

**Image classification Implementation using CNN.**

**CSC462 – Project**

Bader Ahmed al-Khalifa

441103284

Firstly, in this project we focus on the implementation of an image classification. The goal of this project is to build a model that can identify types of dates through processing images and building a model using CNN, so to do that we have to build a model that has a relatively high accuracy so we can classify the types of dates correctly,

To do that I used TensorFlow and other libraries so we can implement the model.

In this project I would be working alone

The platform used in this project is Jupiter notebook in the anaconda package and TensorFlow, Matplotlib, PIL, NumPy, sklearn, seaborn

In this homework I used some libraries in the scikit learn package which are:

The preprocessing used is:

1. Resizing: resizing the images
2. Normalization: rescaling the images
3. Batching: taking the images by batches of 30
4. Encoding: encoding the labels of the images
5. Shuffling: shuffling the training data

The CNN I used is the sequential model which used:

The Keras Sequential model consists of three convolution blocks (tf.keras.layers.Conv2D) with a max pooling layer (tf.keras.layers.MaxPooling2D) in each of them. There's a fully connected layer (tf.keras.layers.Dense) with 128 units on top of it that is activated by a ReLU activation function (relu) and a flatten layer, so we allow the model to learn complex patterns.

In this order:

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(),

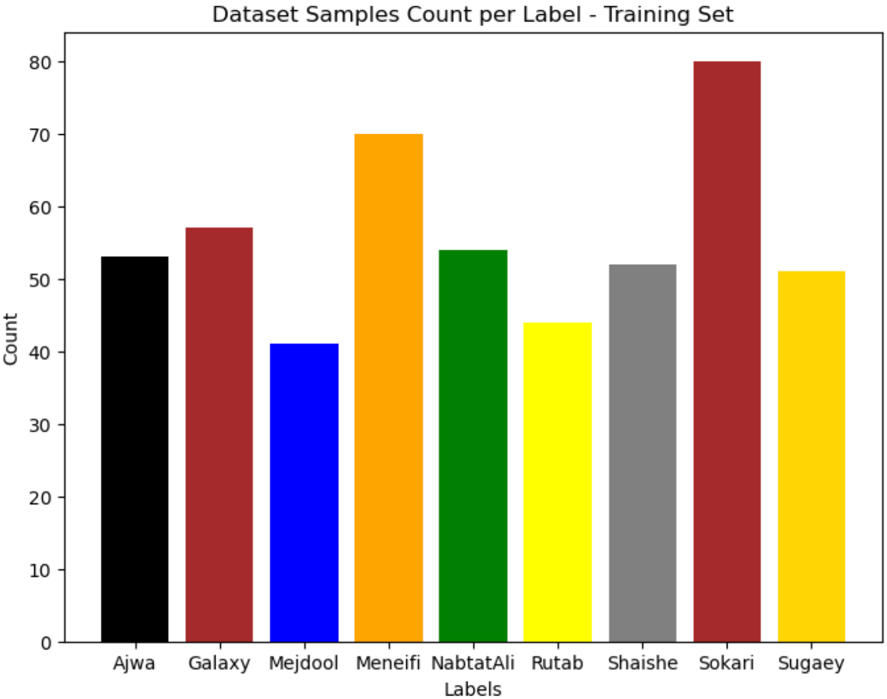
layers.Dense(128, activation='relu'),

layers.Dense(num\_classes)

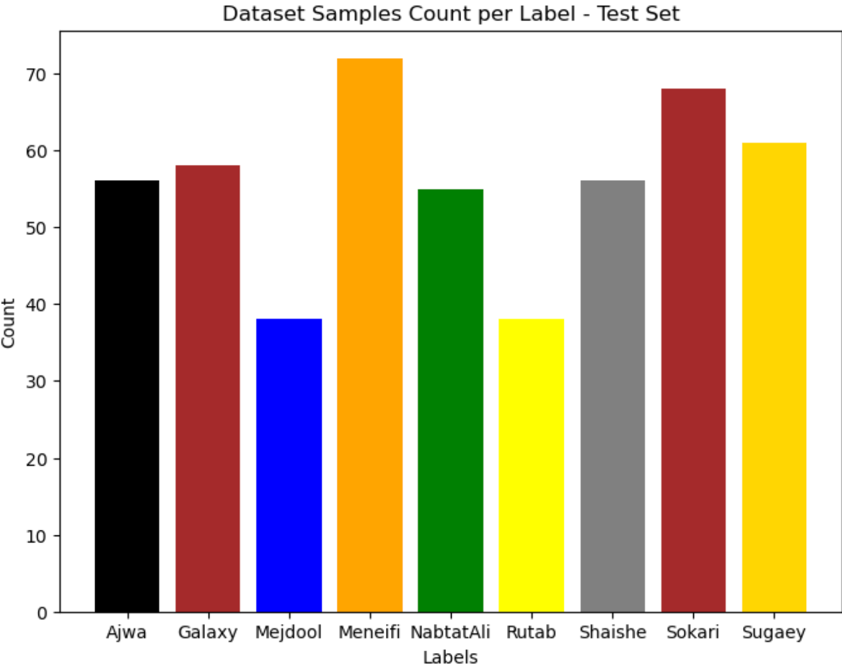
The Optimizer I used in this project was the Adam algorithm with no hyperparameter.

The number of epochs used in this project is 15, I choose 15 because I observed that if I chose a lower number in this model the accuracy isn’t at its highest and didn’t choose a higher number because the model took way too much time and didn’t give any added accuracy.

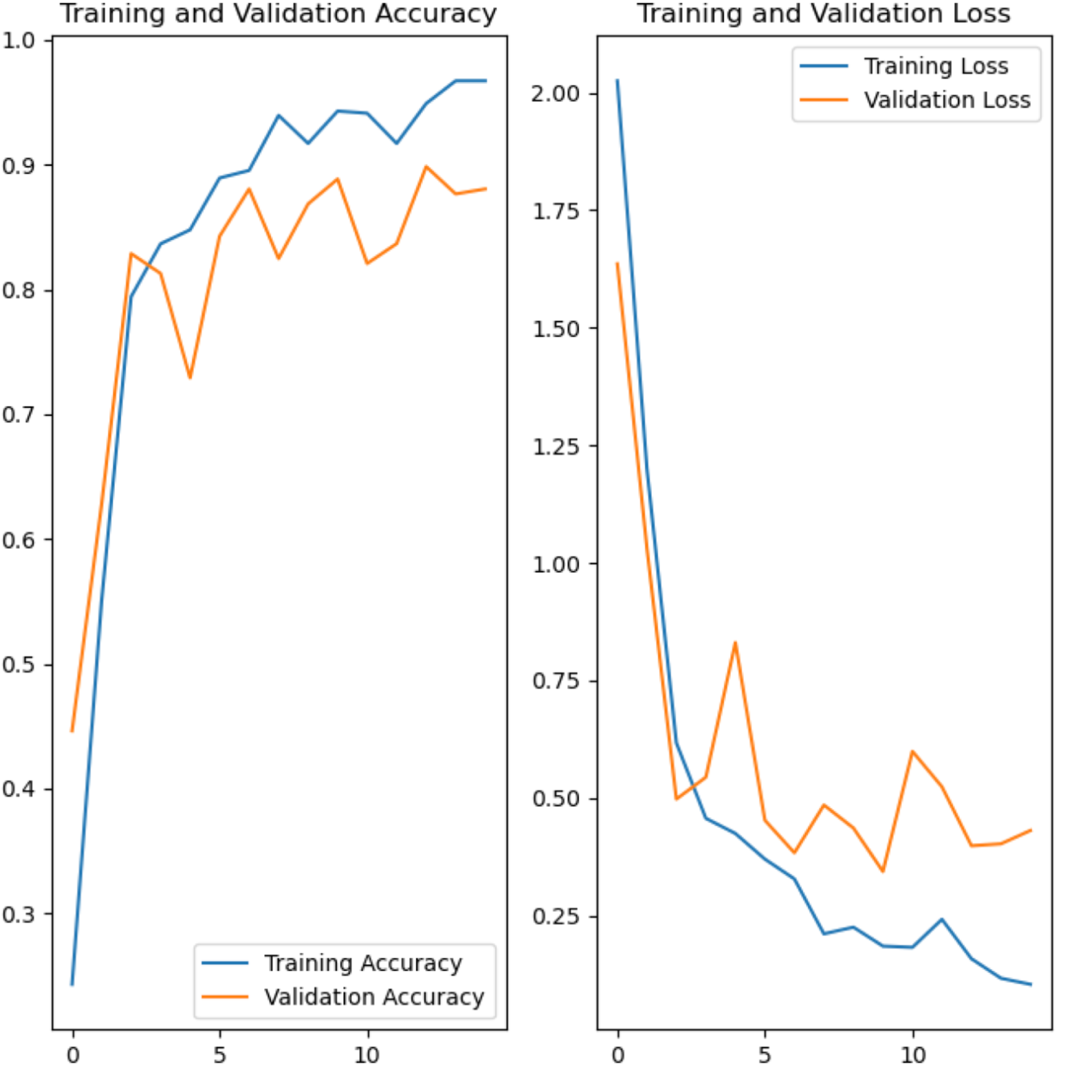
The Dataset samples counts per label charts for training set:



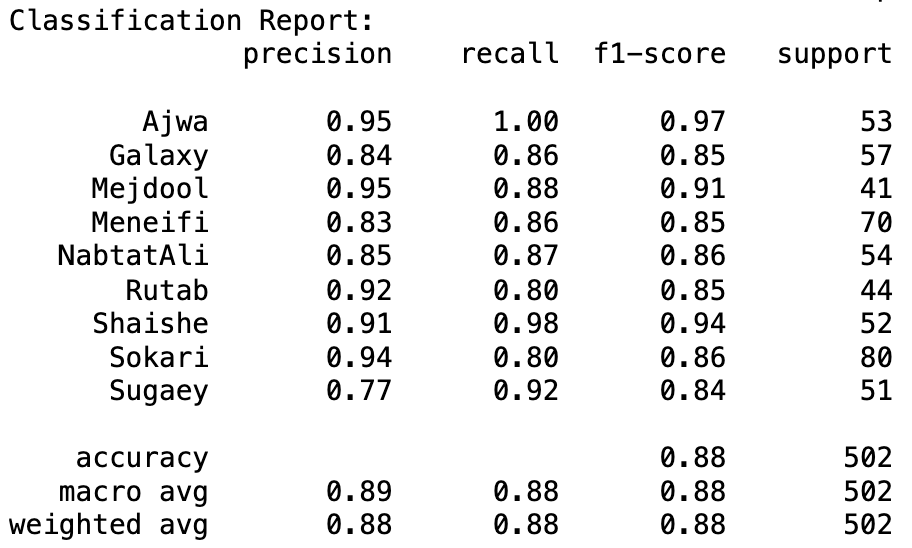
The Dataset samples counts per label charts for testing set:



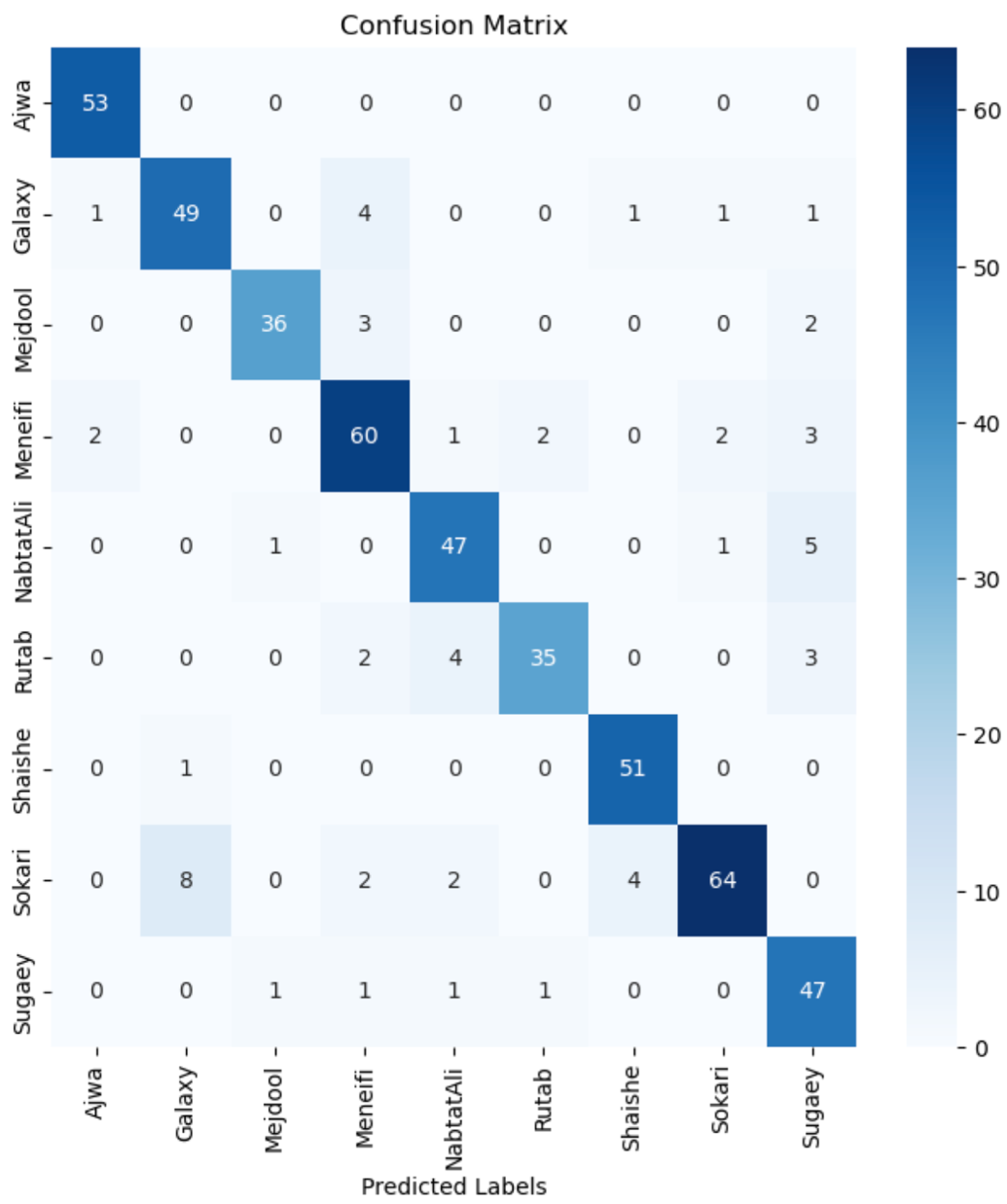
Plots of loss and accuracy on the training and validation sets per epoch:



classification report:



Confusion matrix:



And here’s the screen shoots of the code + the output:A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA table of numbers and symbols

Description automatically generated with medium confidenceA group of dates on a white background

Description automatically generatedA graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA graph with numbers and symbols

Description automatically generatedA close-up of a graph

Description automatically generated