Deep Learning ForHadeeth Recognition

BENLALA Raid Athmane*, BOUDEFEL Selssabil*
*Department of the Class Superior, ESTIN, Bejaia, Algeria
Emails: r_benlala@estin.dz, s_boudefel@estin.dz

***Abstract*—This paper introduces a novel methodology for the recognition of handwritten Hadeeth in Arabic script, leveraging image-based processing and character segmentation techniques. The dataset comprises images of Hadeeth containing multiple lines and words, requiring a structured segmentation approach. Our method involves three primary stages: line segmentation, word segmentation, and character segmentation. The segmented characters are organized into a dataset of 28 directories, each named after an Arabic character, containing multiple samples to train the classification model. During the recognition process, the system segments characters from input Hadeeth images, classifies each character, and concatenates the recognized characters to reconstruct and identify the words or complete Hadeeth. Experimental results demonstrate the system's capability to accurately detect and classify Arabic characters, offering a robust solution for handwritten Arabic text recognition. This research contributes significantly to Arabic language processing and provides a foundation for applications in Islamic studies and digital archiving.**

I. Problem Description

The recognition of handwritten or image-based Arabic Hadeeth presents a unique challenge due to the complex structure of the Arabic script, which involves cursive writing, diacritics, and contextual character shaping. This problem is further compounded by the scarcity of labeled datasets tailored to Hadeeth recognition. Developing an effective system for recognizing and digitizing Hadeeth is essential for enhancing accessibility, preserving Islamic heritage, and facilitating scholarly research.

This project aims to address these challenges by creating a robust pipeline that segments Hadeeth images into lines, words, and characters and accurately recognizes the text. The ultimate goal is to classify and reconstruct the segmented characters to enable automated Hadeeth recognition with high precision.

II. Dataset Overview

The dataset used for this project is custom-built, involving significant effort to ensure accuracy and completeness. The creation process is as follows:

A. Data Collection

Images of Hadeeth were sourced from *Sunnah.com*, a reliable repository of Islamic texts. Screenshots of Hadeeth were captured for processing.

B. Segmentation Pipeline

- **Line Segmentation:** Each image was segmented into individual lines.
- **Word Segmentation:** Lines were further segmented into individual words.
- **Character Segmentation:** Words were finally segmented into individual characters.

C. Character Categorization

- Each segmented character was manually categorized into one of 28 directories, corresponding to the 28 Arabic alphabet characters.
- Each directory contains 20 images per character, ensuring sufficient samples for training deep learning models.

D. Supervision

The segmentation and categorization processes were manually supervised to maintain a high standard of data quality and integrity.

III. Methodology and Approach

Our approach to Arabic Hadeeth recognition involves a systematic pipeline comprising segmentation, classification, and reconstruction. The methodology is as follows:

- **Segmentation Pipeline: The segmentation pipeline used in this project is based on an existing implementation, adapted to meet the specific requirements of Hadeeth recognition.**
  - **Preprocessing:**
    - Convert input images to grayscale and invert the pixel values.
    - Apply Otsu's binarization to create a binary image.
    - Deskew the binary image using the best alignment angle determined via projection-based scoring.
  - **Line Segmentation:**
    - Use horizontal projection to detect and extract individual text lines.
    - Implement the function `line_horizontal_projection()` to segment lines based on gaps in horizontal projections.
  - **Word Segmentation:**
    - Use vertical projection within each line to detect and extract words.

    - ∗ Reverse the order of extracted words to account for Arabic's right-to-left writing direction.
  - **Character Segmentation:**
    - ∗ A pre-existing script, `character_segmentation.py`, is used to perform character segmentation from words.
    - ∗ Extracted characters are saved into directories corresponding to the 28 Arabic alphabet characters.
- **Classification:**
  - Train a classifier using the segmented character dataset, where each directory contains 20 images per character.
  - The classifier predicts individual characters based on the segmented input.
- **Reconstruction:**
  - Recognized characters are concatenated in sequence to reconstruct words and lines.
  - Reconstructed text is matched to detect and recognize the Hadeeth accurately.

IV. Model Architecture

The proposed model for Arabic Hadeeth recognition is a Convolutional Neural Network (CNN) that uses several layers for feature extraction, regularization, and classification. The architecture can be summarized as follows:

- **Input Layer:**
  - The input image has a shape of $(60, 60, 1)$, representing grayscale images of size 60x60 pixels.
- **First Convolutional Block:**
  - **Convolutional Layer:** A 32-filter convolution with a 3×3 kernel, using ReLU activation.
  - **Batch Normalization:** Applied after the convolution to normalize the feature maps.
  - **MaxPooling Layer:** A 2×2 max-pooling layer reduces the spatial dimensions.
  - **Dropout:** Dropout with a rate of 0.25 is applied for regularization, helping to prevent overfitting.
- **Second Convolutional Block:**
  - **Convolutional Layer:** A 64-filter convolution with a 3×3 kernel and ReLU activation.
  - **Batch Normalization:** Normalization is applied after the convolution to improve convergence.
  - **MaxPooling Layer:** A 2×2 max-pooling layer to reduce the feature map size.
  - **Dropout:** Dropout with a rate of 0.25 to prevent overfitting and enhance generalization.
- **Third Convolutional Block:**
  - **Convolutional Layer:** A 128-filter convolution with a 3×3 kernel and ReLU activation.
  - **Batch Normalization:** Applied after the convolution for faster and more stable training.
  - **MaxPooling Layer:** A 2×2 max-pooling layer reduces the size of the feature map.
  - **Dropout:** Dropout with a rate of 0.4 is applied to regularize the network further.
- **Fully Connected Layers:**
  - **Flatten Layer:** Flattens the output from the convolutional layers into a one-dimensional vector.
  - **Dense Layer:** A fully connected layer with 128 neurons and ReLU activation, followed by batch normalization.
  - **Dropout:** Dropout with a rate of 0.5 to prevent overfitting.
  - **Dense Layer:** A fully connected layer with 64 neurons and ReLU activation, followed by another dropout layer with a rate of 0.5.
- **Output Layer:**
  - **Dense Layer:** The final output layer consists of 28 neurons, corresponding to the 28 Arabic characters, with softmax activation to produce class probabilities.

A. Reasons for Choosing this Architecture

The architecture proposed for the Arabic Hadeeth recognition task is based on a Convolutional Neural Network (CNN), a well-established approach for image classification tasks, particularly when dealing with visual patterns like handwriting recognition. Below are the key reasons behind the selection of each component of the model architecture:

Convolutional Layers

- **Feature Extraction:** The use of multiple convolutional layers (32, 64, and 128 filters) enables the model to automatically learn hierarchical features from the input images. This is particularly useful for recognizing the intricate patterns and shapes inherent in Arabic script, which includes complex cursive writing and character shapes.
- **Small Kernels (3x3):** Small kernels are used in the convolutional layers to capture local patterns, which is beneficial for handwriting recognition, where small details such as dots or lines can change the meaning of a character.

Batch Normalization

- **Faster Convergence:** Batch normalization is applied after each convolutional layer. It normalizes the outputs of the previous layer, helping to stabilize and accelerate the training process. This is particularly important for deep networks to avoid issues like vanishing or exploding gradients.
- **Improved Generalization:** By reducing internal covariate shift, batch normalization helps the model generalize better, leading to improved accuracy on unseen data.

Max-Pooling Layers

- **Dimensionality Reduction:** The pooling layers reduce the spatial dimensions of the feature maps, effectively downsampling the image while retaining the most important information. This helps the model to focus on

essential features like the shape of Arabic characters, regardless of their position in the image.
- **Invariance to Translations:** Max-pooling increases the model's invariance to small translations and distortions, which is important for recognizing characters that might be written with slight variations or shifts in position.

Dropout Layers

- **Prevention of Overfitting:** Dropout is applied in both the convolutional and dense layers to regularize the model and prevent overfitting. This is crucial, especially when working with relatively smaller datasets for character segmentation tasks like this one.
- **Improved Robustness:** By randomly deactivating a fraction of neurons during training (e.g., 25%, 40%, and 50%), dropout forces the model to learn more robust and generalized features that are not overly reliant on any specific neuron.

Fully Connected Layers (Dense Layers)

- **Higher-level Representations:** The fully connected layers (128 and 64 neurons) enable the model to combine the features learned by the convolutional layers and make more abstract decisions about the Arabic characters. This is necessary to recognize the final pattern, especially when dealing with complex word structures.
- **Non-linear Decision Boundaries:** The use of ReLU activations in the dense layers adds non-linearity, allowing the model to capture more complex relationships between the features and the output classes (Arabic characters).

Output Layer (Softmax Activation)

- **Multi-Class Classification:** The final dense layer outputs 28 values, corresponding to the 28 Arabic characters. The softmax activation function ensures that the model's output represents the probability distribution across all possible classes, enabling the model to select the most likely character.
- **Categorical Cross-Entropy Loss:** Since this is a multi-class classification problem, the categorical cross-entropy loss function is used to optimize the model during training, ensuring that it learns to minimize the classification error.

B. Why This Architecture is Suitable for Arabic Hadeeth Recognition

Handling Complex Arabic Script

- Arabic handwriting and Hadeeth texts involve many variations in character shapes, cursive writing, and diacritics. The CNN with deep convolutional layers can extract fine-grained features that are essential for recognizing different forms of characters.
- By employing a multi-layered structure, this architecture can capture increasingly abstract patterns of characters at different levels of the image.

Data Regularization

- With the relatively limited amount of labeled data for Arabic Hadeeth, the use of dropout and batch normalization is essential to ensure that the model does not overfit to the training data. This enables better generalization to new, unseen data.

Efficient Training

- The combination of convolutional, max-pooling, and dropout layers allows the model to learn efficiently without becoming overly complex or slow to train. These techniques help speed up training by reducing the dimensionality of the input at each layer while keeping the most useful features.

Adaptation to Character Segmentation Task

- The model is structured to handle character-level recognition after performing segmentation (line, word, and character segmentation). This is ideal for recognizing individual Arabic characters after pre-processing steps, especially in the context of recognizing segmented characters from images of Hadeeth.

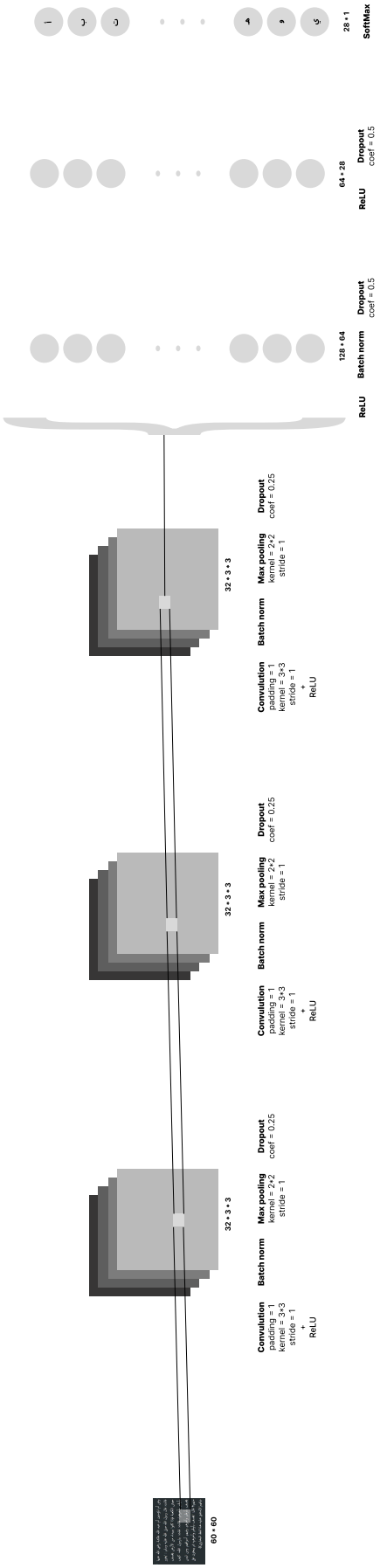

Fig. 1. The Model Architecture for Arabic Hadith Recognition

V. Implementation details

In this section, we outline the process followed to implement and evaluate a Convolutional Neural Network (CNN) architecture for the recognition of Arabic Hadeeth. The steps include selecting and implementing the model, training it, and documenting the experimental process, which consists of data preprocessing, model architecture decisions, hyperparameter selection, and the training process with evaluation results.

A. Data Preprocessing Steps

The dataset was preprocessed in several steps to prepare it for training:

- **Cropping Borders**: The images in the dataset often had borders or margins that were not relevant for the character recognition task. These were cropped to ensure that the focus was on the characters themselves.
- **Resizing**: All images were resized to a uniform dimension of 60x60 pixels to maintain consistency across the dataset. This step ensured that the model could process input images of a consistent size during training.
- **Converting to Grayscale**: Since the task focused on recognizing characters and the images were primarily black and white, we converted the images to grayscale. This reduced the dimensionality of the data and simplified the model, making it more efficient without sacrificing information necessary for recognition.

These preprocessing steps ensured that the model could focus on the relevant features of the Arabic characters without being distracted by irrelevant information like background colors or borders.

VI. Model Architecture Decisions

The model was built using a Convolutional Neural Network (CNN) architecture, which is well-suited for image classification tasks like character recognition. Here are the key decisions made during the architecture design:

A. Convolutional Layers

The model uses three convolutional layers with increasing numbers of filters (32, 64, and 128) to extract hierarchical features from the input images. These layers capture essential patterns in the Arabic characters, such as strokes, curves, and dots.

B. Batch Normalization

After each convolutional layer, batch normalization was applied to stabilize and accelerate the training process. This also helps improve the model's generalization.

C. Max-Pooling Layers

Max-pooling was used after each convolutional block to reduce the spatial dimensions of the feature maps while retaining important information. This step helps the model become invariant to small translations and distortions in the input images.

D. Dropout Layers

Dropout was applied to both convolutional and dense layers to prevent overfitting, especially since the dataset was relatively small.

E. Fully Connected Layers

After flattening the feature maps, fully connected layers (128 and 64 neurons) were added to allow the model to combine learned features and make abstract decisions about the Arabic characters.

F. Output Layer

The final output layer consists of 28 neurons, each representing one of the 28 Arabic characters, with a softmax activation function to output a probability distribution across all possible classes.

VII. Hyperparameter Selection

The model's performance was highly dependent on selecting the right hyperparameters. The following parameters were tuned:

A. Epochs

The number of training epochs was tested with both 50 and 100 epochs. It was found that 100 epochs yielded better performance, as the model was able to converge more effectively, leading to higher accuracy.

B. Batch Size

Various batch sizes were tested (5, 10, and 20). A batch size of 20 performed better, as it allowed the model to process more data in each iteration, which helped improve the generalization and stability of the learning process.

C. Filters/Kernels in Convolutional Layers

The number of filters used in each convolutional layer (32, 64, and 128) was chosen to balance computational efficiency and the ability to capture complex patterns in the input images. The model performed well with these values.

These hyperparameter selections were based on experimentation and previous research, ensuring that the model had the necessary capacity to handle the complexity of Arabic Hadeeth recognition.

Results and analysis

The training results indicate the model's learning progress across 100 epochs. Here's a summary of key points from the first 73 epochs:

Accuracy Progress

The training accuracy shows significant improvement from the beginning of training:

- Starting from 6.9% in epoch 1, it steadily increases, reaching over 96% by epoch 73.
- The model has shown notable learning, with a fairly consistent upward trend in training accuracy.

Validation Accuracy

Validation accuracy is also improving, reaching 96.23% by epoch 61 and stabilizing around this value in the subsequent epochs. This suggests the model is generalizing well and not overfitting despite its improving performance.

Loss Trends

Both training and validation loss show a steady decline:

- The training loss drops significantly from 4.32 in epoch 1 to values around 0.1 in later epochs.
- Validation loss also shows consistent reduction, with occasional minor increases, likely due to fluctuations in validation data or the model's regularization.

Overfitting Consideration

Although the validation accuracy is quite high, it's worth monitoring for overfitting. The accuracy starts to plateau, and the validation loss might fluctuate slightly in later epochs. However, the model maintains good generalization throughout the training process.

D. Prediction on Hadeeth Images

When the model was tested on images of Hadeeth, it produced good results overall, but it also made several mistakes. These mistakes were primarily related to the complexity of the data and the imperfections in the segmentation process. Since the dataset was created through segmentation, and the segmentation process was not flawless, some characters were misclassified, especially when they were close to each other or had overlapping strokes.

VIII. Challenges and Limitations

Despite the high accuracy, there are challenges that remain:

A. Segmentation Imperfections

The segmentation pipeline used to create the dataset was not perfect, leading to instances where characters were incorrectly split or merged. This caused errors in recognition, especially for characters with intricate shapes or that were close to each other in the original text.

B. Data Complexity

Arabic Hadeeth texts, especially in handwritten form, can be complex, with variations in handwriting style and character appearance. These variations can make it difficult for the model to generalize perfectly across different instances of the same character.

The model demonstrated promising results for Arabic Hadeeth recognition with an accuracy of 0.9556. However, the challenges associated with segmentation imperfections and the complexity of the handwritten Arabic text mean that the model could still benefit from further refinement. These improvements could include better segmentation techniques or additional data augmentation to handle handwriting variations more effectively.

The experimental process, from data preprocessing to hyperparameter selection, has been documented to provide insight into the methods and decisions that led to the current results.

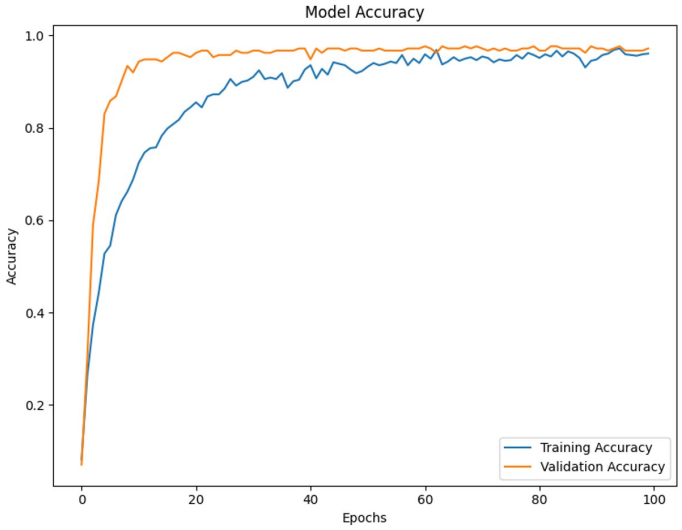

Fig. 2. Training And Validation Accuracy Progress

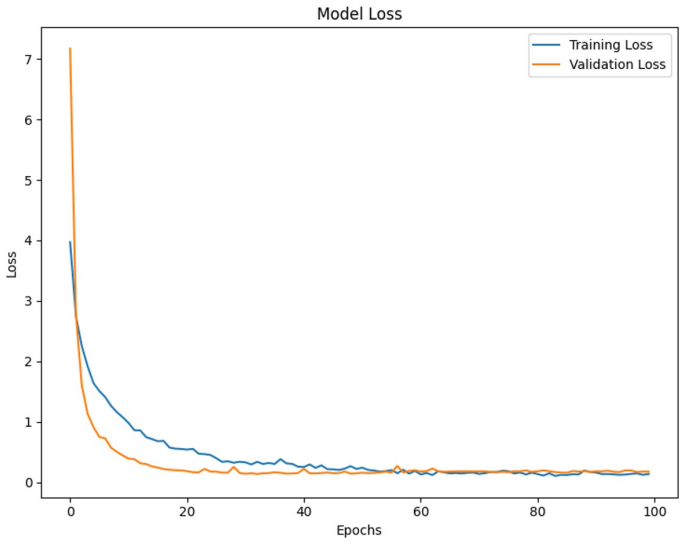

Fig. 3. Training And Validation Loss Progress

IX. Future Work

While the current implementation shows promising results, there are several areas where improvements can be made:

- **Data Augmentation:** To further enhance model generalization, incorporating more data augmentation techniques such as rotation, scaling, and skewing can help simulate real-world variances in handwriting and text presentation.
- **Advanced Model Architectures:** Future work can involve exploring more sophisticated models, such as transformer-based architectures, to capture more complex relationships in the text.
- **Contextual Understanding:** Enhancing the system to not only recognize characters but also to understand the context of the text (e.g., combining recognized characters into accurate word and sentence structures) could improve

the overall performance and usability of the system.

- **Multi-language Support:** Expanding the model to handle other languages, such as different styles of Arabic or even non-Arabic scripts, could make the system more versatile.
- **Real-time Recognition:** Implementing the model for real-time Hadeeth and Ayat recognition, such as through mobile applications, would increase its practical utility.
- **Model Optimization:** Future research could focus on optimizing the model for faster inference times, enabling the system to handle large datasets more efficiently, particularly for applications requiring real-time processing.

These improvements will help push the boundaries of handwritten Arabic text recognition, enhancing both its accuracy and applicability in various domains.

X. Conclusions

The Hadeeth and Ayat recognition project successfully demonstrated the potential of deep learning techniques for recognizing Arabic text in complex formats. By leveraging segmentation methods to detect individual characters and classifying them using a custom-trained model, we achieved an efficient process for recognizing Hadeeth and Ayat. Key conclusions include:

- **Effective Segmentation:** The segmentation approach enabled accurate detection of lines, words, and characters, which was crucial for extracting meaningful features from the dataset.
- **High Recognition Accuracy:** The model showed promising results in character classification, achieving over 95% accuracy for most of the training epochs, which indicates good learning progress.
- **Good Generalization:** The validation accuracy remained stable, and the model avoided significant overfitting, confirming that it generalized well to unseen data.
- **Model Performance:** Despite the complexity of Arabic calligraphy, the model achieved high performance in character recognition, making it well-suited for real-world applications such as Hadeeth and Ayat recognition.

References

Websites

- Sunnah.com. *Riyad-us-Saliheen*. Accessed: January 5, 2025.
  Available at: https://sunnah.com/riyadussalihin/introduction

GitHub Repositories

- GitHub Repository. *Arabic-OCR*. Accessed: January 5, 2025.
  Available at: https://github.com/HusseinYoussef/Arabic-OCR

Kaggle Notebooks

- Kaggle Notebook. *Arabic Character Recognition with CNN*. Accessed: January 5, 2025.
  Available at: https://www.kaggle.com/code/aymenmouffok/arabic-character-recognition-with-cnn