Université d'Alger

Département Informatique

Module Data Mining

Master 1 ISII Ingénier des systèmes informatiques intelligents

---

Rapport de projet

Arabic letters recognition

Réalisé par :

Mokhtari Dhia elhak
Ait Mokhtar Amine
Benseghir Lila Siham

Année universitaire : 2023 / 2024

General Introduction

The recognition of handwritten characters is a complex problem in the field of computer vision and pattern recognition. With the increasing digitization of handwritten documents, automated recognition systems have become essential. This project focuses on developing Convolutional Neural Network (CNN) models to classify handwritten Arabic letters. Leveraging the architecture from the paper "Recognizing Basic Compound and Numerical Bangla Handwritten Characters with Convolutional Neural Network," we applied and adapted these methodologies to our dataset of Arabic letters.

Handwritten character recognition has a rich history, with early methods relying heavily on traditional image processing and feature extraction techniques. More recently, the advent of deep learning, particularly CNNs, has revolutionized the field. CNNs have demonstrated exceptional performance in image classification tasks due to their ability to automatically learn hierarchical features from raw pixel data.

Arabic script adds an extra layer of complexity to the problem due to its cursive nature, contextual character shapes, and the presence of numerous ligatures and diacritical marks. Arabic characters can have up to four different forms depending on their position in a word (isolated, initial, medial, and final), making the classification task more challenging compared to scripts with non-cursive writing systems.

This project aims to develop and evaluate Convolutional Neural Network (CNN) models for the classification of handwritten Arabic letters. CNNs have shown significant success in various image classification tasks due to their ability to automatically learn and extract hierarchical features from raw pixel data. By leveraging the architecture from the paper "Recognizing Basic Compound and Numerical Bangla Handwritten Characters with Convolutional Neural Network," we applied and adapted these methodologies to our dataset of Arabic letters.

1. Methodology:

The methodology section outlines the systematic approach taken to develop and evaluate the CNN models for Arabic handwritten letter recognition. This section details the dataset used, the preprocessing steps applied to prepare the data for modeling, and the architectural design of the CNN models. Each step in this process is crucial for ensuring the accuracy and reliability of the classification models. By carefully collecting and processing the data and designing effective model architectures, we aim to build robust systems capable of recognizing Arabic handwritten characters with high accuracy.

1.1 Dataset Collection and Description:

The first step in our methodology involved gathering a comprehensive dataset of Arabic handwritten letters. The quality and diversity of the dataset are critical to the success of the CNN models. The dataset used in this project was collected through a collaborative effort involving students and supervised by our instructor. The dataset consists of 23,327 images of handwritten Arabic letters, divided into 62 classes. Each image was originally a 128x128 color image. The classes are not well balanced, with some letters having significantly more samples than others.

1.2 Data Preprocessing:

To prepare the data for training, several preprocessing steps were performed:

- **Resizing:** Each image was resized from 128x128 to 32x32 pixels to reduce computational load and memory usage.
- **Grayscale Conversion:** The color images were converted to grayscale to simplify the data and focus on the shapes of the letters.
- **Binarization:** The grayscale images were binarized to create black and white images, highlighting the letter shapes.

- **Inversion:** The binarized images were inverted so that the letters appeared in white on a black background, which is more intuitive for the network to process.

The processed dataset was then split into training, validation, and test sets with the following distribution:

Training set: 70% (16,329 images)

Validation set: 15% (3,499 images)

Test set: 15% (3,499 images)

These steps ensure that the data fed into the CNN models is optimized for learning the relevant features of Arabic letters.

1.3 Model Architecture:

The architecture of the neural network significantly impacts its ability to learn and generalize from the training data. In this project, we explored two different CNN architectures inspired by a successful model used for Bangla handwritten character recognition. This section will delve into the design of these architectures, detailing the layers, activation functions, and the rationale behind choosing these specific configurations. Understanding the model architecture helps in appreciating how the network processes the input images and learns to classify them accurately.

Model 1 Architecture:

Input layer: 32x32 grayscale image

Convolutional layers: A series of convolutional layers with ReLU activations

Pooling layers: Max-pooling layers to reduce spatial dimensions

Fully connected layers: Dense layers to integrate features before classification

Output layer: Softmax activation for multi-class classification with 62 neurons

Parameters: Total trainable parameters: 170,580

Model 2 Architecture:

Input layer: 32x32 grayscale image

Convolutional layers: More layers and filters compared to Model 1

Pooling layers: Max-pooling layers interspersed between convolutional layers

Fully connected layers: Larger dense layers to handle increased feature complexity

Output layer: Softmax activation for multi-class classification

Parameters: Total trainable parameters: 8,157,438

2. Experimental Setup

Setting up the experiments involves defining the training process, selecting appropriate hyperparameters, and establishing evaluation metrics. This section outlines the experimental protocols followed to train and validate our CNN models. By detailing the training regimen and the methods used for hyperparameter tuning, we aim to provide a clear understanding of how the models were optimized and evaluated. This section ensures that our approach is transparent and reproducible, which is essential for scientific rigor.

2.1 Model training:

Training a neural network involves iteratively updating the model's parameters to minimize the error in predictions. This section describes the training process, including the choice of optimizer, loss function, and the number of epochs. We will also discuss how the training data was utilized and the role of the validation set in monitoring the model's performance during training. Understanding the training process is key to interpreting the results and ensuring that the model learns effectively from the data.

Both models were trained using TensorFlow with the following configuration:

Optimizer: Adam optimizer

Loss function: Categorical cross-entropy

Metrics: Accuracy

Model 1 Training

Epochs: 50

Batch size: 32

Model 2 Training

Epochs: 25 (observed to reach performance peak around 20 epochs)

Batch size: 32

2.2 Hyperparameter Tuning

Hyperparameters such as learning rate, batch size, and the number of epochs were tuned through a combination of grid search and manual experimentation. Early stopping and model checkpointing were employed to prevent overfitting and to save the best performing model during training.

3. Results

In this section, we present the accuracy of the two CNN models trained on our dataset.

3.1 Model performance

The performance of the two models on the test set is summarized in the table below:

Model	# Of parameters	Training epochs	Accuracy
Model 1	170,580	50	90.23%
Model 2	8,157,438	25	92.58%

3.2 Comparative analysis

Model 2, with its deeper architecture and significantly larger number of parameters, achieved a higher test accuracy compared to Model 1. The additional complexity allowed Model 2 to capture more intricate features of the handwritten letters, leading to better classification performance.

4. Discussion

4.1 Analysis of model performance:

The higher accuracy of Model 2 highlights the advantage of using deeper and more complex architectures for image classification tasks. However, this comes at the cost of increased computational resources and longer training times. Despite the data imbalance, both models performed reasonably well, suggesting that the CNN architectures were effective in learning discriminative features from the input images.

4.2 Challenges and limitations:

Every research project faces challenges and limitations that can impact the outcomes. This section will discuss the main challenges encountered during the project, such as data imbalance, computational constraints, and any issues related to model training. We will also outline the limitations of our study, providing a realistic assessment of what was achieved and what could be improved. Acknowledging these challenges and limitations is crucial for setting realistic expectations and planning future work. the limitation of our project are the following :

- **Data Imbalance:** The dataset's imbalance may have caused the models to perform better on classes with more samples while underperforming on less represented classes.

- **Computational Resources:** Training the more complex Model 2 required significant computational power and memory, which could be a limitation in resource-constrained environments.

- **Image Quality:** Variations in handwriting quality and style added noise to the dataset, potentially affecting model performance.

5. Future Work and discussion:

Based on our findings and the challenges encountered, this section will propose recommendations for future work. We will suggest ways to improve the dataset, enhance the model architectures, and apply additional techniques to boost performance. Future work may also involve exploring new technologies and methodologies to address the limitations identified in this study. This section aims to provide a roadmap for further research and development in the field of Arabic handwritten character recognition.

- **Image Processing Techniques:**

  Applying advanced image processing techniques, such as noise reduction, edge detection, and morphological operations, could enhance the quality of the input images and improve classification accuracy.

- **Data Augmentation Strategies**

  Implementing data augmentation techniques, such as rotation, scaling, and translation, can artificially increase the dataset size and balance the classes. This would help the models generalize better to new, unseen data.

- **Dataset Improvement**

  Collecting more samples for underrepresented classes and ensuring all Arabic letters are included in the dataset would address the data imbalance issue. Additionally, creating a more diverse dataset with various handwriting styles would enhance the model's robustness.

- **Model Optimization**

  Experimenting with different architectures, hyperparameters, and regularization techniques (e.g., dropout, L2 regularization) can lead to further improvements in model performance. Transfer learning from pre-trained models on large datasets could also be explored.

Development of a GUI

Creating a graphical user interface (GUI) that can automatically segment and classify handwritten Arabic letters would make the system more

accessible and practical for real-world applications. The GUI could include features for real-time handwriting recognition and feedback.

6. Conclusion

This project successfully developed and evaluated two CNN models for the classification of Arabic handwritten letters, achieving high accuracy levels. The results demonstrate the effectiveness of CNNs in this domain and provide a solid foundation for further improvements. The recommended enhancements, such as advanced image processing, data augmentation, and model optimization, can lead to even better performance and practical applications. The output models and their configurations have been saved in H5 and JSON formats, respectively, and are included in the project solution for further reference and use. The development of a user-friendly GUI is the next step to making this technology accessible for broader use.