Handwritten Arabic Character Recognition using CNN

Authors

- Mariam Mohamed sayed
- Ahmed Khaled mohamed
- Haidy abobaker mohamed

Contents

---

1. Project Overview

This project aims to develop a Convolutional Neural Network (CNN) model that can accurately recognize and classify handwritten Arabic letters. The system processes grayscale images of individual Arabic characters (each of size 32×32 pixels) and predicts the corresponding letter from 28 different classes.

---

2. Objective

To build a high-performance deep learning model capable of classifying handwritten Arabic characters with high accuracy. This model can serve as a core component in applications like:

- Optical Character Recognition (OCR)
- Educational tools for Arabic learning
- Digitization of handwritten Arabic text

- Accessibility tools for the visually impaired

---

3. Dataset Description

Feature	Description
Training Images	13,440
Test Images	3,360
Number of Classes	28 (Arabic letters from “ا” to “ي”)
Image Format	Grayscale, 32x32 pixels
Data Format	CSV files (flattened pixels + labels)

Labels are encoded from 1 to 28 corresponding to Arabic letters.

---

4. Data Preprocessing

- **Reshaping**: 1D to 2D (32x32), then to 4D for CNN input.
- **Normalization**: Pixel values scaled to [0, 1].
- **Orientation Fix**: Images rotated/flipped to correct orientation.
- **Label Encoding**: One-hot encoding using `to_categorical()`.
- **Train/Validation Split**: 80% training, 20% validation.

---

5. Model Architecture (CNN)

```
Input: 32x32x1

→ Conv2D (128 filters, 3x3, ReLU)
→ BatchNormalization
→ MaxPooling2D (2x2)

→ Conv2D (256 filters, 3x3, ReLU)
→ BatchNormalization
→ MaxPooling2D (2x2)

→ Conv2D (256 filters, 3x3, ReLU)
→ BatchNormalization
→ MaxPooling2D (2x2)

→ Flatten
→ Dense (256, ReLU) + Dropout (0.5)
→ Dense (128, ReLU) + Dropout (0.5)
→ Dense (28, Softmax)
```

6. Model Compilation & Training

Setting	Value
Optimizer	Adam
Loss	Categorical Crossentropy
Batch Size	64
Epochs	50 (EarlyStopping used)

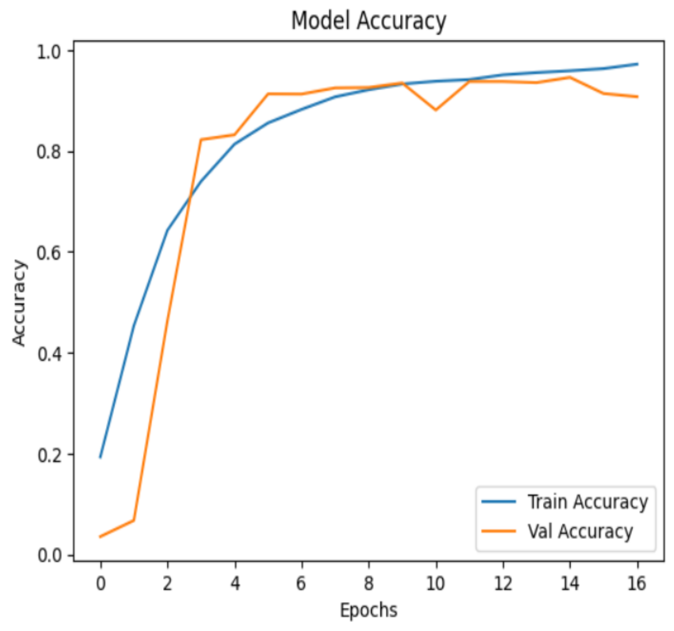
7. Training Performance (Sample Epochs)

Epoch	Accuracy	Val Accuracy	Val Loss
1	11.71%	5.13%	7.2483
4	72.13%	61.01%	1.1367
6	85.57%	89.92%	0.2832
12	94.90%	94.87%	0.1924
16	97.25%	93.49%	0.3048

Model stopped early at epoch 17.

8. Test Results

- **Test Accuracy**: 94.82%
- **Test Loss**: 0.2121



9. Classification Report

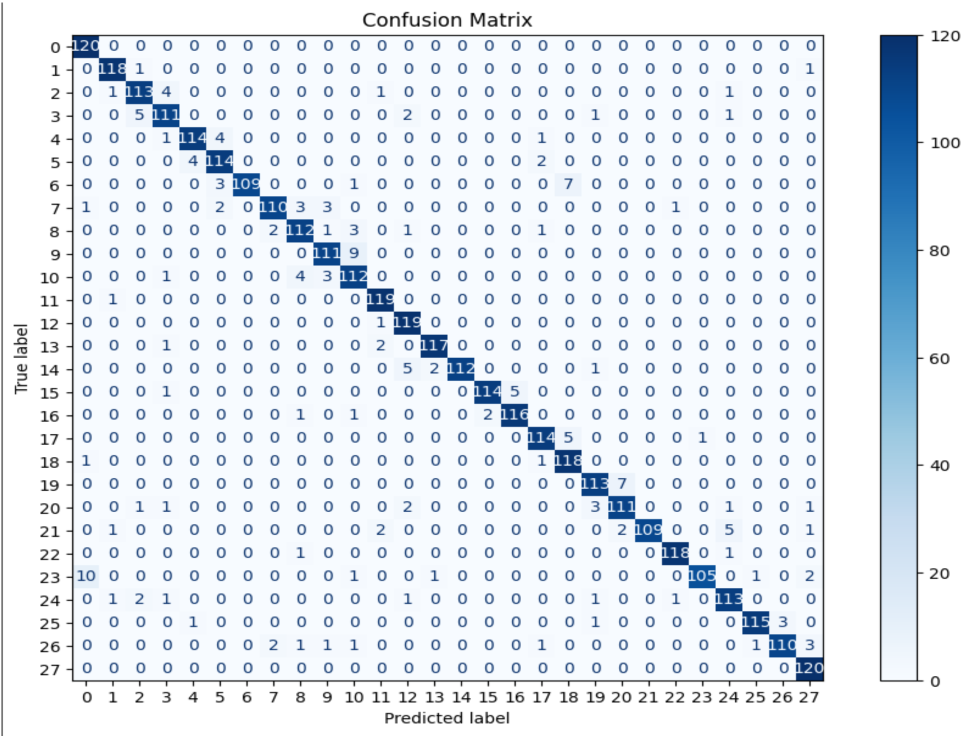
Metric	Value
Accuracy	94.82%
F1 Score	0.95
Precision	0.95
Recall	0.95

10. Confusion Matrix

- Diagonal = correct predictions
- Off-diagonal = common errors

Common Confusions:

- “ذ” vs “ز”
- “خ” vs “ح”
- “ص” vs “ض”



11. Challenges

- High similarity between many Arabic characters
- Handwriting variation among individuals
- Data imbalance for less frequent characters

---

12. Conclusion

The project demonstrates the successful implementation of a CNN-based handwritten Arabic character recognizer with ~95% accuracy. This can aid OCR systems, educational platforms, and accessibility tools.

---

13. Deployment

This section outlines the deployment process of the trained CNN model using **Flask**.

Full Flask App Code

```
import os
import numpy as np
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from PIL import Image
from arabic_mapping import get_arabic_letter

app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = load_model("model.h5")

def preprocess_image(image_path):
    img = Image.open(image_path).convert('L')
    img = img.resize((32, 32))
    img = np.array(img)
    img = np.rot90(img, k=3)
    img = np.fliplr(img)
    img = img.reshape(1, 32, 32, 1)
    img = img.astype('float32') / 255.0
    return img

@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    if request.method == 'POST':
        file = request.files['file']
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(filepath)

        image = preprocess_image(filepath)
        pred = model.predict(image)
        class_index = np.argmax(pred)
        prediction = get_arabic_letter(class_index + 1)

        return render_template('index.html', prediction=prediction,
image=file.filename)

    return render_template('index.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```

Notes

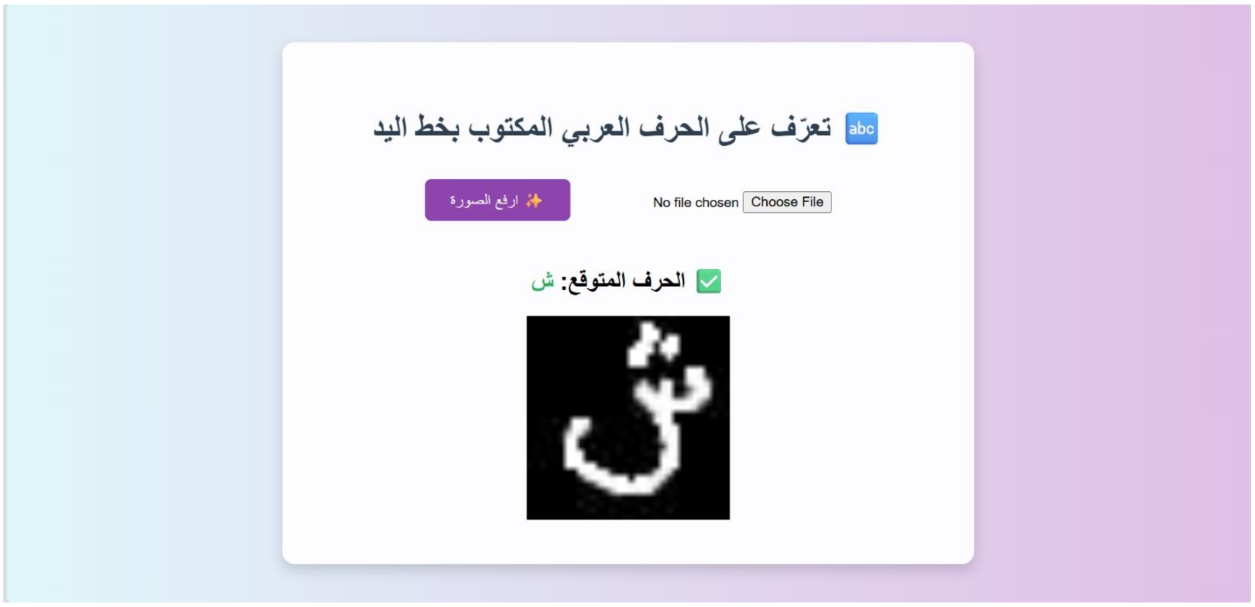
- `arabic_mapping.py` contains a helper function to convert class index to Arabic letter.
- Images are saved to the `static/uploads` folder.
- The result is rendered using `index.html` template.
-

Diagram of End-to-End Deployment Pipeline

The following diagram outlines the flow of data and processing in the system:

1. **User Interface (HTML)**: A simple form for users to upload handwritten Arabic character images.
2. **Flask Backend**: Receives the image, processes it, and runs prediction.
3. **Model Inference**: Uses the trained CNN model to classify the image.
4. **Arabic Mapping**: Converts predicted class index to Arabic letter.
5. **Result Display**: Returns the result along with the uploaded image.

```
          User Uploads Image (.png/.jpg)
                       │
                       ▼
                  [ HTML Form ]
                       │
                       ▼
           [ Flask App Receives Image ]
                       │
                       ▼
     [ Preprocess Image (resize, normalize) ]
                       │
                       ▼
          [ Predict with CNN Model (.h5) ]
                       │
                       ▼
        [ Map Class Index → Arabic Letter ]
                       │
                       ▼
     [ Show Result on Webpage (image + text) ]
```

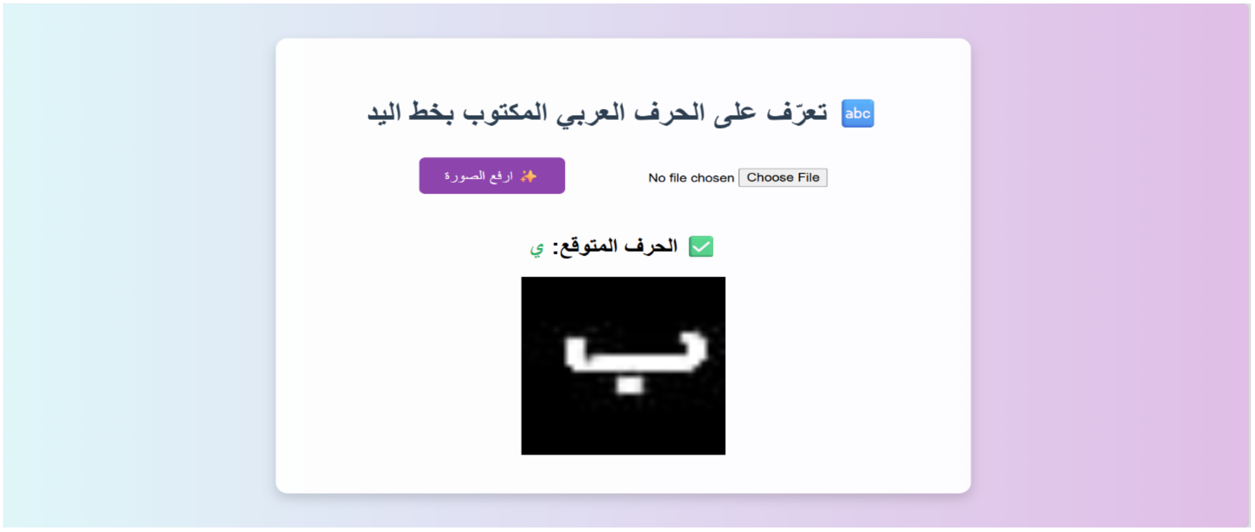


🔤 تعرّف على الحرف العربي المكتوب بخط اليد

✨ ارفع الصورة

Choose File No file chosen

✅ الحرف المتوقع: ش

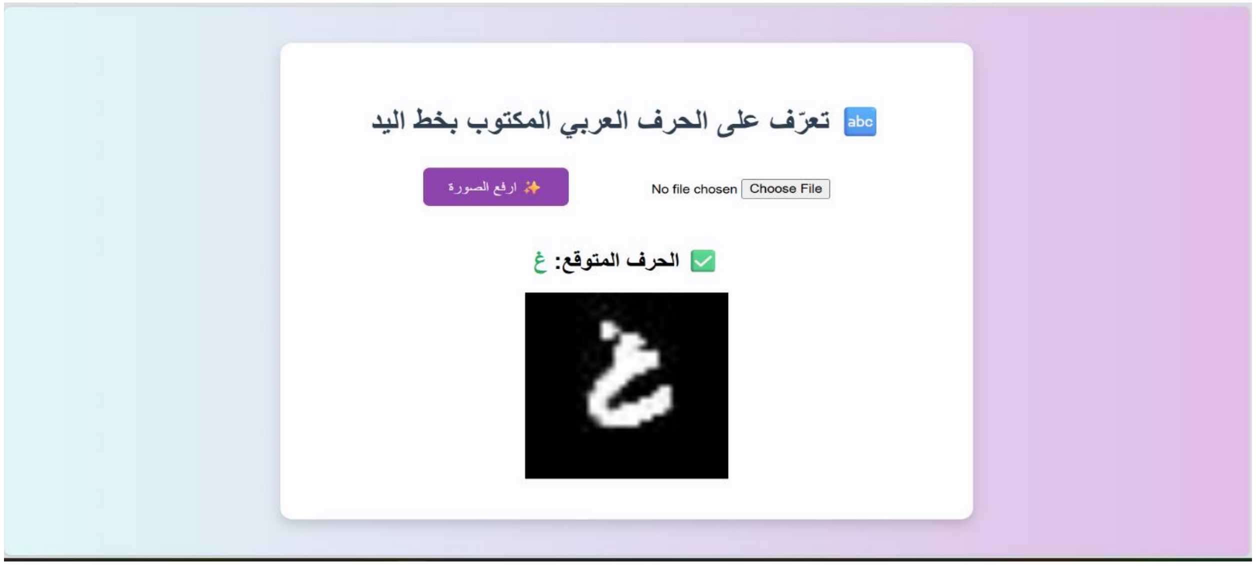


🔤 تعرّف على الحرف العربي المكتوب بخط اليد

✨ ارفع الصورة

Choose File No file chosen

✅ الحرف المتوقع: ي



🔤 تعرّف على الحرف العربي المكتوب بخط اليد

✨ ارفع الصورة

Choose File No file chosen

✅ الحرف المتوقع: غ