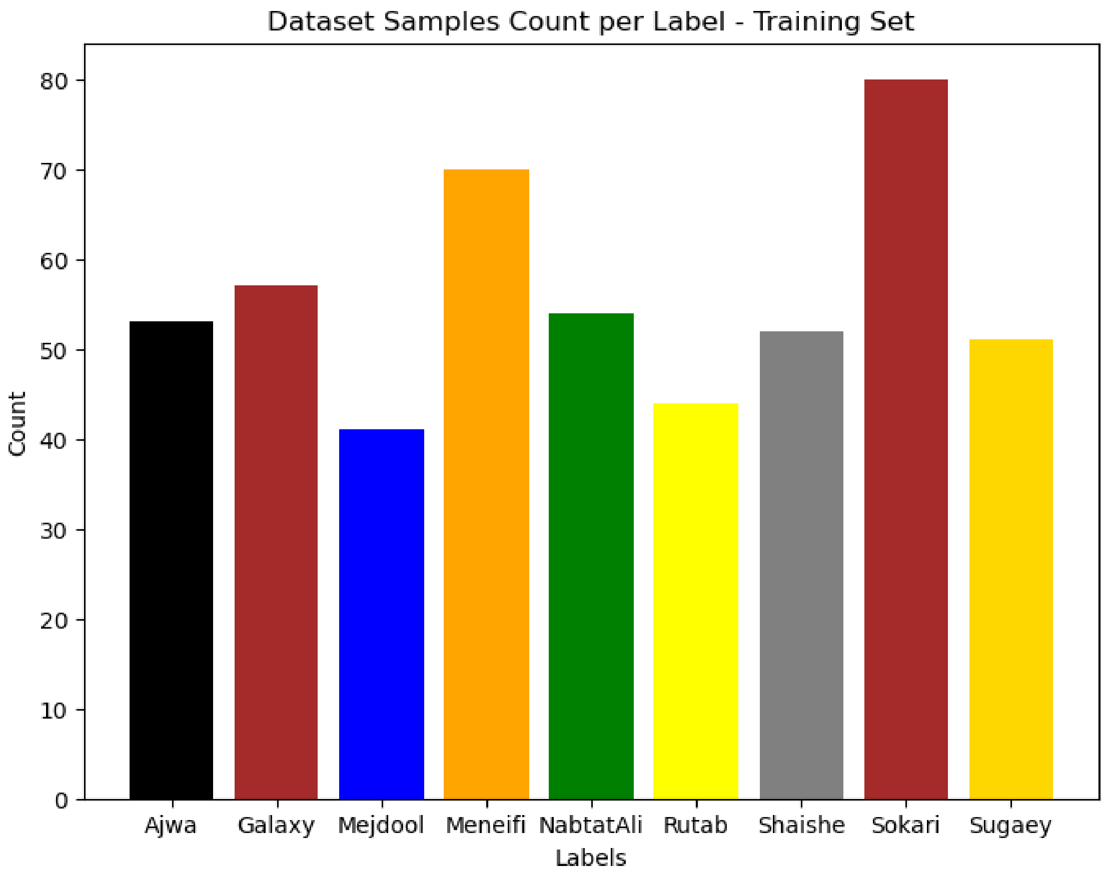
[15]:
```
plt.figure(figsize=(8, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_names,
 ↪yticklabels=class_names)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
```

```
plt.show()
```

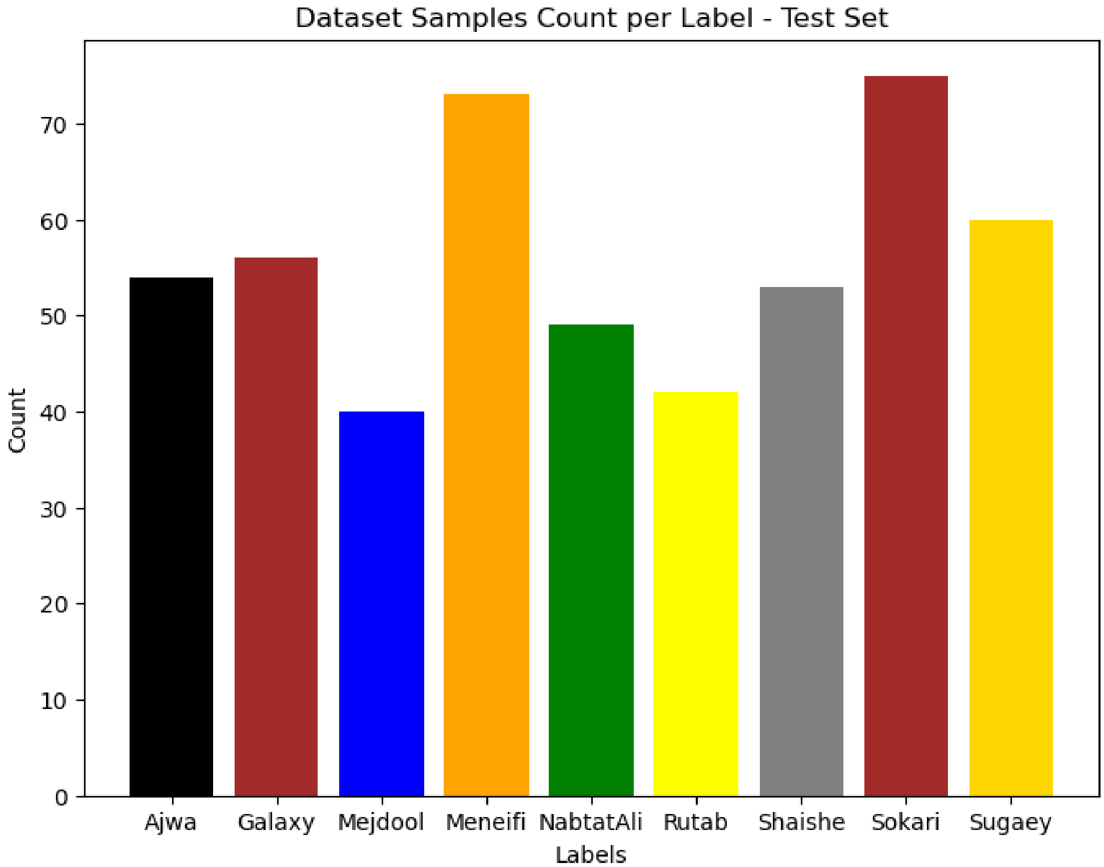


[16]:
```
c = ['black', 'brown', 'blue', 'orange', 'green', 'yellow', 'grey', 'brown',␣
 ↪'gold']

plt.figure(figsize=(8, 6))
train_labels_count = [len(np.where(test_labels == i)[0]) for i in␣
 ↪range(num_classes)]
```

```
plt.bar(class_names, train_labels_count, color = c)
plt.title("Dataset Samples Count per Label - Training Set")
plt.xlabel("Labels")
plt.ylabel("Count")
plt.show()
```



[17]:
```
plt.figure(figsize=(8, 6))
test_labels_count = [len(np.where(predicted_labels == i)[0]) for i in␣
 ↪range(num_classes)]
plt.bar(class_names, test_labels_count, color = c)
plt.title("Dataset Samples Count per Label - Test Set")
plt.xlabel("Labels")
plt.ylabel("Count")
plt.show()
```


Dataset Samples Count per Label - Test Set
Count
Labels
0
10
20
30
40
50
60
70
Ajwa
Galaxy
Mejdool
Meneifi
NabtatAli
Rutab
Shaishe
Sokari
Sugaey

```
[ ]:
```

```
[ ]:
```