“An AI-based Arabic handwritten character recognition module as a foundational component for automated exam grading systems”

Domain:

Artificial Intelligence / Machine Learning – Computer Vision (Handwritten Text Recognition)

Problem Statement:

Manual grading of Arabic handwritten exam papers is time-consuming, error-prone, and inconsistent due to handwriting variations. Automated grading systems require accurate recognition of handwritten Arabic text as a prerequisite step. However, Arabic handwriting recognition is challenging because of complex character shapes and visual similarity between letters. This project addresses this challenge by developing a Convolutional Neural Network (CNN) model capable of recognizing individual Arabic handwritten letters, which serves as a critical building block toward an automated Arabic exam grading system.

Project Statement

This project implements a CNN-based system to classify Arabic handwritten letters from scanned images. The trained model accurately recognizes 28 Arabic characters from grayscale images and is deployed using a Streamlit interface for real-time prediction. The system demonstrates the feasibility of Arabic handwritten character recognition as a first step toward automatic grading of handwritten exams.

Data collection and Processing using Python

1-anaconda installed, I used Anaconda prompt

2-Jupyter Notebook installed

3- Python libraries installed

Data source Link :

The dataset used for training and testing was sourced from a publicly available GitHub repository consisting of Arabic handwritten letter images labeled into 28 classes.

[Arabic-Handwritten-Characters-Dataset/Train Images 13440x32x32.zip at master · mloey/Arabic-Handwritten-Characters-Dataset]

Coding part :

Link to my files: train.py and app.py

https://github.com/UniverseTalker/AR-HW-CNN.git

- Input: 32x32 grayscale images

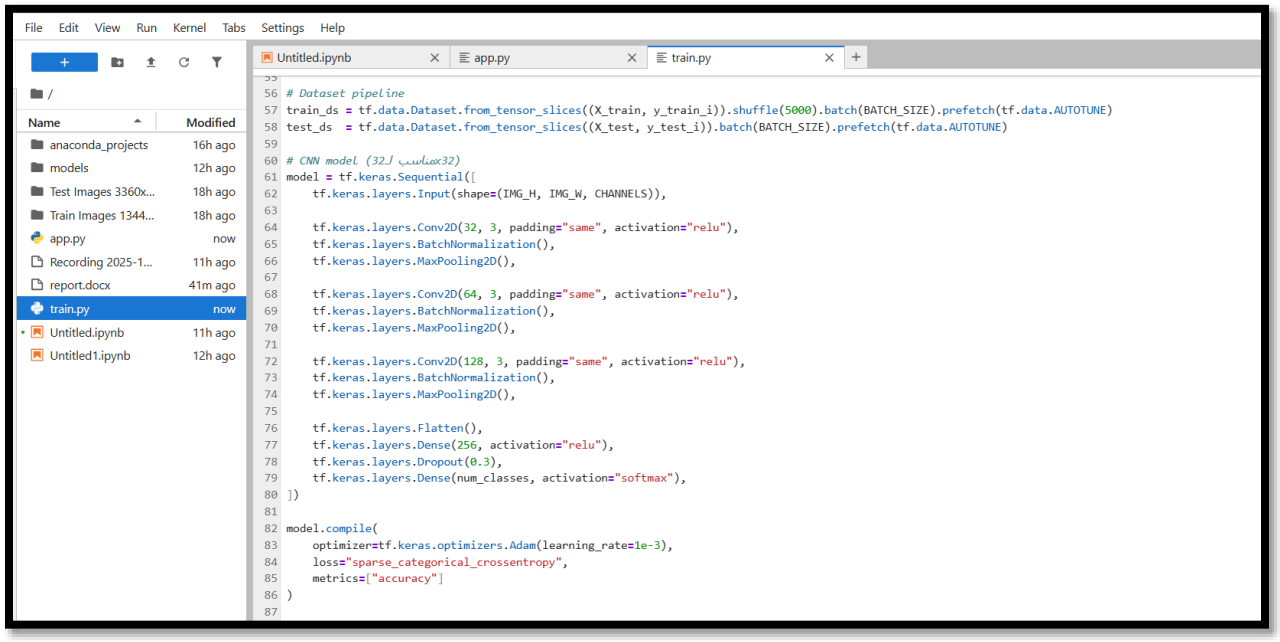
- Classes: 28 Arabic letters

- Model: **Convolutional Neural Network (CNN)**

- Framework: **TensorFlow / Keras**

- Deployment: Streamlit

In this picture part of the code related to the CNN Model training

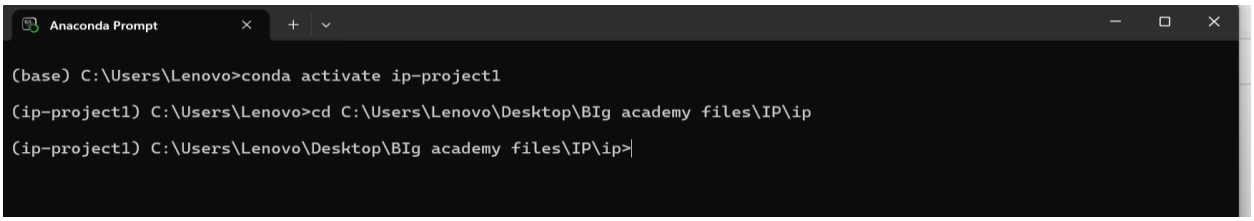


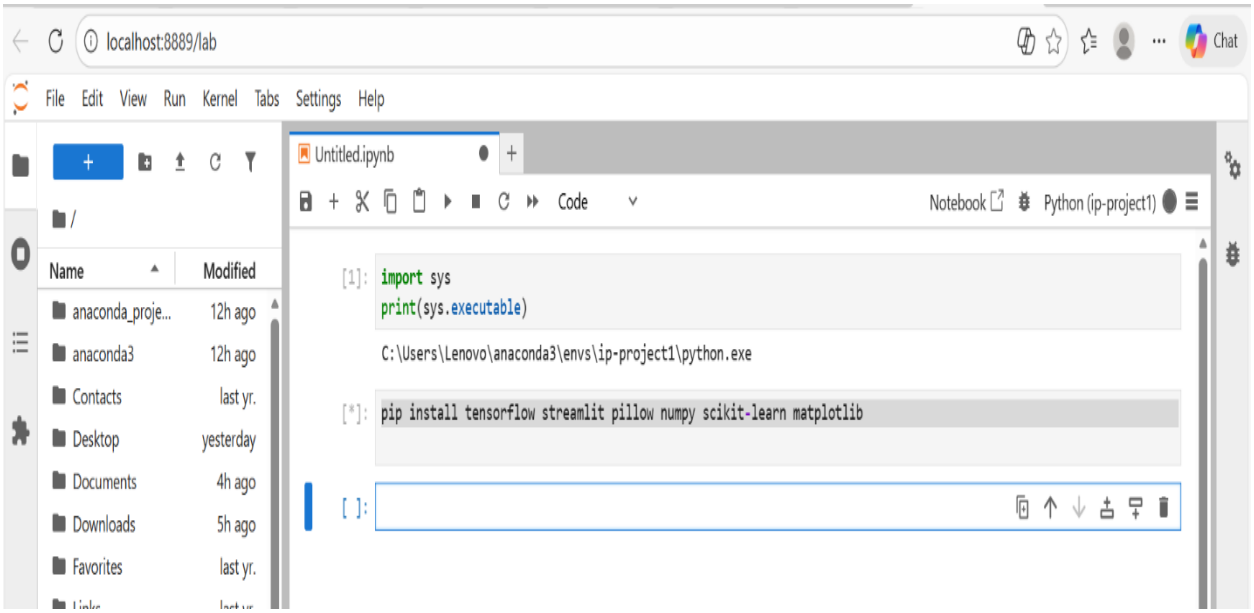
How to run the application:

1. Activate environment (mine is: ip-project1)

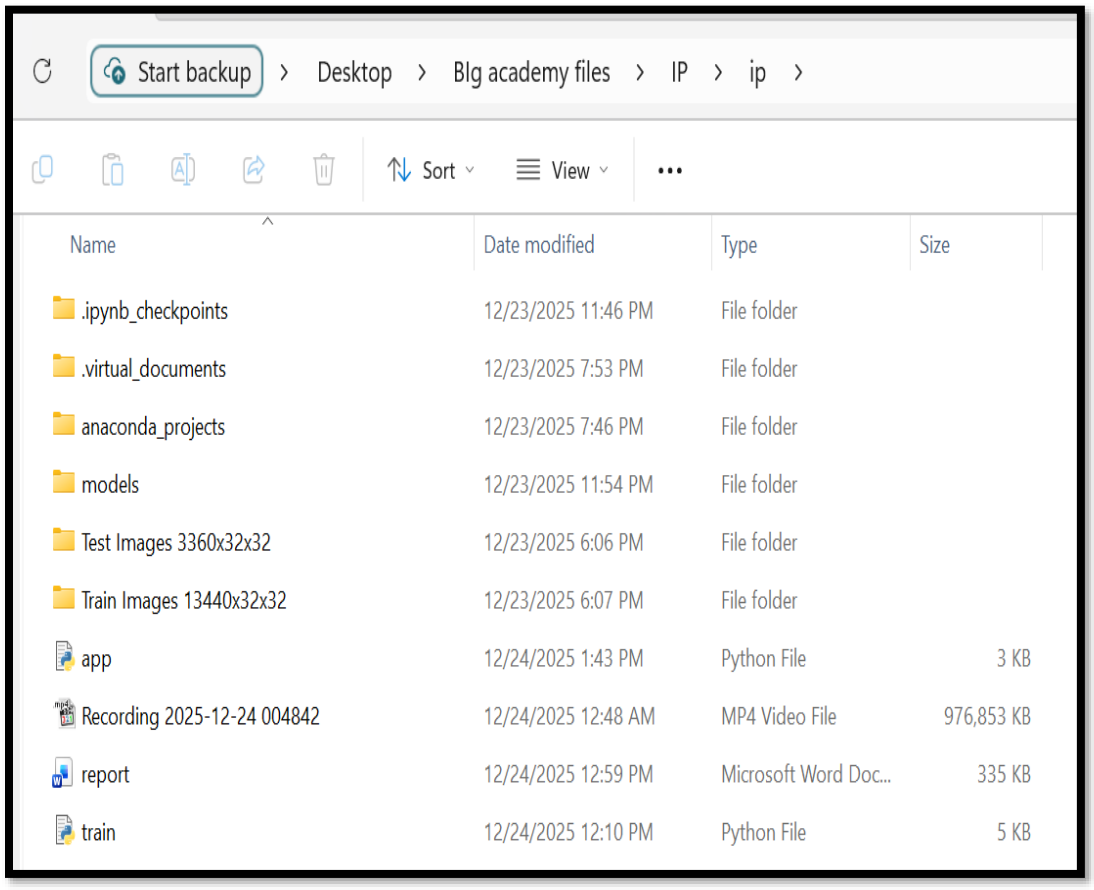
2. Run training: python train.py (then model file and json file created)

3. Run streamlit app : python -m streamlit run app.py





My project folder structure:



Examples of Prediction :

Arabic Letter Recognizer (CNN)

Upload a letter image (32x32 recommended). The model will predict the letter.

Upload an image

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files

id_7_label_4.png 171.0B

Uploaded image

Predicted letter

ث

Label: 4

Prediction

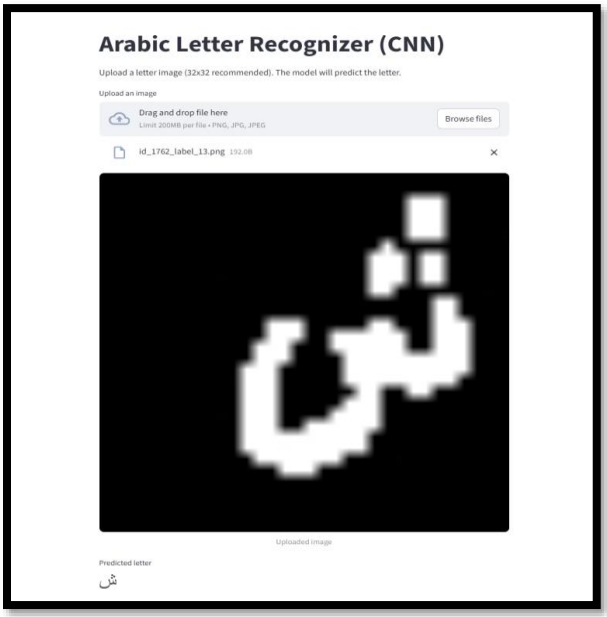
Confidence

99.12%

Top 3 predictions

1. ث (label 4) — 99.12%
2. ق (label 13) — 0.66%
3. ن (label 25) — 0.11%


Arabic Letter Recognizer (CNN)
Upload a letter image (32x32 recommended). The model will predict the letter.
Upload an image
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files
id_55_label_28.png 177.0B
Uploaded image
Predicted letter
ي
Label: 28


Arabic Letter Recognizer (CNN)
Upload a letter image (32x32 recommended). The model will predict the letter.
Upload an image
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files
id_1762_label_13.png 192.0B
Uploaded image
Predicted letter
ش

Model Evaluation : Accuracy, Confusion Matrix, and Precision/Recall:

Model accuracy : 0.954

In the code we can see it inside train.py file :

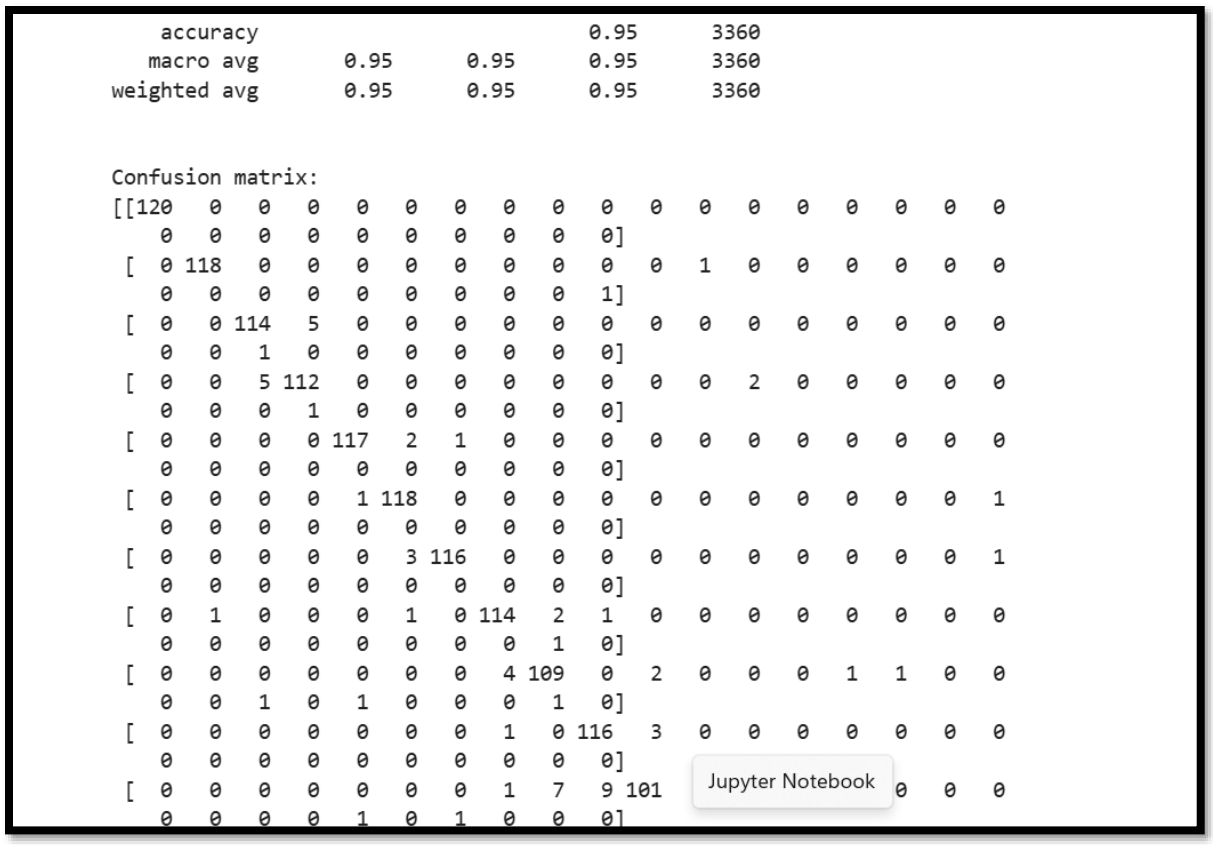
```
acc = accuracy_score(y_test_i, y_pred)
print("\nTest Accuracy:", acc)
```

Results while running file :train.py

```
Test Accuracy: 0.9541666666666667

Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       120
           1       0.99      0.98      0.99       120
           2       0.92      0.95      0.93       120
           3       0.93      0.93      0.93       120
           4       0.99      0.97      0.98       120
           5       0.93      0.98      0.96       120
           6       0.97      0.97      0.97       120
           7       0.94      0.95      0.95       120
           8       0.90      0.91      0.90       120
           9       0.91      0.97      0.94       120
```



```
              accuracy                           0.95      3360
             macro avg       0.95      0.95      0.95      3360
          weighted avg       0.95      0.95      0.95      3360


Confusion matrix:
[[120   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0 118   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   1]
 [  0   0 114   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   1   0   0   0   0   0   0   0]
 [  0   0   5 112   0   0   0   0   0   0   0   0   2   0   0   0   0   0
    0   0   0   1   0   0   0   0   0   0]
 [  0   0   0   0 117   2   1   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   1 118   0   0   0   0   0   0   0   0   0   0   0   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   3 116   0   0   0   0   0   0   0   0   0   0   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   1   0   0   0   1   0 114   2   1   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   1   0]
 [  0   0   0   0   0   0   0   4 109   0   2   0   0   0   1   1   0   0
    0   0   1   0   1   0   0   0   1   0]
 [  0   0   0   0   0   0   0   1   0 116   3   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   1   7   9 101   0   0   0   0   0   0   0
    0   0   0   0   1   0   1   0   0   0]
```

Jupyter Notebook

Dalal Adnan zamzam