People's Democratic Republic of Algeria

Ministry of higher education and scientific research
KASDI MERBAH UNIVERSITY - OUARGLA

Faculty of New Technologies of Information and telecommunication

Department of Computer Science and Information Technologies

Master Thesis

Domain:Computer Science

Field: Computer Science Fundamentals

Hybrid framework for Arabic handwriting recognition using deep learning and NLP

Presented by:: CHINE KAWTHER

Evaluation Date:12-06-2025

the Jury:

Dr.Benchabana Ayoub	**Supervisor**	**UKM Ouargla**
Dr.Azzaoui Nadjet	**President**	**UKM Ouargla**
Dr.Aicha Korichi	**Examiner**	**UKM Ouargla**

Academic Year: 2024/2025

Acknowledgements

First and foremost, I thank Allah, the Most Gracious and the Most Merciful, for granting me the strength, patience, and guidance throughout the journey of completing this work.

I would also like to express my deepest gratitude to my beloved parents, whose unwavering support, encouragement, and prayers have been the foundation of my academic and personal growth.

My sincere appreciation goes to my supervisor, Mr. Ben Chabane Ayoub, for his valuable guidance, continuous support, and constructive feedback throughout this research. His insights and dedication played a crucial role in shaping the outcome of this project.

Dedication

To the One who never ceases to bless me — Allah, the source of all strength and wisdom.

To my dear parents, whose love, sacrifices, and unwavering support have been my greatest motivation.

To my respected supervisor, Mr. Ben Chabane Ayoub, for his guidance and encouragement.

I dedicate this work to all of you, with deepest gratitude and respect.

Abstract

The scientific community continues to face challenges in developing effective solutions for recognizing handwritten Arabic words, primarily due to limited research addressing the unique complexities of the Arabic language, such as its cursive nature and the variability in handwriting styles across individuals and over time. This study proposes a hybrid framework that integrates deep learning and natural language processing to enhance handwritten Arabic word recognition. The model combines Convolutional Neural Networks (CNN) for extracting visual features with Long Short-Term Memory (LSTM) networks for processing the features extracted from CNN, further improved by an N-gram-based correction mechanism. The framework was trained and evaluated using the Arabic Handwritten Database (AHDB), comprising 6,615 images of 63 distinct words written by 100 different writers. While not fully overcoming the inherent challenges of handwritten Arabic recognition, the proposed approach yielded promising results, achieving an accuracy of 84%, with additional improvements from the correction mechanism, demonstrating its potential to advance the field and laying the groundwork for future enhancements.
Keywords: Arabic word recognition, Deep learning, CNN, LSTM, N-gram

Résumé

La communauté scientifique continue de faire face à des défis dans le développement de solutions efficaces pour la reconnaissance des mots arabes manuscrits, principalement en raison du nombre limité de recherches abordant les complexités uniques de la langue arabe, telles que sa nature cursive et la variabilité des styles d'écriture manuscrite entre les individus et au fil du temps. Cette étude propose un cadre hybride qui intègre l'apprentissage profond et le traitement du langage naturel pour améliorer la reconnaissance des mots arabes manuscrits. Le modèle combine des réseaux neuronaux convolutionnels (CNN) pour extraire les caractéristiques visuelles avec des réseaux à mémoire à long et court terme (LSTM) pour traiter les caractéristiques extraites par le CNN, avec une amélioration supplémentaire grâce à un mécanisme de correction basé sur un modèle N-gram. Le cadre a été entraîné et évalué en utilisant la base de données de l'écriture manuscrite arabe (AHDB), comprenant 6 615 images de 63 mots distincts écrits par 100 écrivains différents. Bien qu'il n'ait pas complètement surmonté les défis inhérents à la reconnaissance de l'écriture manuscrite arabe, l'approche proposée a donné des résultats prometteurs, atteignant une précision de 84 %, avec des améliorations supplémentaires grâce au mécanisme de correction, démontrant ainsi son potentiel pour faire avancer le domaine et poser les bases d'améliorations futures.

Mots clés: Reconnaissance des mots arabes, Apprentissage profond, Réseaux neuronaux convolutifs, Réseaux de mémoire à long et court terme,N-gramme

ملخص

تواصل الأوساط العلمية مواجهة تحديات في تطوير حلول فعالة للتعرف على الكلمات العربية المكتوبة يدويًا، ويرجع ذلك بشكل رئيسي إلى محدودية الأبحاث التي تتناول التعقيدات الفريدة للغة العربية، مثل طبيعتها المتصلة وتنوع أنماط الكتابة اليدوية بين الأفراد وعبر الزمن. تقترح هذه الدراسة إطارًا هجينًا يدمج التعلم العميق ومعالجة اللغة الطبيعية لتحسين التعرف على الكلمات العربية المكتوبة يدويًا. يجمع النموذج بين الشبكات العصبية الالتفافية (CNN) لاستخلاص الميزات البصرية وشبكات الذاكرة طويلة المدى قصيرة الأجل (LSTM)لمعالجة الميزات المستخرجة منCNN ، مع تحسين إضافي باستخدام آلية تصحيح تعتمد على نموذج .N-gram تم تدريب الإطار وتقييمه باستخدام قاعدة بيانات الكتابة اليدوية العربية(AHDB) ، التي تتضمن 6,615 صورة لـ63 كلمة متميزة كتبها 100 كاتب مختلف. على الرغم من عدم التغلب الكامل على التحديات المتأصلة في التعرف على الكتابة اليدوية العربية، أظهر النهج المقترح نتائج واعدة، محققًا دقة بنسبة 84%، مع تحسينات إضافية من آلية التصحيح، مما يبرز إمكاناته لتطوير المجال ويؤسس الأرضية لتحسينات مستقبلية.

الكلمات المفتاحية:
التعرف على الكلمات العربية, التعلم العميق, الشبكات العصبية التلافيفية CNN, الشبكات طويلة الدمى قصيرة الذاكرة N-gram, LSTM

Table of contents

List of Figures

List of Tables

Chapter 1

General Introduction:

1.1 Introduction:

The field of Artificial Intelligence (AI) and Machine Learning (ML) has experienced rapid advancements, profoundly impacting various sectors, including science, technology, and industry.[1] Among these advancements , deep learning (DL) has emerged as a powerful tool, particularly in computer vision applications such as object detection, image classification, and video analysis [2]. More recently, deep learning has been successfully applied to natural language processing (NLP)[3], leading to significant progress in areas like machine translation, speech recognition, and text generation [4].

One of the emerging applications of deep learning and NLP is handwriting recognition, which is particularly challenging for scripts with complex structures, such as Arabic. Handwriting word recognition (HWR) has become one of the most intriguing yet challenging areas in computer vision and pattern recognition[5]. Automatic handwriting recognition refers to a system's ability to accurately interpret and understand handwritten input [6]. The main objective of such systems is to correctly identify input characters or images, which are then analyzed and processed through various automated operations. These systems are applied in multiple domains to detect and interpret handwritten text in different formats and writing styles[7].

Despite the widespread use of the Arabic script, developing a robust handwriting recognition system continues to present significant challenges. This difficulty is primarily due to the inherently cursive nature of Arabic script,variability in personal handwriting styles, and the contextual shape changes of Arabic letters [8] . Moreover, Arabic poses unique challenges for natural language processing (NLP) due to several inherent features, while significant progress has been made in handwriting recognition for Latin-based scripts, research on Arabic handwriting recognition (AHR) remains limited [9].

Although some studies have attempted to address the problem of Arabic handwriting recognition, many of them focus solely on visual features without considering the linguistic context, or do not effectively integrate between visual and semantic aspects. This highlights the need for more comprehensive solutions that leverage both the visual and semantic aspects of the texts.

Based on the above, this research aims to propose a hybrid framework that combines deep learning techniques with natural language processing (NLP) to improve the accuracy and robustness of Arabic handwriting recognition systems. This framework addresses both the visual and semantic challenges associated with Arabic script. It consists of multiple stages, including image pre-processing and feature extraction, both of which contribute to improving recognition accuracy and reducing error rates, particularly in cursive texts with diverse handwriting styles.

1.2 Problematic:

1.2.1 Linguistic Complexity of Arabic Script:

The recognition of Arabic handwritten characters remains a challenging problem due to the unique linguistic and structural features of the Arabic language .[10]

- **Several aspects contribute to this complexity:**

  Arabic is inherently a cursive script, where most characters are connected within a word. This nature complicates the process of segmenting handwritten text into individual characters and reduces recognition accuracy [11].

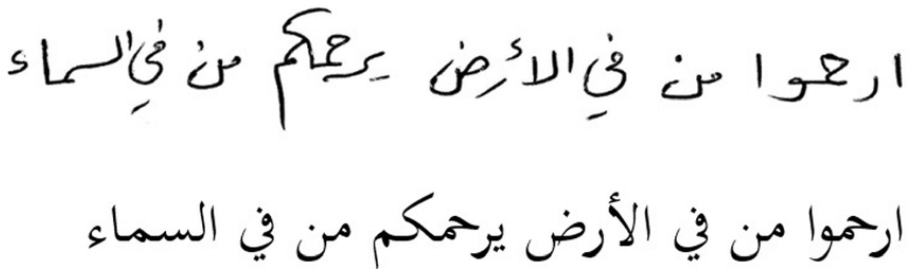


Figure 1.1: Example of cursive nature in Arabic handwritten script.[12]

- Some Arabic characters share highly similar shapes, with differentiation primarily based on the position and number of dots placed above or below the character body. For example, the letters Thaa ث, Taa ت, and Baa ب have an identical base structure but are distinguished solely by the number and placement of their dots[13].

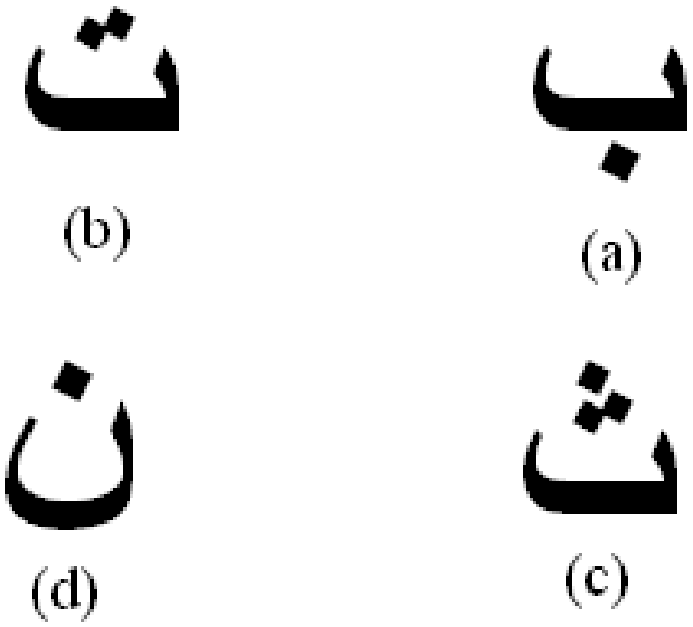


Figure 1.2: Example of Arabic Characters with Similar Base Structures Differentiated by Dots.[14]

- Arabic characters are linked along an imaginary reference line called the baseline, which adds further complexity to the segmentation and recognition processes.[15]

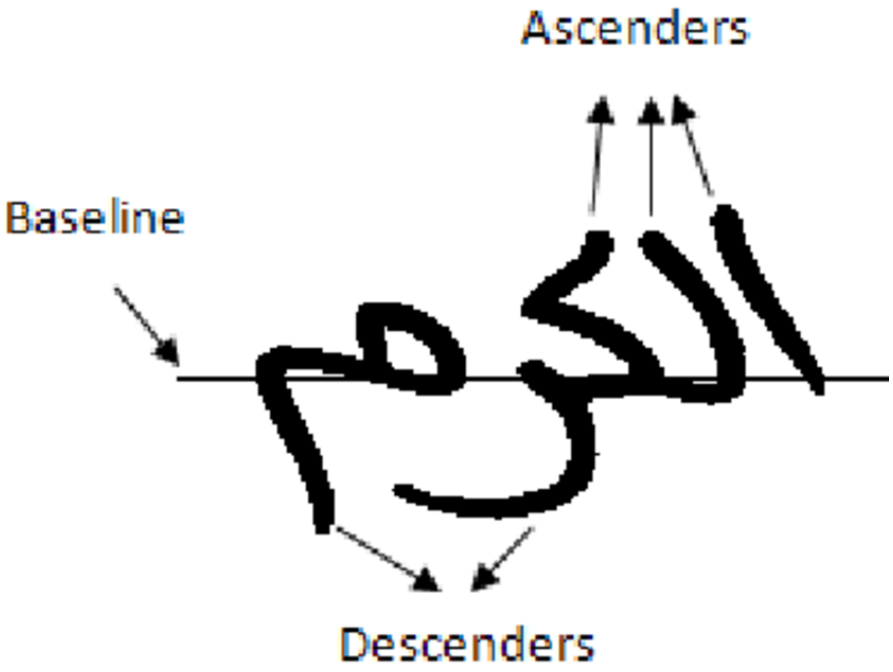


Figure 1.3: Baseline, Ascenders and Descenders as shown in a Word.[16]

- Each Arabic letter can have two to four distinct shapes, depending on its position within

a word-whether at the beginning, middle, end, or in isolation.[17]

Figure 1.4: The letter ك in its three positions: the beginning, middle, end of the word.

1.2.2 Scarcity of Comprehensive and Open Datasets:

- The limited availability of large-scale, publicly accessible datasets for Arabic handwriting hinders the training of models and limits their performance improvement.[15]

1.2.3 The impact of diverse handwriting styles on recognition systems:

- **Variability in individual handwriting styles:** Individual differences in handwriting styles pose significant challenges to accurate character recognition, as each person's unique writing characteristics complicate the identification process.[17]
- **Character Connectivity:** In cases where characters lack sufficient spacing, particularly in cursive scripts, makes it difficult to segment and recognize individual characters accurately, hindering effective segmentation and recognition.[17] For example, the same Arabic letter, such as سين or " may appear completely differently depending on the writer's style or writing speed, complicating the process of feature extraction and classification.

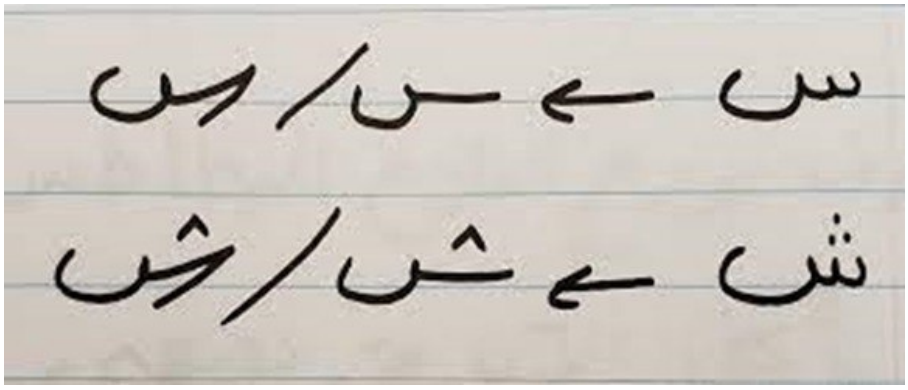


Figure 1.5: Variation in the shape of the same Arabic letter based on handwriting style and speed..[18]

Previous studies have attempted to address these challenges using various approaches. For example:

- Biadsy et al. (2011) [19] proposed a system for Arabic handwriting recognition without the need to segment words into individual letters, relying on Hidden Markov Models (HMM) with effective handling of delayed strokes. Training was conducted at the letter level, while recognition was performed at the word-part level. The system achieved high accuracy, exceeding 98% in writer-dependent cases and 94% in writer-independent ones.

  However, the study did not incorporate contextual linguistic analysis (NLP) after recognition, which may limit the system's ability to distinguish between visually similar words, especially in the absence of diacritics.

- Dreuw and colleagues [20] proposed an offline Arabic handwriting recognition system based on Hidden Markov Models (HMM), introducing explicit white-space models between characters or pieces of Arabic words (PAWs). Unlike systems that implicitly embed white-spaces within character models, they used a separate white-space character model, which significantly improved recognition accuracy. The approach also included model length adaptation and the use of virtual training samples to enhance model robustness. The system achieved strong performance on the IFN/ENIT database, outperforming previous systems in various benchmark evaluations.

  However, the system does not incorporate contextual or lexical analysis (e.g., NLP or language modeling), which limits its effectiveness in cases where visually similar words require semantic understanding for accurate recognition.

- In their study [21], Altwaijry and Al-Turaiki developed an automatic handwriting recognition model trained on the Hijja and Arabic Handwritten Characters (AHCD) datasets.

The model achieved an accuracy of 97% on the AHCD dataset and 88% on the Hijja dataset.

- El-Sawy et al.[22] proposed a deep learning architecture for recognizing handwritten Arabic characters. The system was trained on the Arabic Handwritten Character Dataset (AHCD), comprising 16,800 handwritten Arabic characters. They utilized a Convolutional Neural Network (CNN) model, implementing various optimization techniques to enhance performance. The model achieved an accuracy of 94.9% on the AHCD dataset.

Despite previous research efforts that have addressed some of these challenges using statistical models or conventional neural networks, the lack of integration between visual analysis and contextual linguistic processing remains a significant research gap. Therefore, there is a pressing need for a hybrid framework that combines Deep Learning techniques with Natural Language Processing (NLP) to enhance the accuracy of Arabic handwriting recognition, particularly in visually complex and cursive scripts. This research aims to fill this gap by proposing a comprehensive solution that addresses both the structural and semantic aspects of handwritten Arabic text.

1.3 Motivation:

The selection of this research topic is driven by several scientific, practical, and personal motivations, in addition to the potential impact of developing an efficient system for Arabic handwriting recognition using artificial intelligence..

1.3.1 Scientific Motivation handwriting arabic:

The existence of unresolved research challenges in Arabic handwriting recognition, such as difficulty in character segmentation, variations in handwriting styles, and handling diacritical marks.

The need to develop more accurate Al techniques capable of better understanding Arabic script

1.3.2 Practical Motivation:

The importance of converting handwritten text into digital text for use in electronic archiving, governmental institutions, and banking sectors.

Enhancing processing speed and efficiency of documents while reducing the need for manual text entry, thereby saving time and effort.

Supporting Optical Character Recognition (OCR) technologies, which currently suffer from low accuracy in Arabic text recognition.

1.3.3 Personal Motivation:

A strong interest in deep learning and natural language processing (NLP) and exploring their integration to solve complex linguistic challenges. A motivation to contribute toward the development of cutting-edge technological solutions that benefit both the academic and industrial communities.

1.3.4 Potential Impact of the Research:

If successful, this research will enhance Al-based Arabic handwriting recognition systems, facilitating their application in education, digital archiving, and historical manuscript analysis. Providing a precise and efficient tool that can be utilized in various fields such as translation, scientific research, and handling legal documents.

- **Bank Check Recognition:** Handwriting recognition helps scan checks and convert their content. into digital text, enabling signature verification and real-time check processing, which enhances the speed of banking transactions.
- **Healthcare Sector:** Contributes to the digitization of medical records and patient history, facilitating access to past diagnoses, test results, and insurance records while reducing reliance on paper files.
- **Legal Sector:** Used for digitizing legal documents such as certificates, court rulings, contracts, and wills, making it easier to store them in searchable databases instead of relying on traditional paper records.

1.4 Contributions:

This research aims to provide scientific and applied contributions in the field of Arabic handwriting recognition, through:

Developing a new hybrid framework that integrates Deep Learning (DL) and Natural Language Processing (NLP) to enhance the accuracy of Arabic handwritten text recognition.

Analyzing the impact of NLP on improving handwriting recognition by incorporating NLP techniques to correct errors and enhance contextual understanding of texts. Enhancing the understanding of Arabic handwriting characteristics, particularly regarding connected letters, diacritical marks, and shape variations based on letter positioning within words.

Developing an advanced Al model capable of handling the unique challenges of the Arabic language, such as connected letters and variations in handwriting styles.

1.5 Thesis Structure

- **Chapter 1: General Introduction**

  This chapter provides a general background on the research topic, highlighting its significance, the addressed problem, the motivations for its selection, and the expected contributions.

- **Chapter 2: Work Background:** This chapter aims to present the scientific and technical context of the research, covering fundamental concepts, applied techniques, and previous studies.

- **Chapter 3: Methods Used in the Model:** This chapter presents a detailed overview of the components used to build the proposed model, including Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and a linguistic correction mechanism based on the N-gram model.

- **Chapter 4: Implementation and Results:**

  In this chapter, the results of the experiments conducted using the proposed model are presented and analyzed, along with a comparison of its performance with other models

Chapter 2

Work Background

2.1 Introduction :

This chapter provides the scientific and technical foundation of our research. It presents the essential theoretical concepts, related works, and technologies relevant to the field of Arabic handwriting recognition. By exploring fundamental elements such as the characteristics of Arabic handwriting, digital image processing, Natural Language Processing (NLP), and deep learning techniques, we establish a clear understanding of the research domain. In addition, this chapter reviews existing studies and systems that have addressed similar challenges, highlighting their contributions and limitations. The goal is to provide the reader with the necessary background to understand the motivations and methodologies adopted in this study.

2.2 Arabic Handwriting:

Handwriting is one of the oldest and most fundamental forms of communication, continuously evolving in response to cultural and technological advancements. Before the advent of writing, verbal communication and sign language were the primary means of conveying information. The development of writing enabled the documentation of history, events, culture, literature, law, science, mathematics, and various other domains. Writing can be defined as a structured system of standardized symbols governed by specific rules to represent and convey ideas.[23]

Arabic writing is a complex system consisting of a set of letters that historically evolved from the Aramaic alphabet, with the earliest recorded document dating back to 512 AD. As the Arabic language expanded, diacritical marks were added to some letters in the 7th century AD to distinguish different phonetic sounds, leading to the emergence of multiple forms of the same letter. Today, this writing system is used by more than 100 million people in over 20 countries, in both printed and handwritten texts.[24]

On the other hand, Arabic handwriting is a specific form of Arabic writing that relies on fine motor skills, requiring coordination among the human visual, cognitive, and motor systems. The quality of handwriting is influenced by several factors [25], such as motor control, writing speed, and precision in letter formation. Unlike printed text, which follows standardized typographical rules, handwriting varies significantly from one individual to another, making the automatic recognition of handwritten Arabic text more challenging than recognizing printed text.

Figure 2.1: Handwritten vs. Machine-Generated Arabic Writing.[26]

2.2.1 Characteristics And Challenges Of Arabic Handwriting:

- The Arabic script is inherently cursive, meaning that the characters within a word are connected along an imaginary horizontal guideline known as the baseline. Additionally, certain strokes extend above and below this baseline, commonly referred to as ascenders and descenders [13].

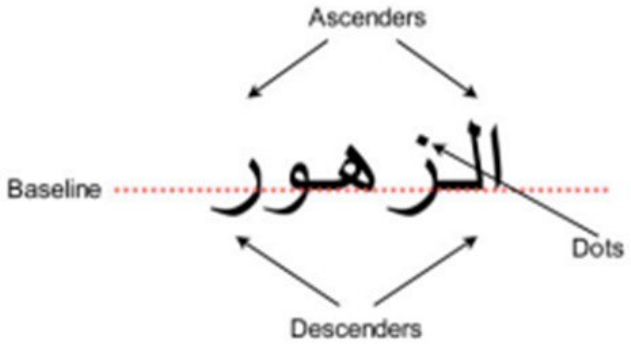


Figure 2.2: Example of baseline, ascenders, and descenders in arabic script[27].

- Words in Arabic are written in horizontal lines from right to left, while numerals follow a left-to-right writing direction.[28]

Figure 2.3: Arabic writing direction.[29]

- Most Arabic letters change their shape based on their position within a word, whether they appear at the beginning, middle, end, or in isolation.[28]

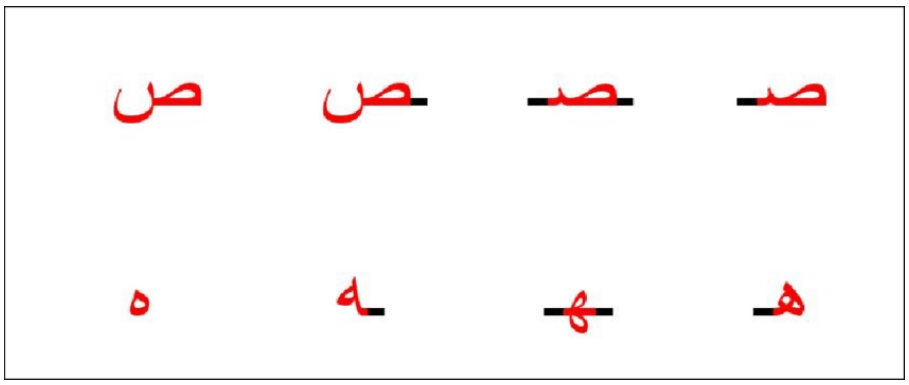


Figure 2.4: The change of letter shape with position in the word.[30]

- Irregular spacing between words and characters, along with curved or non-linear text lines.[31]
- In certain Arabic calligraphic styles, such as the Diwani script, characters within a word may overlap or intertwine, resulting in complex and highly stylized word formations.[28]

Figure 2.5: Some overlapping letters in Arabic fonts.[29]

2.2.2 Types of Handwriting Recognition:

Handwriting recognition is a specialized area within pattern recognition and document image analysis that focuses on identifying and interpreting handwritten content. This content can be either textual or graphical, including elements such as musical notation, mathematical expressions, sketches, and diagrams.[32]

Handwriting recognition systems are generally classified into two main types: online handwriting recognition and offline handwriting recognition.

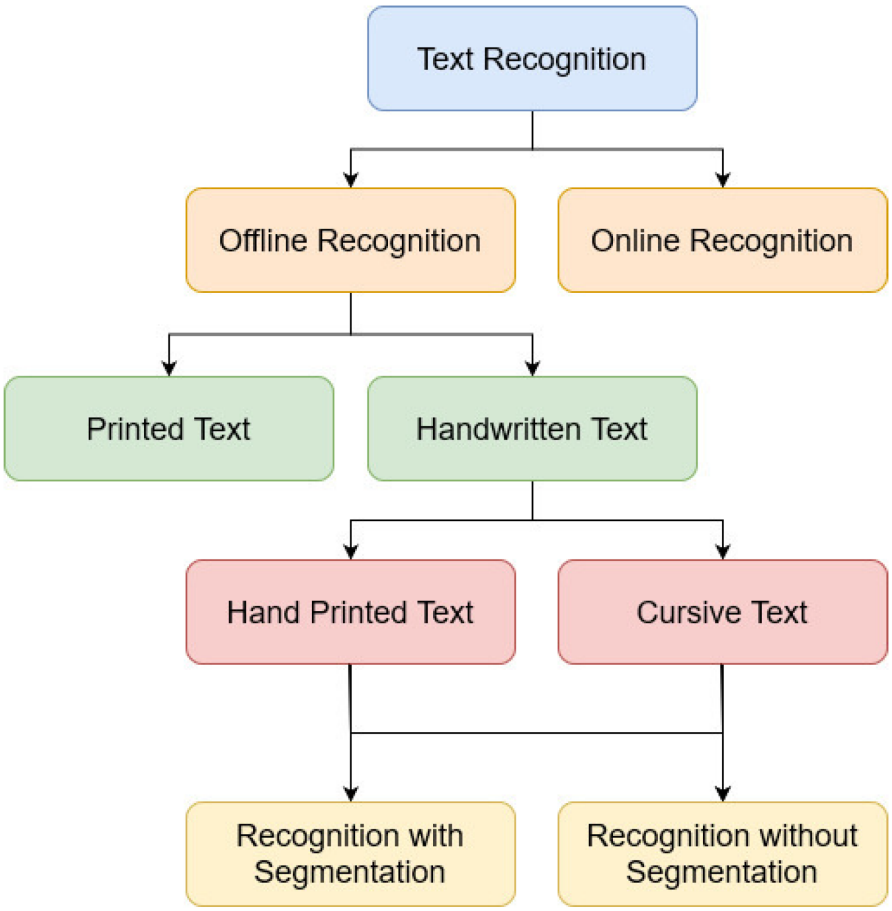


Figure 2.6: Classification of handwritten text recognition systems.[33]

2.2.2.1 Offline handwriting recognition:

Offline handwriting recognition is an advanced technique aimed at identifying handwritten text from images. Due to the variability in individual writing styles, accurately recognizing handwritten characters and digits presents a significant challenge [34]. The process of recognizing words from images is a complex task, as each person has a unique writing style, making it difficult to achieve correct recognition of handwritten characters and digits.[34]

Offline handwriting recognition is generally considered more challenging than online handwriting recognition. This is because, in offline recognition, the system works with static images of handwritten text, without any information about the process of writing[35], Offline character recognition, a subset of this technique, focuses on identifying the final form of al character or digit directly from the image. In this approach, there is no knowledge of how the writer composed the character; recognition relies solely on the final written form, which adds an additional layer of complexity due to the diverse nature of individual writing styles.[36]

It has several applications, such as:

- **Mail sorting:** recognizing addresses or handwritten text on envelopes.

- **Bank check reading:** identifying handwritten text and numbers on checks.
- **Transcription of books and handwritten notes:** converting handwritten content into digital text.[37]

2.2.2.2 On-line handwriting recognition:

On-line handwriting recognition refers to the process in which a system identifies handwritten text as the user writes. This method is often referred to as real-time or dynamic handwriting recognition, as the recognition occurs simultaneously with the writing process .[38]

In online handwriting recognition, the trajectory of the pen tip is captured continuously by recording its position on the writing surface at regular time intervals. The primary objective is to translate this sequence of pen movements into a corresponding sequence of written words.[39]

Online handwriting recognition technology has a wide range of applications, such as:

- **Pen input devices:** Recognizing handwriting in real time on digital tablets and drawing boards.[37]
- **Personal digital assistants (PDAs):** Allowing users to take notes and issue handwritten commands.[37]
- **Smartphones:** Supporting handwriting recognition through stylus input for texting, searching, or interacting with applications.[37]
- **Computer-assisted instruction:** Enabling students to write answers or notes by hand and receive immediate feedback.[37]
- **Interactive learning tools:** Facilitating dynamic, handwriting-based assessments and exercises.[37]

2.3 Digital Image:

2.3.1 Introduction to Digital image:

A digital image is a finite set of pixels, each encoded by bits derived from a sampling process. Each pixel is represented using 8 bits, allowing color values to range from 0 to 255. Each pixel contains three primary components: Red (R), Green (G), and Blue (B), with 8 bits allocated to each component to represent the intensity of each color in the RGB color model. This representation forms a vector that characterizes the intensity of each color within the RGB space. Consequently, the digital image can be viewed as a matrix of numbers that represent the color values for each pixel.[40]

2.3.2 definition of Digital Image (Mathematical Perspective):

An image can be defined as a two-dimensional function, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When the values of x, y, and the intensity of f are all finite and discrete, the image is referred to as a digital image. The field of digital image processing refers to processing digital images using a digital computer. [41]

2.3.3 Digital Image Representation and the Digitization Process

The process of converting an image from analog to digital representation is a fundamental step in digital image processing. This process, known as digitization, involves two main stages: sampling and quantization. It can be summarized as follows:

- **Analog Representation of the Image:**

  The original image is represented as a continuous function in two-dimensional space: a(x,y)a(x, y)a(x,y).

  This function may depend on other variables such as depth (z), color λ, and time (t), but in most applications, it is processed as a static 2D image.[42]

- **Spatial Sampling:**

  The continuous image is divided into a grid of rows and columns to define a finite number of spatial coordinates.

The intersection of each row and column defines a pixel.

Each point on the grid is mapped to integer coordinates $[m, n]$, where $m = 0 \dots M - 1$ and $n = 0 \dots N - 1$.[42]

- **Quantization:**

  A numerical value is assigned to each pixel, representing the intensity (in grayscale images) or color (in colored images) at that location.

  In grayscale images, this value typically ranges from 0 to 255 (8 bits).

  In color images, each pixel contains three values (R, G, B), each represented using 8 bits to define the intensity of the corresponding color channel.[42]

- **Result of Digitization:** The digital image becomes a two-dimensional matrix of numbers representing the intensity or color values at each pixel location.

  This matrix is then used for further analysis and processing using digital computers.[42]

2.3.4 Digital Image Processing Stages:

Digital image processing encompasses a sequence of systematic steps that transform a raw image into a form suitable for analysis and interpretation. These steps can be broadly categorized based on whether the process outputs an image or extracts specific features from the image. The primary stages are as follows:

- **Image Acquisition:**

  This initial phase involves capturing the digital image through devices such as scanners or digital cameras. It may also include preliminary preprocessing (e.g., scaling) to ensure that the image is in an appropriate format for further analysis.[43]

- **Image Enhancement:** In this stage, the quality of the captured image is improved. Techniques are applied to adjust brightness, contrast, and sharpness, thereby enhancing the visibility of critical details and making the image more suitable for subsequent processing tasks.[44]

- **Image Restoration:** Unlike enhancement—which is subjective—restoration focuses on objectively correcting image degradations. This process uses mathematical and probabilistic models to remove artifacts such as blur or noise, thereby recovering the original appearance of the image as closely as possible.[44]

- **Color Processing:** Color processing involves manipulating color information to extract useful features or convert the image into grayscale or binary formats. This step simplifies analysis when structural features are more important than color information.[43]

- **Transforms:** Mathematical transformations (e.g., Fourier or Wavelet transforms) are employed to extract essential frequency and spatial features from the image. These transforms facilitate better compression, enhancement, and recognition by representing the image at various levels of resolution.[43]

- **Compression and Watermarking:** This step focuses on reducing the storage requirements or transmission bandwidth of the image while preserving its critical information. Although not always essential for recognition, compression plays a significant role in managing large datasets.Watermarking is used for authentication and copyright protection. It has two major approaches: (a) Lossless Compression and (b) Lossy Compression.[45]

- **Morphological Processing:** Morphological operations are applied to extract and analyze the shape and structure of the image components. This stage is crucial for separating individual elements and removing noise or irrelevant details from the image.[46]

- **Segmentation:** Image segmentation is a fundamental step in image processing that aims to divide an image into distinct regions or segments with homogeneous characteristics. This process is used to simplify the representation of an image or transform it into a more meaningful and analyzable form. By focusing on the most important parts of the image and ignoring irrelevant elements, segmentation enhances the efficiency and accuracy of automated systems in image analysis and interpretation. [47]

- **Feature Extraction:** In this phase, key features of the segmented regions—such as edges, curves, and intersections—are detected and quantified. These features serve as the basis for further classification tasks.[48]

- **Pattern Classification:** With the extracted features, machine learning or deep learning models (e.g., convolutional neural networks) are utilized to classify and identify patterns within the image. This classification assigns a label or identity to each recognized component.[43]

- **Knowledge Base:** A repository of prior knowledge, such as language models or character databases, is integrated into the system. This knowledge base supports and refines the

recognition process, ensuring that the final interpretation aligns with established data and linguistic patterns.[43]

- Example: Application to Arabic Handwriting Recognition

Stage	**Application to Arabic Handwriting Recognition**
Image Acquisition	Capturing an image of handwritten Arabic text via scanner or camera.
Image Enhancement	Enhancing contrast or removing noise to make the characters clearer.
Image Restoration	Correcting distortions caused by poor lighting or camera shake.
Color Processing	Usually converts the image to grayscale or binary to simplify processing.
Transforms	Used to extract features or improve the image using techniques like Wavelets.
Compression and Watermarking	Not directly essential, but useful for reducing the size of stored data.
Morphological Processing	Used to distinguish character boundaries or remove impurities such as isolated dots.
Segmentation	Very important! To divide the image into lines, words, and individual characters.
Feature Extraction	Extracting character features (such as curves or lengths) for recognition.
Pattern Classification	The core of the project! Classifying each character to its numeric or textual equivalent using algorithms like CNN.
Knowledge Base	Can include a database of characters, dictionaries, or language models to improve recognition.

Table 2.1: Organized Table for Arabic Handwriting Recognition Process

2.3.5 Key Challenges and Limitations in Digital Image Processing:

Digital image processing technologies face a number of challenges that affect their accuracy and efficiency. The most notable challenges include:

- **Nontrivial Issues:** Image processing involves multiple stages such as filtering, registration, classification, and merging. These steps requireconsiderable time and computational effort. Dealing with images that are visually similar but belong to different categories adds complexity and can significantly delay further processing steps.[49]

- **Limited Accuracy:** Achieving 100% accuracy in image processing is extremely challenging, especially when handling low-quality images or unclear handwritten texts. This limitation directly affects the reliability and acceptability of the results.[49]

- **Hard Coded Solutions:**

  Embedding data directly into the program's source code limits flexibility. It makes it difficult to adapt or improve algorithms, as hard coding restricts the ability to generalize and modify processing methods efficiently.[49]

- **Ongoing Need for Research and Development:** The field of image processing requires continuous innovation to address emerging challenges. This involves identifying weaknesses in existing techniques and developing smarter, more efficient algorithms.19

- **High Demand for Storage and Processing Power:** Digital image processing consumes large amounts of memory and computational resources. Progress in this field heavily depends on advancements in digital computers and supporting technologies such as data storage, display systems, and transmission infrastructure.[50]

- **Environmental Influence on Image Quality:** Environmental factors such as lighting, noise, and interference can degrade the quality of captured images, leading to less accurate processing outcomes.[50]

- **Data Redundancy:** Digital images often exhibit several types of redundancy, including spatial, spectral, and inter-pixel redundancy. These redundancies necessitate efficient compression techniques to optimize performance and reduce resource usage.[50]

- **Difficulty in Segmenting Complex Images:** Segmenting nontrivial or complex images remains one of the most difficult tasks in digital image processing due to overlapping elements or unclear features.[50]

2.4 Deep Learning:

2.4.1 Definition of deep learning:

Deep learning is a specialized subset of machine learning that employs artificial neural networks composed of multiple hierarchical layers—hence the term "deep"—to learn and model complex patterns within data. Inspired by the structure and function of the human brain, deep learning enables systems to automatically extract high-level features from raw inputs and perform sophisticated tasks such as image recognition, natural language understanding, and speech processing . [51]

It encompasses a family of algorithms designed to learn multiple levels of abstraction, where higher-level representations are built upon lower-level ones, forming what is known as a deep architecture. These models can operate in both supervised and unsupervised settings for tasks such as feature extraction, transformation, and pattern classification. Furthermore, deep learning contributes to the broader goal of artificial intelligence by facilitating the discovery of more effective data representations across various domains, including vision, speech, and text.[52]

2.4.2 Deep learning characteristics:

One of the main limitations of traditional machine learning methods lies in their heavy reliance on manual feature extraction from raw data. This process requires significant human effort and may not be effective when dealing with unstructured data.This is where deep learning (DL) comes in. It is capable of handling raw data such as images, texts, and audios, and can learn directly from them without the need for prior transformation. DL is based on artificial neural networks (ANNs), which are inspired by the human brain in how they analyze and understand information.Unlike shallow learning methods, which are composed of a limited number of layers, deep learning consists of multiple interconnected layers—forming deep neural networks. This allows DL models to learn representations at multiple levels of abstraction, enabling them to model highly complex functions more effectively.[53]

2.4.3 Deep Learning Approaches

- **Supervised Learning:**

  Supervised learning is one of the fundamental approaches in deep learning. It involves constructing a mathematical model based on a dataset that includes both input variables

XXX and target outputs YYY. This approach teaches the model to learn the relationship between inputs and outputs using an algorithm that approximates the mapping function Y=f(X)Y = f(X)Y=f(X).

During the training process, the model compares its predicted outputs with the actual target values to iteratively adjust and reduce prediction errors. Learning continues until the model achieves satisfactory accuracy across all input samples.

Typical tasks in supervised learning include regression, classification, and prediction. In deep learning, commonly used models include deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs).[54]

- **Unsupervised Learning:**

  In unsupervised learning, the primary task is to uncover hidden relationships, structures, or associations within the provided data without relying on predefined outputs. Similar samples are grouped into clusters based on similarity or distance measures, which facilitates a deeper understanding of the data and the underlying processes generating it. Among the commonly used techniques are those that rely on calculating distances between samples to form clusters, or extracting highly discriminative features that allow for better data separation.

  In some cases, supervised learning mechanisms can be indirectly applied, as seen in anomaly detection. Here, the model is trained using only normal samples, enabling it to later recognize unfamiliar patterns as anomalies.[55]

- **Semi-Supervised Deep Learning:**

  In recent years, semi-supervised deep learning has emerged as an exciting new research direction in deep learning. These methods address situations where only a few labeled training examples are available, accompanied by a significant number of unlabeled samples. In such scenarios, semi-supervised learning (SSL) methods are particularly applicable to real-world applications where unlabeled data is readily available and easy to acquire, while labeled instances are often hard, expensive, and time-consuming to collect.

  The main advantage of semi-supervised deep learning techniques is their ability to build better classifiers that compensate for the lack of labeled training data. However, for these methods to be effective, certain assumptions must hold, such as assuming that the decision boundary should avoid regions with high density. This facilitates the extraction of additional information from unlabeled instances to regularize training.

Despite their advantages, SSL techniques require careful implementation. Misalignment between the problem structure and the model's assumptions can lead to a degradation in classification performance.[56]

2.4.4 Deep Learning Techniques in Arabic Handwriting Recognition:

- **Convolutional Neural Networks (CNNs):** Used for hierarchical feature extraction from images of handwritten Arabic text. CNNs are effective in distinguishing visually similar characters, especially in the presence of diacritics, and in identifying spatial structures within connected scripts.

- **Recurrent Neural Networks (RNNs):** Applied to model the sequential nature of handwriting by processing the writing order and temporal dependencies between characters. However, vanilla RNNs are limited in handling long sequences due to vanishing gradients.

- **Long Short-Term Memory Networks (LSTMs):** A specialized form of RNNs designed to capture long-term dependencies. LSTMs are particularly suitable for Arabic handwriting due to their ability to manage variations in character connectivity and writing styles over long text lines.

- **Recursive Neural Networks (RecNNs):** Used to model hierarchical or structured information, though they are less commonly applied in handwriting recognition. In theory, they may be used to represent structural relationships in Arabic words, such as root-pattern morphology, but practical applications in handwriting recognition remain limited.

- **Hybrid Models:** Combine multiple architectures (e.g., CNN + LSTM) to leverage both spatial and temporal features. Such models have shown improved performance in recognizing cursive and connected Arabic handwriting by integrating visual and sequential analysis.

2.5 Natural Language Processing (NLP):

2.5.1 Definition NLP:

Natural Language Processing (NLP) is a research and application field concerned with enabling computers to understand, interpret, and generate human language in a way that is both

meaningful and contextually appropriate[57]. NLP relies on a set of computational techniques supported by linguistic theories, used to analyze and represent naturally occurring texts at one or more levels of linguistic analysis. The overarching goal is to model language processing in a manner that approaches human-like understanding, facilitating a wide range of applications across both text and speech domains.[58]

In recent years, this field has expanded significantly with the integration of machine learning approaches, particularly statistical methods, which have greatly enhanced the accuracy and efficiency of language interpretation. As a result, NLP has become an interdisciplinary domain, drawing from areas such as psychology, cognitive science, and linguistics, and has led to the development of various software tools for modeling and understanding human language more effectively.[59]

2.5.2 The Importance of Natural Language Processing:

Natural Language Processing (NLP) is considered one of the fundamental pillars of artificial intelligence, as it enables machines to effectively understand and analyze human language, whether written or spoken. Its importance lies in its ability to bridge the gap between human communication and computational systems, making it a vital component in the development of intelligent applications such as machine translation, virtual assistants (e.g., Siri and Google Assistant), chatbots, sentiment analysis, information extraction, and text summarization. The significance of NLP becomes even more prominent when dealing with Arabic texts, due to the linguistic complexity of the language in terms of diacritics, morphology, and the diversity of dialects—making it a rich field for research and innovation.

2.5.3 Basic Techniques in Natural Language Processing:

NLP involves a sequence of techniques to process textual data, beginning with tokenization and text preprocessing. Tokenization splits the text into individual units known as tokens—such as words or subwords—which serve as the foundation for further analysis. Preprocessing then cleans and standardizes the text, preparing it for interpretation by machine learning algorithms. For instance, the sentence "I love NLP" would be divided into tokens like ["I", "love", "NLP"][60]. Part-of-Speech (POS) Tagging assigns grammatical categories (e.g., noun, verb) to tokens, enabling syntactic understanding [61]. Named Entity Recognition (NER) identifies proper nouns like “Algiers” or “Mahrez” [62]. Parsing constructs syntactic trees to model sentence structure .

These techniques often leverage statistical models or machine learning. For instance, Hidden Markov Models were historically used for POS tagging, while neural networks dominate modern approaches [6]. Figure 1 illustrates a basic NLP pipeline.

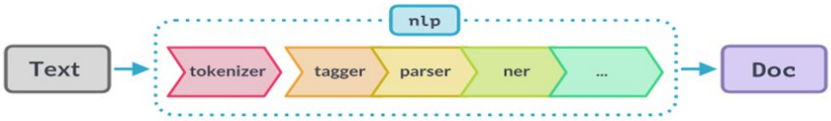


Figure 2.7: Classification of handwritten text recognition systems.[33]

2.5.4 Challenges of Natural Language Processing in the Arabic Language:

Morphological Richness and Linguistic Complexity: Arabic poses unique challenges for natural language processing (NLP) due to its linguistic complexity, particularly its rich and intricate morphological structure. A single Arabic word can encapsulate multiple layers of grammatical meaning through the use of prefixes, suffixes, infixes, and various grammatical markers such as gender, number, person, tense, case, and mood. Additionally, clitics—such as pronouns, conjunctions, and prepositions—can attach directly to words, making them semantically and syntactically dense.

Such morphological richness results in a significantly higher number of unique word forms compared to languages like English. As a consequence, tasks like tokenization, part-of-speech tagging, and morphological analysis become considerably more challenging for machine learning models in Arabic NLP.[61]

Diacritics: Short vowels, often omitted in modern texts, lead to ambiguity (e.g., كتب can mean “books” or “he wrote”) [8]. Dialectal Variation: Over 20 Arabic dialects differ significantly from Modern Standard Arabic (MSA), hindering model generalization [61].

Limited resources exacerbate these issues. Arabic datasets for training NLP models are smaller than English ones, reducing performance . Recent efforts, like AraBERT, adapt transformer models to Arabic, but dialectal coverage remains sparse .[63]

2.5.5 The Role of NLP in Handwriting Recognition Systems:

Handwriting recognition systems convert handwritten text into digital formats, relying heavily on NLP for post-processing . Optical Character Recognition (OCR) extracts characters, but

NLP corrects errors and interprets context . For example, in Arabic handwriting, distinguishing between similar characters (e.g., ب and ت) requires contextual analysis, which NLP provides via language models [64].

NLP techniques like N-gram models predict likely word sequences, improving recognition accuracy . Deep learning models, such as Recurrent Neural Networks (RNNs), integrate OCR and NLP to process entire lines of text . In applications like digitizing historical Arabic manuscripts, NLP ensures semantic coherence .

2.5.6 Techniques in Natural Language Processing:

Techniques in Natural Language Processing (NLP) Several fundamental techniques are widely used in NLP, including:

- **Tokenization:** Dividing text into individual units such as words, phrases, or other meaningful components.
- **Part-of-Speech Tagging (POS):**Determining the grammatical roles of words, such as nouns, verbs, adjectives, etc.
- **Named Entity Recognition (NER):** Identifying and classifying entities within text, including names of people, locations, organizations, and dates.
- **Word Embeddings:** Encoding words into continuous vector representations that capture semantic meaning (e.g., Word2Vec, GloVe).
- **Transformer Models:** Advanced models like BERT and GPT that utilize attention mechanisms to process large-scale text data and generate context-aware representations.[65]

2.6 Related Work on Arabic Handwriting Recognition

- Afafe Lahreche (2019) [66], proposed a model for recognizing Arabic handwritten characters using a Convolutional Neural Network (CNN) and TensorFlow. The dataset used contained 16,800 handwritten characters, divided into a training set (13,440 images) and a test set (3,360 images).The model was implemented using Google Colab and included convolutional layers, max-pooling, normalization, fully connected layers, and dropout layers. The best accuracy of 93% was achieved using the ReLU activation function with the SGD optimizer, highlighting the effectiveness of this configuration for Arabic character recognition.1

- Mahdi et al. (2023) [67] proposed a hybrid model for recognizing handwritten Arabic characters that combines deep learning techniques with the concept of Neutrosophic Sets, a powerful approach for handling uncertain or inconsistent data. The model first transforms images into the neutrosophic domain, then passes them through a Convolutional Neural Network (CNN) to extract spatial features. This is followed by a Bidirectional Recurrent Neural Network (Bi-LSTM or Bi-GRU) to capture the temporal and contextual features of the handwritten script.The model was tested on two datasets: • Hijja, a dataset of handwritten characters collected from children, • and AHCD (Arabic Handwritten Character Dataset). The model achieved an accuracy of: • 92.38% using Bi-LSTM on the Hijja dataset, • and 97.38% using Bi-GRU on the AHCD dataset, surpassing previous models in the field without the need for explicit segmentation of characters.

- In their study, the authors [68]of emphasized the pivotal role of the feature extraction stage in Arabic OCR systems, pointing out that it remains a key challenge in the development of robust recognition models. They proposed a hybrid approach that integrates multiple models, each equipped with distinct feature extraction techniques, to enhance the overall recognition performance. This ensemble strategy led to an impressive recognition accuracy of 99.88% on the IFN/ENIT dataset [14]. Nevertheless, a notable limitation of this work is its restricted evaluation scope—limited to only 100 word classes—making it less generalizable. For more comprehensive solutions, a shift toward character-level processing, such as segmentation-based methods, is considered more scalable and effective than relying solely on filtered feature extraction.

- Elsayed et al.[69] (2024) proposed a recent model for Arabic handwritten text recognition that combines EfficientNetB3 with BiLSTM and a Multi-Head Attention mechanism. The model adopts a hybrid CNN-RNN architecture, where features are extracted using EfficientNetB3 and sequentially processed through three BiLSTM layers. The attention mechanism enhances focus on different parts of the input, while a CTC layer aligns input and output sequences without the need for character segmentation. The model achieved remarkable results, with a Character Error Rate (CER) of 17.26% on the KHATT dataset and 0.28% on the AHAWP dataset, outperforming previous models such as MDLSTM and Transformer-based architectures. However, the model remains limited by the scope of training data and lacks the ability to handle long texts or understand linguistic context, suggesting room for future improvements.

- Mustafa, M. E. and Elbashir, M. K.[70] proposed a model for handwritten Arabic name recognition using Convolutional Neural Networks (CNNs) and a holistic approach that avoids segmenting words into individual characters. They used the SUST-ARG dataset, which contains 40,000 images of Arabic names. However, experiments were limited to 20 common male names, representing over 50% of the student names. The model architecture includes four convolutional layers, two max pooling layers, and two dense layers, utilizing ReLU and Softmax activation functions. The model achieved an accuracy of 99.14% using 10-fold cross-validation. Despite the high accuracy, the model's generalizability is limited by the small number of classes and its sole reliance on the holistic approach, which requires integration with an analytic method to handle uncommon names.

- El-Melegy et al [71]. proposed a deep convolutional neural network (CNN) model for the recognition of Arabic handwritten literal amounts, as an alternative to traditional methods based on handcrafted features. The model was applied to the AHDB dataset, which consists of 50 classes written by 100 writers, with a total of 4,971 samples. The data was split into 80% for training and 20% for testing. The proposed network comprises 17 layers, including convolutional, batch normalization, max pooling, and fully connected layers, ending with a Softmax classification layer. Initial experiments yielded an accuracy of 82.7%, but the researchers identified overfitting issues. To address this, they tuned hyper parameters such as the L2 regularization and the number of epochs, and applied data augmentation techniques using random rotations and translations. These adjustments significantly improved the model's performance, achieving an accuracy of 97.8%. When compared with classical approaches, the CNN model demonstrated clear superiority, achieving 100% accuracy on 25 classes, with only 34 misclassified samples out of 994 in the test set.

- Ghanim et al[72]. conducted a comparative study to evaluate the performance of various deep convolutional neural network (DCNN) models in Arabic handwritten text recognition using the IFN/ENIT dataset. The methodology included three stages: matching using the HAC clustering algorithm to group visually similar words, ranking to reduce the number of candidate classes, and final classification using six DCNN models: AlexNet, VGG16, GoogleNet, ResNet50, ResNeXt, and DenseNet. The results showed that AlexNet outperformed the other models, achieving an accuracy of 95.6%, despite being shallower than the others. This challenges the assumption that deeper models always perform better, especially when working with relatively small datasets. However,

the study has some limitations, such as its reliance on the accuracy of the clustering and ranking stages, and the need for further experiments on diverse datasets to ensure generalizability.

- Rabi et al [73] . proposed a hybrid model combining Convolutional Neural Networks (CNN) and Hidden Markov Models (HMM) for offline Arabic handwritten word recognition using the IFN/ENIT dataset. The model employs CNN as an automatic feature extractor from word images, replacing handcrafted features, and uses HMM for the classification process. Experiments were conducted on two standard evaluation scenarios ("abcd" and "abcd-e"), where the proposed model achieved recognition accuracies of 88.95% and 89.23%, respectively, outperforming the baseline HMM system that relies on manually designed features. The results demonstrate the effectiveness of the approach and the ability of CNN to extract salient and discriminative features. However, the generalizability of the model remains limited, and the achieved accuracy may still fall short for real-world applications, suggesting that future work should explore integrating statistical language models to further enhance recognition performance.

- Khosravi and Chalechale (2022) [74]proposed a novel hybrid model called AECNN for recognizing Persian and Arabic handwritten words by combining Convolutional Neural Networks (CNN) with Autoencoder networks. The model is based on segmenting words into subword units, extracting features using an Autoencoder, and classifying them using a CNN, followed by a dedicated merging algorithm that reconstructs the final word based on a subword dictionary. The model was evaluated on the Iranshahr dataset, which contains 17,000 images of handwritten names of 503 Iranian cities. It achieved an accuracy of 91.09%, outperforming the standalone CNN model by 3%. The subword merging mechanism played a significant role in correcting a considerable number of misclassifications.

- Zermi et al. (2007) [75] proposed a hybrid model for recognizing Arabic handwritten words by combining Hidden Markov Models (HMM) with Artificial Neural Networks (ANN). The model aims to leverage the temporal modeling capabilities of HMM alongside the classification strength of ANN. Experimental results showed that the hybrid model outperformed the individual use of HMM or ANN, particularly in handling the cursive nature of Arabic script. However, the study lacks evaluation on standard benchmark datasets and does not incorporate modern deep learning techniques, which have since proven to be more effective.

- Alabodi and Li [76] proposed an efficient system for recognizing offline Arabic handwritten words without relying on deep learning techniques. The approach is based on analyzing the geometric structure of letters and converting them into numerical vectors. The system begins with a skeletonization stage, followed by skew correction and the transformation of diagonal pixels into horizontal and vertical lines. These are then encoded into vectors representing each character and matched against a manually constructed reference dictionary.The system was evaluated on the IFN/ENIT dataset using 1300 images for training and 860 for testing, achieving an accuracy of 93.7% on the training set and 93.1% on the test set. The method proved effective in handling the complex segmentation challenges of Arabic script and demonstrated the potential for generalization to other languages. However, it remains limited by its reliance on a fixed encoding table and the absence of modern techniques that could further enhance its performance.

- Abdelkarim Mars and Georges Antoniadis [77]proposed an Arabic online handwriting recognition system using neural networks, combining a Time Delay Neural Network (TDNN) and a Multi-Layer Perceptron (MLP). The system was trained on a custom-built dataset that includes 6090 handwritten characters and 1080 handwritten words, collected using a digital pen (Nokia SU-1B) on Anoto paper with a resolution of 677 dpi and a sampling rate of 100Hz. The approach involved several stages including preprocessing (noise removal, slant correction), feature extraction (7 features per point such as normalized coordinates, direction, curvature, and pen state), segmentation, and classification. The TDNN was used to compute the probability of sequences of characters forming a word.The model achieved 98.5% accuracy on character recognition and 96.9% on word recognition, outperforming the commercial MyScript system, which achieved 90.3% and 88.9% respectively. However, some errors were noted due to diacritics (e.g., shadda, long alif), overlapping strokes, and character omission during writing.

2.7 Conclusion

This chapter concludes an exploratory journey through the scientific and technical dimensions of Arabic handwriting recognition, highlighting the intricacies of Arabic script and the role of advanced technologies in addressing these complexities. The foundations of digital image processing, deep learning architectures, and natural language processing mechanisms were examined, contributing to a deeper understanding of the challenges and potential solutions. Prior

studies showcased notable achievements but underscored the ongoing need for an innovative approach that effectively balances both visual and linguistic contexts. This integrated understanding sets the stage for Chapter 3, which focuses on detailing the hybrid framework composed of Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and n-gram models for enhancing the accuracy of recognizing handwritten Arabic words.

Chapter 3

Methods Used in the Model

3.1 Introduction

Building on the foundational concepts established in previous chapters, this chapter presents a hybrid framework designed to enhance the accuracy of handwritten Arabic word recognition. The framework integrates three core components: Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and n-gram language models. LSTM networks model sequential dependencies in handwritten word sequences, leveraging their capacity to capture long-term contextual relationships. CNNs extract robust spatial features from word images, enabling effective representation of visual patterns. Complementing these, n-gram models incorporate statistical linguistic patterns to improve word prediction accuracy.

3.2 Convolutional Neural Networks:

Convolutional Neural Networks (CNNs) are a specialized subset of deep learning algorithms within the machine learning taxonomy, as shown in Figure 1 [78]. First introduced by LeCun in the 1980s, they have become a cornerstone of deep learning for processing visual data, with widespread applications in data analytics, natural language processing, and image and signal processing. These networks have significantly advanced deep learning by effectively mimicking the human brain's functionality in machine learning models [79]. Inspired by the animal brain's visual cortex, where neurons connect to specific regions of the visual field[80], they are designed to progressively learn visual patterns—such as edges, shapes, or objects—through multiple layers that process data efficiently .

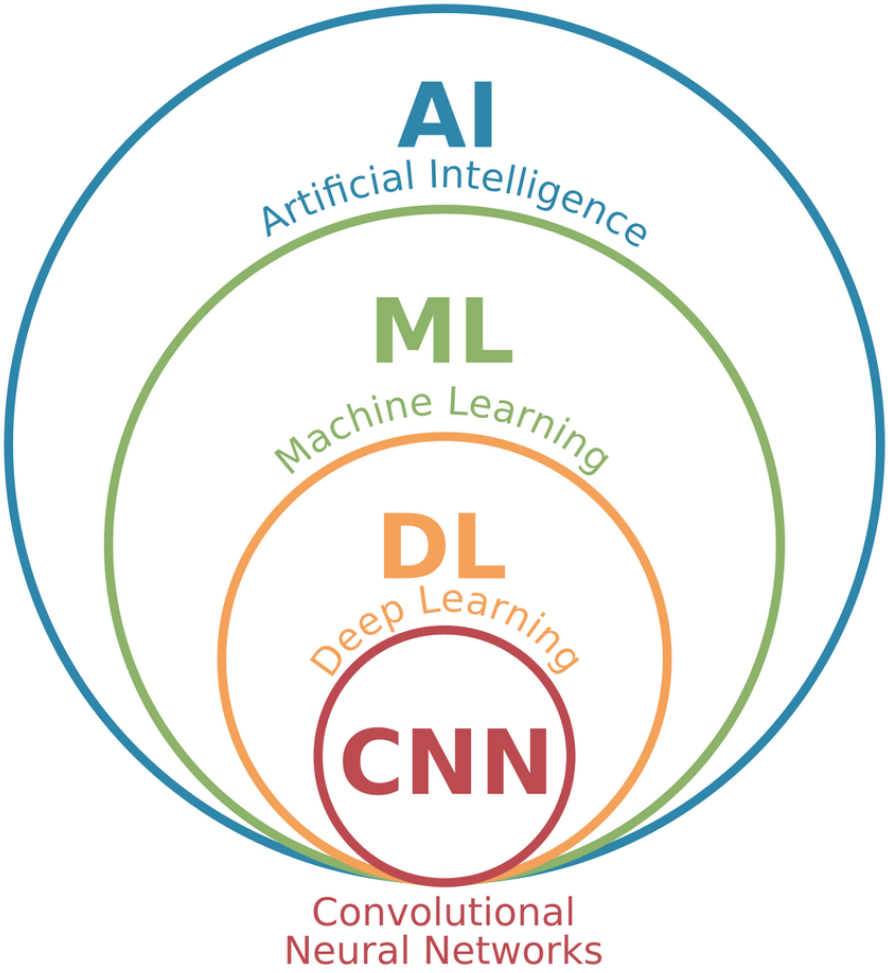


Figure 3.1: Relation between the terms artificial intelligence (AI), machine learning (ML), deep learning (DL), and convolutional neural network (CNN).[81]

3.2.1 Importance of CNN:

- Automatic Feature Extraction: CNNs learn relevant features directly from raw data, reducing the need for manual feature engineering. [82]
- Efficiency and Scalability: Through local connectivity and weight sharing, CNNs reduce the number of parameters, improving training and inference speed.[83]
- Versatility: CNNs handle various data types-images, audio, sensor signals, and even multidimensional scientific data-making them widely applicable.[84]
- Edge AI and Embedded Vision: CNNs enable powerful machine learning on resource-constrained devices, expanding AI applications to edge computing.[85]
- Improved Accuracy: CNN-based models often outperform traditional methods in tasks like image classification, fault detection, and forecasting.[86]

3.2.2 CNN Architectures:

- **Input Layer:** This layer receives an image characterized by its height, width, and depth, incorporating either a single channel (grayscale) or three color channels (Red, Green, and Blue) along with their raw pixel values.[80]

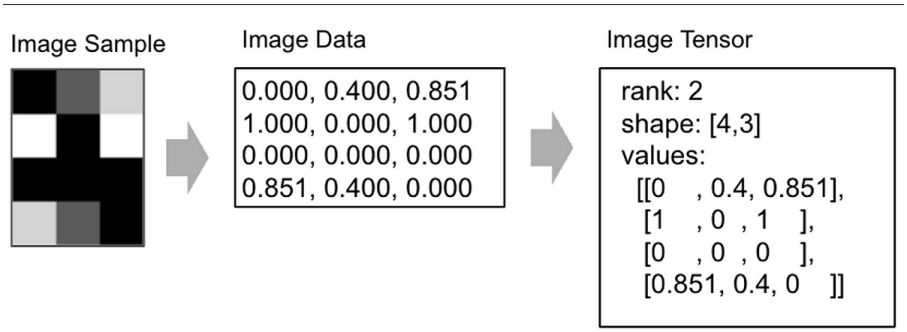
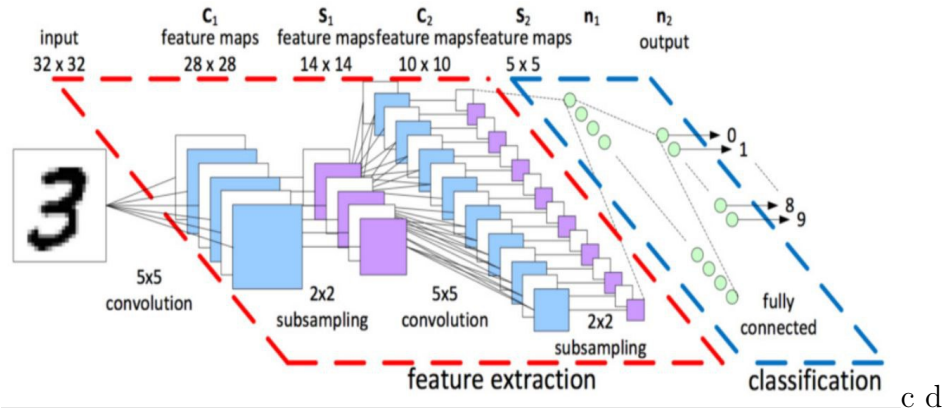


Figure 3.2: Representation of grayscale image data as a matrix.[87]

- **Convolutional layer:** The convolutional layer is one of the fundamental and essential components in the architecture of Convolutional Neural Networks (CNNs). It receives an image as input and applies filters, commonly of size 3×3 or 5×5, as illustrated in Figure 3, to extract important spatial features from the data.[88]



c d

Figure 3.3: Convolutional layer.[88]

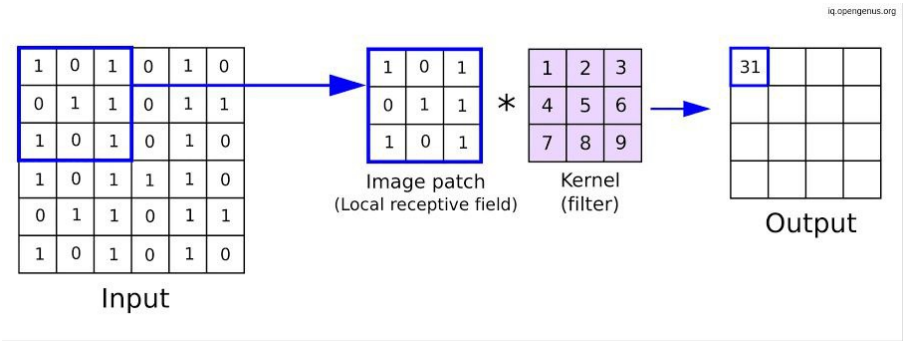


Figure 3.4: Illustration of the convolution operation using a 3×3 kernel over a local image patch.[89]

Figure 3.4 illustrates the basic operation of a **convolutional layer** in a Convolutional Neural Network (CNN). It shows how a kernel (filter) is applied to a small region of the input image, known as the *image patch* or *local receptive field*, to extract features and generate an activation map.

- **Input (original image):** A 6×6 matrix representing a grayscale image. A 3×3 patch is selected from the top-left corner.
- **Image Patch:**

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

- **Kernel (Filter):**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- **Convolution Operation:** The element-wise multiplication between the image patch and the kernel is computed, followed by summation:

$$(1 \times 1) + (0 \times 2) + (1 \times 3) + (0 \times 4) + (1 \times 5) + (1 \times 6) + (1 \times 7) + (0 \times 8) + (1 \times 9)$$

$$= 1 + 0 + 3 + 0 + 5 + 6 + 7 + 0 + 9 = \mathbf{31}$$

- **Output (Activation Map):** The result 31 is placed at the corresponding position in the output feature map. As the kernel continues sliding across the input image, similar computations are performed to generate the full activation map.

- **Pooling Layer:**

  Pooling Layer The pooling layer is a fundamental step that typically follows convolutional layers. Its main purpose is to reduce the spatial dimensions of the feature maps, thereby decreasing the number of parameters, shortening training time, and mitigating overfitting. Pooling operations are applied independently to each feature map, reducing the height and width while preserving the depth.

  Max pooling is the most commonly used pooling technique. It selects the maximum value within a defined window that slides over the feature map. Unlike convolutional layers, pooling layers do not involve learnable parameters; instead, they rely on simple mathematical operations, where the window size and stride are predefined.2

  On the other hand, average pooling computes the average value within each window as it moves across the feature map. [90]

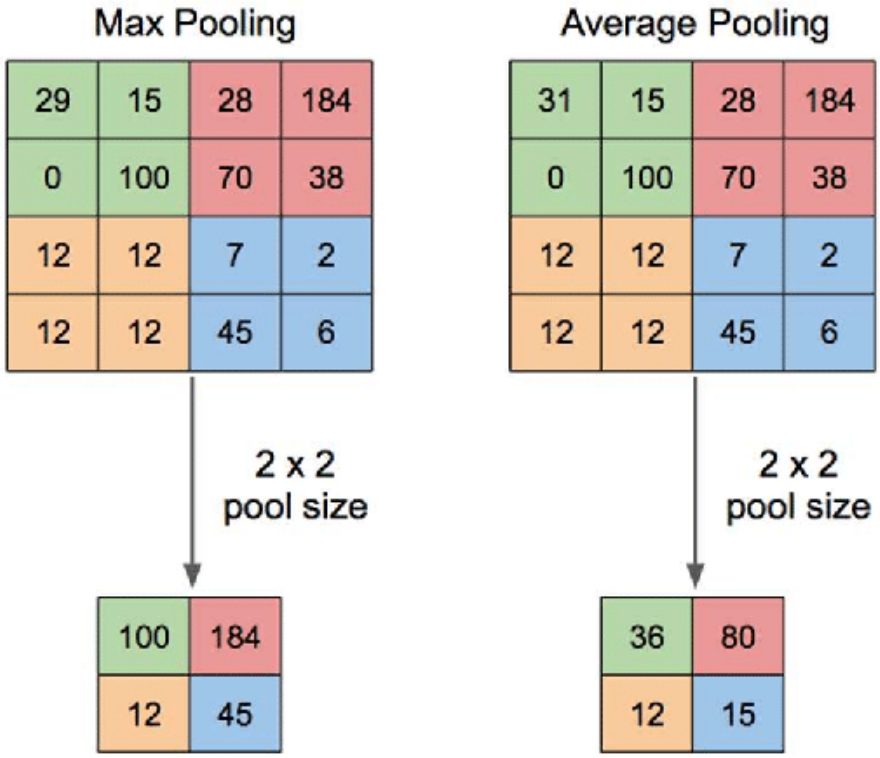


Figure 3.5: Classification of pooling.[90]

- **Activation Layers(RELU):**

The ReLU (Rectified Linear Unit) activation function is a widely used activation function in Convolutional Neural Networks (CNNs) to introduce nonlinearity to the model. It is mathematically defined as: f(x) = max(0,x) .

where the function takes an input value x and outputs x if the value is greater than or equal to zero, and outputs zero if the value is negative. This property makes ReLU computationally simple and efficient, while also helping to accelerate training and mitigate the vanishing gradient problem that may occur with other activation functions like sigmoid or tanh. In CNNs, ReLU is typically applied after convolutional layers to transform the filter outputs into nonlinear representations, enabling the network to learn complex patterns in the data.[91]

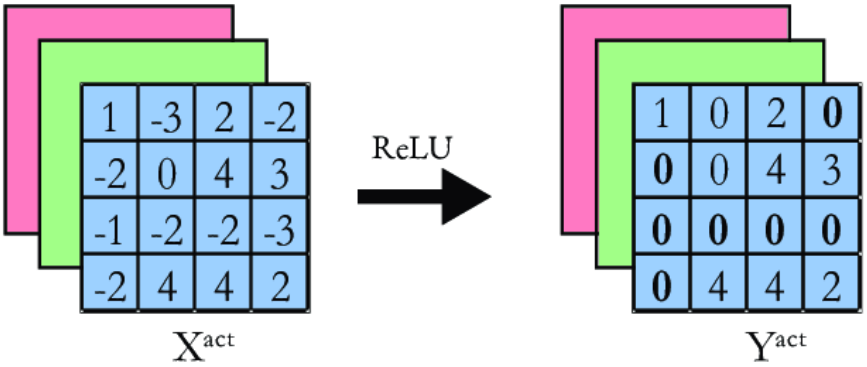


Figure 3.6: Example of ReLU activation function.[92]

- **Fully-Connected Layer:** Fully-Connected Layers are a fundamental component of Convolutional Neural Networks (CNNs), primarily used in the final stage for classifying images into specific categories based on training. These layers consist of weights, biases, and neurons, and they work by connecting each neuron in the previous layer to every neuron in the next layer, a process known as full connectivity.[93] The fully-connected layer receives its inputs from the final outputs of the preceding layer, which may be an activation layer (e.g., ReLU), a pooling layer, or a convolutional layer. The feature maps produced by the final convolutional or pooling layer are transformed into a one-dimensional (1D) vector, known as a flattened vector, which is then fed into the fully-connected layer (Yamashita et al., 2018). This layer produces a vector of dimensions (N), where (N) represents the number of categories into which the program classifies the images.[93]

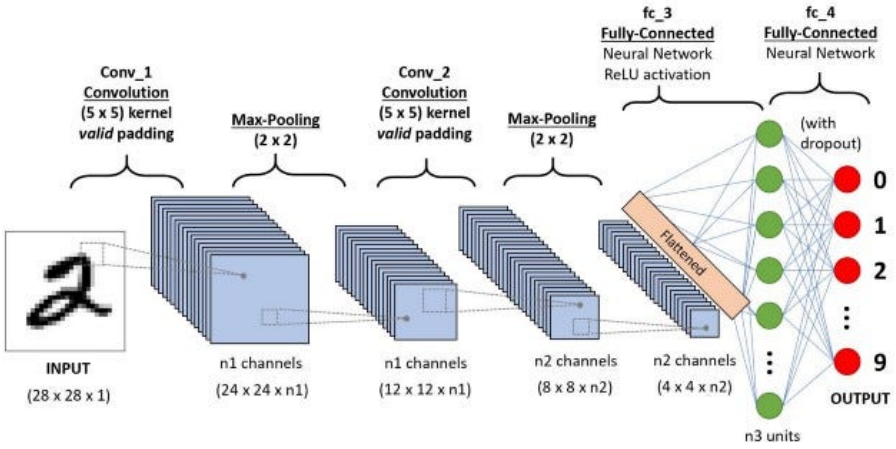


Figure 3.7: Role of Fully-Connected Layer in CNN Architecture.[94]

3.3 Long Short-Term Memory (LSTM):

3.3.1 Definition:

Long Short-Term Memory (LSTM) networks, first introduced by Hochreiter and Schmidhuber in 1997 [95], were designed to overcome the limitations of traditional recurrent neural networks (RNNs), particularly the vanishing and exploding gradient problems that impede learning long-term dependencies in sequential data. Unlike standard RNNs, LSTM networks incorporate memory cells that store information over extended periods, regulated by three core gates: the input gate, output gate, and forget gate. These gates control the flow of information, allowing the network to selectively retain or discard data based on context [95] [96]. This architecture enables LSTM networks to effectively capture long-range dependencies without losing past information or causing premature convergence, resulting in more stable and accurate training outcomes.

LSTM networks are highly versatile, capable of processing not only single data points (such as images) but also entire sequences of data (such as speech or video). Their ability to capture the temporal structure of sequences makes them well-suited for a wide range of applications, including machine translation, text generation, speech recognition, unsegmented connected handwriting recognition, and anomaly detection in network traffic or Intrusion Detection Systems (IDS) [96] [97]. These capabilities have established LSTM networks as a powerful tool for handling complex sequential data across diverse domains.

3.3.2 The architecture of LSTM:

The architecture of the Long Short-Term Memory (LSTM) block, as illustrated in Figure III.8, enhances the memory capabilities of traditional Recurrent Neural Networks (RNNs) by incorporating mechanisms for selective retention and elimination of information. This is achieved through specialized structures, namely the cell state and three regulatory gates. In contrast to the conventional RNN, which relies solely on a hidden state, the LSTM block integrates four additional components: the cell state (Ct), the input gate (it), the output gate (Ot), and the forget gate (ft). These components interact in a sophisticated manner to process and generate meaningful information from sequential training data, enabling the LSTM to effectively capture long-term dependencies.[98]

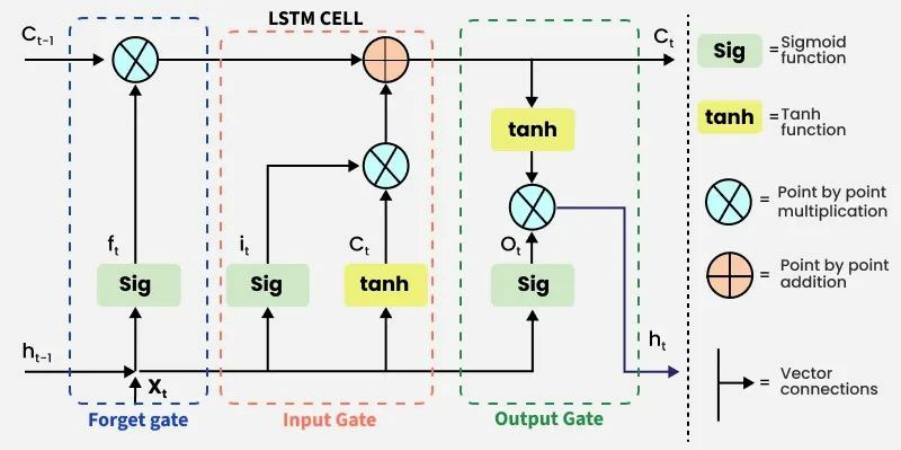


Figure 3.8: LSTM Model.[99]

- **Cell state:**

  The cell state in an LSTM (Long Short-Term Memory) is a key component that acts as the memory of the network, allowing it to retain information over long time steps. It stores long-term dependencies and is updated at each time step through carefully controlled gates (input, forget, and output gates) that regulate the flow of information. This cell state enables the LSTM to remember or forget information as needed, which helps in modeling sequences with long-range dependencies. [100]

  More specifically, the cell state carries the accumulated information, and its stability or change determines how much precision or attention the network gives at each time step. When the cell state changes significantly, the network requires higher precision; when it remains stable, lower precision suffices.[101]

- **Forget Gate:**

The forget gate determines which components of the previous cell state, c_{t-1}, to retain or discard. It applies a sigmoid activation function, σ, to produce an output vector, $f_t \in [0, 1]$, where values near 1 indicate retention and values near 0 indicate removal. The forget gate's output is computed as [102]:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{3.1}$$

where W_f is the weight matrix, b_f is the bias, h_{t-1} is the previous hidden state, and x_t is the current input. The filtered cell state, c_f, is obtained via element-wise multiplication:

$$c_f = c_{t-1} \odot f_t. \tag{3.2}$$

- **Input Gate**

The input gate controls updates to the cell state by determining the extent and nature of new information to incorporate. It consists of two functional units. The first uses a hyperbolic tangent (tanh) activation to generate a candidate cell state, $\tilde{c}_t \in [-1, 1]$, calculated as [102]:

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c). \tag{3.3}$$

The second unit employs a sigmoid activation to produce a modulation vector, $m_t \in [0, 1]$, which governs the magnitude of the update:

$$m_t = \sigma(W_m \cdot [h_{t-1}, x_t] + b_m). \tag{3.4}$$

The updated cell state, c_t, is computed by combining the filtered state and the modulated candidate state:

$$c_t = c_f + \tilde{c}_t \odot m_t. \tag{3.5}$$

- **Output Gate**

The output gate generates the hidden state, h_t, which serves as the output for the current time step and input for the next. It uses a sigmoid activation to select relevant information from the concatenated previous hidden state and current input [102]:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o). \tag{3.6}$$

The final hidden state is obtained by combining this output with the updated cell state, applying a tanh activation:

$$h_t = o_t \odot \tanh(c_t). \tag{3.7}$$

This hidden state, h_t, is passed to the next time step or layer, or serves as the prediction in a single-layer LSTM.

3.3.3 The Role of Combining CNN and LSTM in Improving Handwritten Text Recognition Accuracy

The reason for combining CNN and LSTM networks in handwritten text recognition is to leverage the strengths of each to achieve better performance, as:

CNN (Convolutional Neural Network) is capable of extracting spatial features from images, such as patterns and the general shape of letters and symbols, which is crucial for understanding the details of handwritten text.

LSTM (Long Short-Term Memory) specializes in processing sequential and temporal data, which helps capture the temporal order or sequence in handwriting, such as the strokes or the order of letters and words.

Therefore, combining CNN and LSTM allows the model to first extract spatial features from the image via CNN, then process the temporal sequence or context through LSTM, improving the accuracy of handwritten text recognition.

This combination is also used in other fields like sign language recognition or mixed text recognition, as it combines the power of CNN in feature extraction with LSTM's ability to understand temporal context.

3.4 N-Gram Model

Word n-grams are defined as contiguous sequences of n words that appear adjacent to each other in a given text. This technique is widely used in Natural Language Processing (NLP) to model linguistic context and analyze text statistically.

Word n-grams have shown high effectiveness in various NLP applications, such as next-word prediction, speech recognition, handwriting recognition, augmentative communication for individuals with disabilities, and spelling error detection.[103]

N-gram matching has achieved notable success in handling noisy textual input across various domains, such as:

- Interpreting postal addresses
- Text information retrieval
- Various applications in natural language processing

The main advantage of this type of matching lies in its very nature: Since each string is broken down into small segments, any errors present affect only a limited number of those segments, while the rest remain intact.[104]

3.4.1 Typed n-grams

We briefly recall the notion of typed character n-grams (in short, typed n-grams). The category and supercategory of an n-gram depend on its content and position within a word or sentence. We can distinguish between affix, word, and punct supercategories, reflecting morpho-syntax, document topic, and the author's style, respectively. Within each supercategory, we can further distinguish fine-grained categories.

Within the affix supercategory, prefix and suffix categories denote n-grams that are proper prefixes and proper suffixes of words, while the space-prefix and space-suffix categories denote n-grams that begin or end with a space, respectively.

Categories in the word supercategory (whole-word, mid-word, multi-word) are assigned to n-grams covering an entire word, the non-affix part of a word, or spanning multiple words, respectively.

The specific categories of the punct supercategory (beg-punct, mid-punct, end-punct) are assigned to n-grams containing one or more punctuation characters.

3.4.2 Conclusion

To conclude, this chapter presented the architectures of the Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) network, highlighting their roles in Arabic handwritten word recognition. It also introduced the N-gram correction mechanism as a linguistic post-processing step to refine recognition results. This combination of visual and linguistic components provides a solid foundation for the experimental implementation discussed in the next chapter.

Chapter 4

Implementation and Results

4.1 Introduction

Following the theoretical and architectural foundations laid out in the previous chapters, this chapter is dedicated to the practical implementation of the proposed hybrid model. It details the setup of the working environment, data preparation, model training, and prediction stages. The chapter also presents the experimental results and provides an initial analysis of the model's outputs to evaluate its responsiveness to the intended objectives. This practical phase serves as a key step in transitioning from theory to a functional, scalable prototype.

4.2 Work Environment

4.2.1 Kaggle

Kaggle is primarily used as a platform for data science and machine learning competitions, where individuals and organizations collaborate to solve complex data problems. Kaggle facilitates the exchange of data and knowledge by hosting datasets, organizing competitions, and providing a collaborative environment for problem-solving.

Kaggle also serves as a rich repository of machine learning code and interactive notebooks, which support continuous learning and development in the field of data science.

In this work, I used two T4 GPUs (GPU T4 $\times 2$) to accelerate the training process of the model, due to the high performance of these processors in executing deep learning tasks and their enhanced efficiency in handling large volumes of data.

4.2.2 Python language:

Python version 3.11.12 was used in this project.

Python is a high-level, open-source programming language known for its clear syntax and ease of use. It supports multiple programming paradigms and runs across various platforms including Windows, macOS, and Unix. Python is widely used in fields such as web development, data science, and artificial intelligence. Its flexibility and large ecosystem of libraries make it a preferred language for rapid development and research.[105]

4.2.3 The used python libraries:

- **OS:** The os library in Python is a standard module that provides an interface for interacting with the operating system. This library offers a wide range of functions for managing files, directories, processes, and system-related commands.

- **TensorFlow:** TensorFlow is an open-source library developed by Google for machine learning. It supports training neural networks to classify handwritten characters, process images, and handle natural language efficiently.

- **OpenCV**

  OpenCV is a comprehensive open-source library widely used in the fields of computer vision, machine learning, and image processing. It plays a critical role in enabling real-time operations, which are essential in modern computing systems. Through OpenCV, it is possible to process images and videos for tasks such as object detection, facial recognition, and handwriting analysis.

- **Keras:** Keras is a free and open-source library in Python that is known for its power and ease of use in developing and evaluating deep learning models.[105]

- **Matplotlib:** Matplotlib is a Python library that enables the creation of static, animated, and interactive visualizations.[105]

- **Python-Levenshtein Module:** The Python-Levenshtein package provide fast and easy python code access to the Levenshtein algorithm to perform operations on string. It is highly optimized and is comparability faster than other Python implementation because it is implemented using the C extension.[106]

- **NumPy:** NumPy is a fundamental Python library for scientific computing, providing support for multidimensional arrays along with fast and efficient functions for mathematical and logical operations, shape transformations, linear algebra, statistics, and random simulations.[105]

4.2.4 Data Sets Used:

The dataset used in this project is the **AHDB (Arabic Handwritten Database)**, developed by **Somaya Al-Ma'adeed**, **Dave Elliman**, and **Colin Higgins**. It consists of images of Arabic words handwritten by 100 different writers to ensure a diversity of writing styles. The

set includes **6,615 images** representing **63 distinct words**. Each category corresponds to a specific word, a concept known as **Class to Word**, and each image is associated with its corresponding class, known as **Class to Image**. The images are stored in **TIFF format**, ensuring high quality and clarity of handwriting details, making this database highly suitable for training and evaluating Arabic handwritten word recognition models in this project.

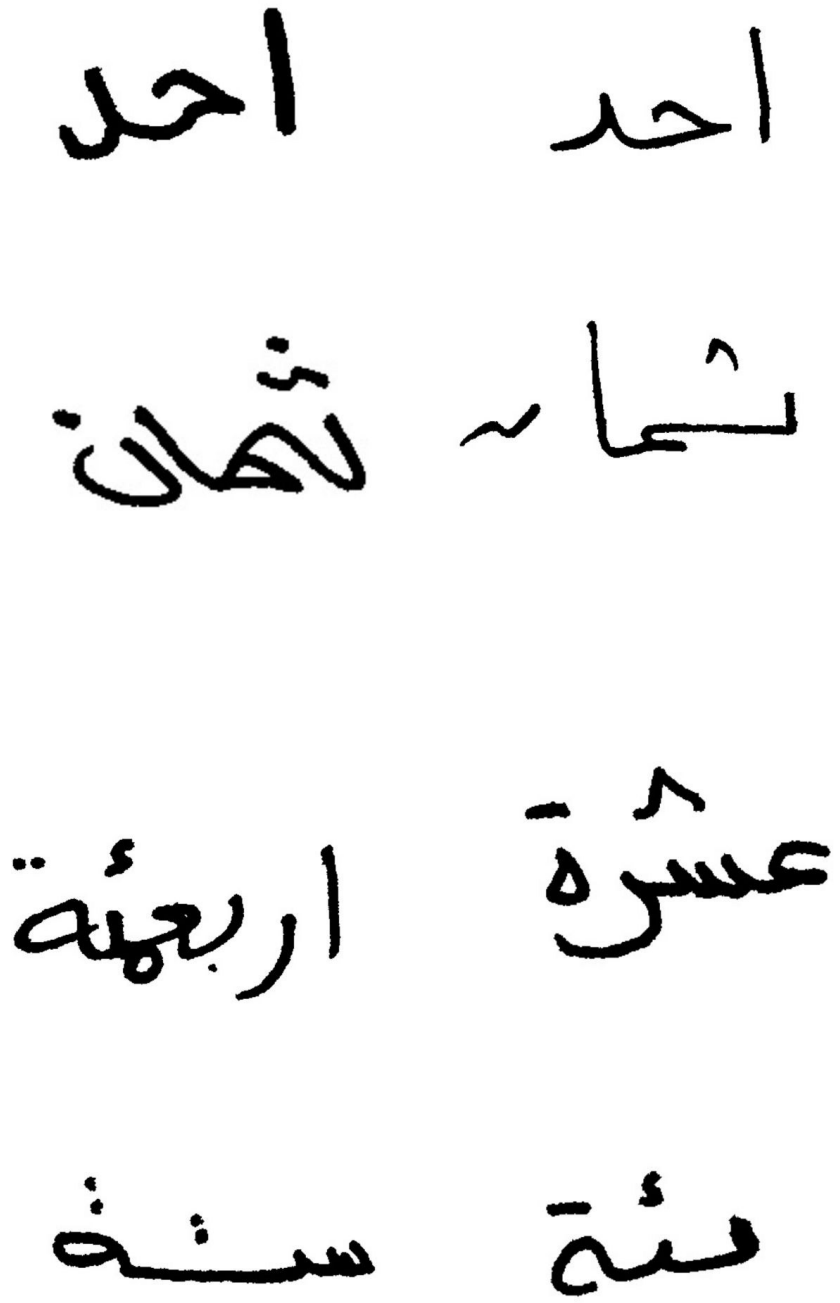


Figure 4.1: Examples of handwritten Arabic words extracted from the dataset used in this study.

4.2.5 Local Device Information

For tasks related to data preparation, debugging, and project management, a personal laptop was used with the following specifications:

- **Device:** Lenovo Laptop
- **Operating System:** Windows 7 (Edition Intégrale)
- **Processor:** Intel(R) Celeron(R) CPU N3060 @ 1.60GHz
- **RAM:** 4GB

4.2.6 Workflow Diagram of Arabic Handwritten Word Recognition Using CNN-LSTM and Linguistic Correction)

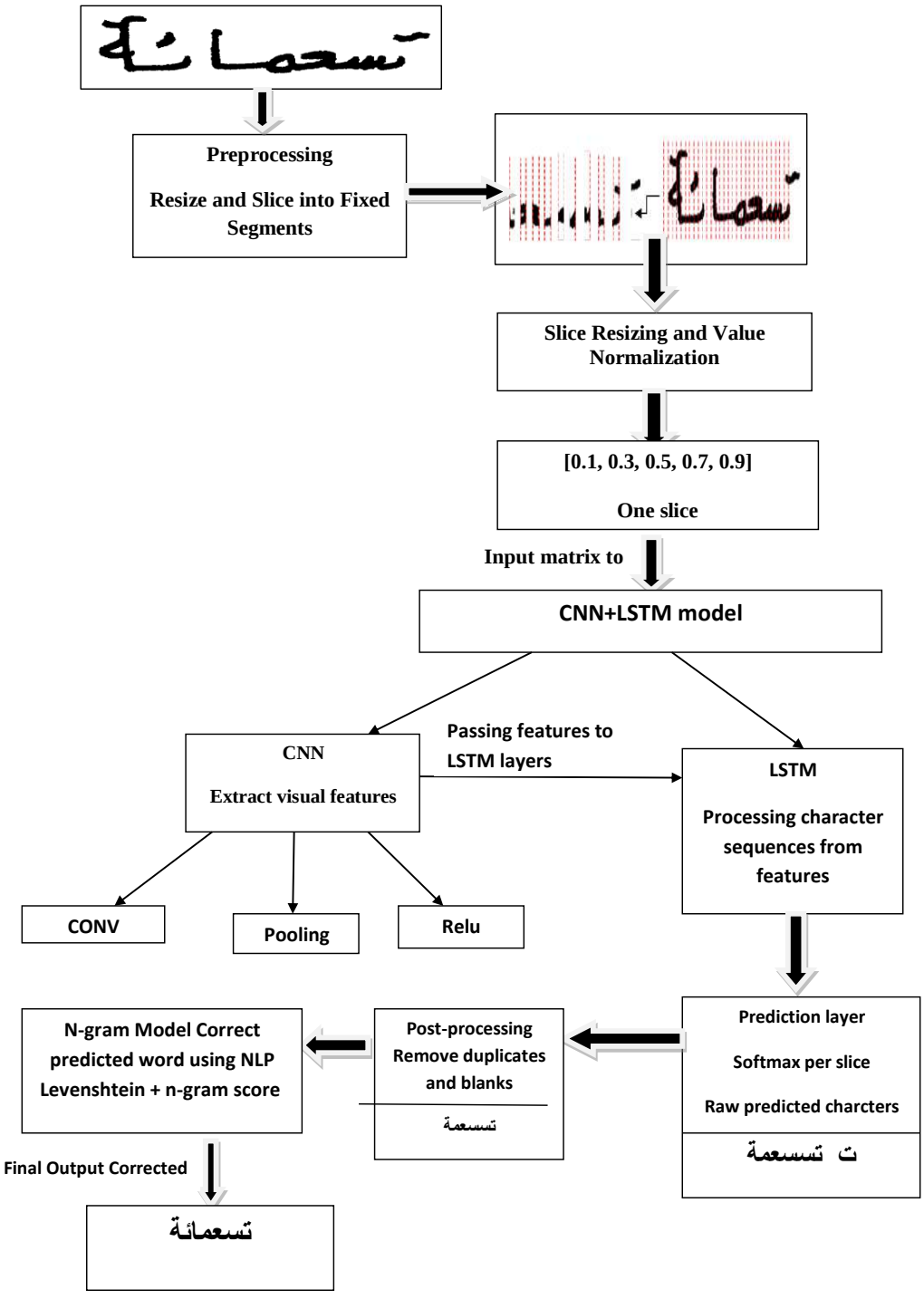


Figure 4.2: Diagram of Arabic Handwritten Word Recognition

4.3 Implementation Steps

4.3.0.1 Input Processing: Image Slicing

- **Each image is divided into 32 slices (NUM_SLICES=32).**
  Reason: The choice of 32 slices is based on the average number of characters in Arabic words in the dataset and the image width. This number provides a balance between fine-grained slicing to capture parts of characters or complete characters and reducing computational complexity. The 32 slices can accommodate most words containing up to 10–15 characters, with additional slices for spaces or elongated characters.

- **Each slice has a width of 8 pixels and a height of 64 pixels.**
  Reason: A width of 8 pixels is sufficient to capture a part of a character or a complete character in Arabic text, which is often connected and requires limited horizontal resolution. The height of 64 pixels ensures the capture of vertical features of Arabic characters (e.g., dots, loops) while maintaining a size suitable for processing by convolutional neural networks (CNNs), reducing memory usage.

- **This slicing allows the image to be analyzed as a sequence of slices, where each slice may contain a part of a character or a complete character.**
  Reason: Analyzing the image as a sequence of slices aligns with the right-to-left nature of Arabic text, enabling the model (LSTM) to understand sequential relationships between characters. This approach is suitable for text recognition in images, where each segment may contain information about a character or part thereof.

4.3.0.2 Preprocessing

- **The image is converted to grayscale to reduce dimensionality.**
  Reason: Converting the image to grayscale reduces the number of color channels from 3 (RGB) to 1, decreasing computational complexity and memory usage by approximately two-thirds. Arabic text relies primarily on contrast between characters and the background, making color information unnecessary for character recognition.

- **The image is resized to a height of 64 pixels while maintaining the aspect ratio.**
  Reason: Resizing to a height of 64 pixels standardizes all images in the dataset, facilitating processing by the neural network. Maintaining the aspect ratio prevents distortion of

Arabic characters, which depend on their geometric shapes (e.g., curves, dots) for accurate recognition.

- **Pixel values are normalized by dividing by 255 to range between 0 and 1.**
  Reason: Normalization transforms pixel values from [0, 255] to [0, 1], improving neural network training performance by accelerating convergence with optimizers like Adam and reducing the impact of large pixel value variations.

- **A matrix with dimensions (32, 64, 8, 1) is produced, where 32 is the number of slices, 64×8 is the size of each slice, and 1 is the channel (grayscale).**
  Reason: These dimensions align with the neural network architecture (CNN+LSTM), allowing each slice to be processed independently by the CNN, followed by sequence analysis by the LSTM. The single channel reflects the use of grayscale images.

4.4 CNN+LSTM

4.4.1 Convolutional Neural Network (CNN)

The convolutional neural network (CNN) is applied independently to each slice using the **TimeDistributed** layer, enabling parallel processing of the 32 slices. (To process each slice uniformly.) The CNN architecture is designed to extract robust visual features from each slice to identify patterns associated with characters. The network consists of the following layers:

- **Conv2D Layer (32 filters)**: Utilizes a 3×3 kernel with ReLU activation and `padding='same'` to extract low-level features such as edges and patterns. (32 filters: Sufficient for simple edges without complexity.)

- **MaxPooling2D Layer**: Employs a 2×2 pool size to reduce spatial dimensions while preserving important features. (2×2 pooling: To reduce size and speed up processing.)

- **Conv2D Layer (64 filters)**: Uses a 3×3 kernel with ReLU activation to extract deeper features. (64 filters: For more complex patterns.)

- **MaxPooling2D Layer**: Further reduces spatial dimensions. (2×2 pooling: To reduce size and speed up processing.)

- **Flatten Layer**: Converts the feature map into a vector. (Prepares data for dense layer.)

- **Dense Layer (128 units)**: Applies ReLU activation to reduce the dimensions to a feature vector of size 128. (Summarizes features compactly.)
- **Output**: The output of the CNN is a feature vector of shape (32, 128), where each of the 32 slices is represented by a 128-dimensional feature vector. (Matches LSTM input needs.)

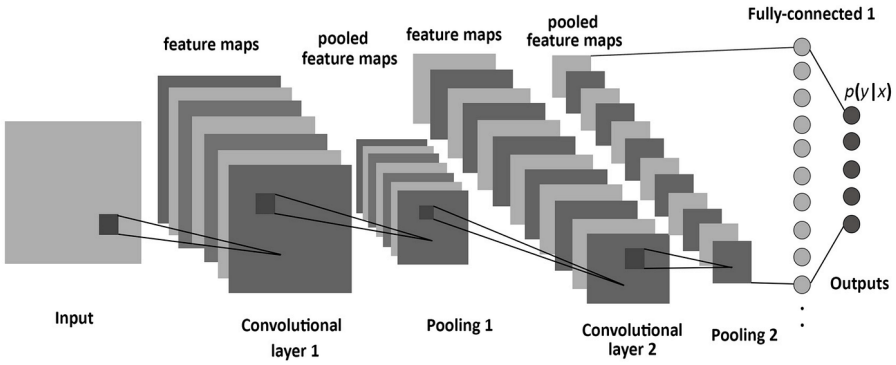


Figure 4.3: CNN Architecture

4.4.2 LSTM

After receiving the CNN output in the shape `(batch_size, 32, 128)`, the LSTM layer begins to operate as follows:

- **Input**: Consists of a sequence of 32 slices, with each slice containing 128 features. (32 slices, 128 features: Matches CNN output.)
- **Bidirectional LSTM Layer**: Contains 64 units, reads the sequence in both directions (forward and backward) using `return_sequences = True`, and produces `(batch_size, 32, 128)` with enhanced context. (64 units: Sufficient for sequence modeling; Bidirectional: Understands Arabic text order; `return_sequences = True`: Outputs for all slices.)
- **Output**: A sequential representation used later for character prediction. (Prepares for character classification.)
- **LSTM helps understand the order of characters in Arabic text** thanks to bidirectional analysis. (Captures right-to-left context.)

4.5 TimeDistributed Dense

4.5.1 Layer Description

The **TimeDistributed Dense** layer is the final layer in the model, responsible for predicting the character present in each slice of the image. The image is divided into 32 small slices, where each slice may contain part of a character or be empty. This layer examines each slice individually to predict its character.

- **Input**: The layer receives information about 32 slices, each described by 128 features (numbers) that represent its shape. The input shape is `(batch_size, 32, 128)`.
- **Operation**: For each slice, the layer calculates the probability of each possible character (e.g., 60% for "", 30% for ""). The number of possible characters is `num_classes` (including Arabic characters and an empty symbol). The `softmax` function is used to convert the numbers into probabilities.
- **Output**: The layer produces a list of probabilities for each character in each slice. The output shape is `(batch_size, 32, num_classes)`, which is used to form the final word after removing repeated or empty characters.

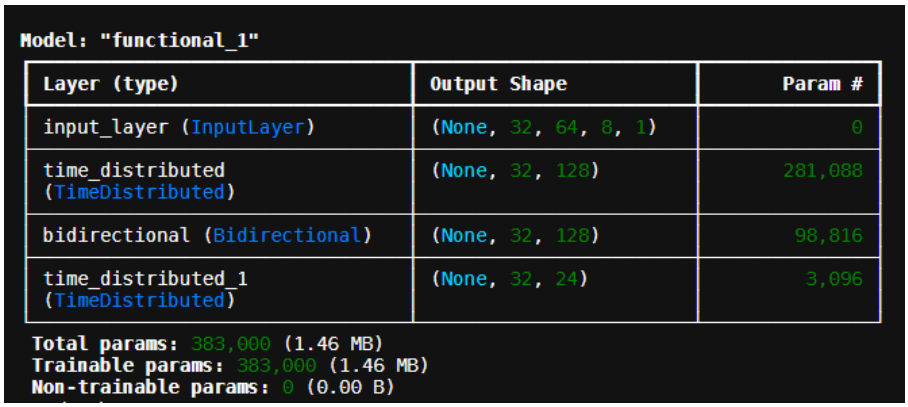
Number of Characters (`num_classes`) : This equals the number of unique characters in the dataset (e.g., Arabic characters) plus an empty symbol for slices that contain no characters, ensuring the model can predict all possible characters.

Softmax Function : It converts numbers into probabilities, making it easy for the model to select the most likely character.

The Functions Used:

- **ReLU**: Transforms negative values to 0, used in Conv2D and Dense to add non-linearity and improve learning.
- **Softmax**: Converts the output into a probability distribution, used in the final layer to predict 24 characters.
- **MaxPooling2D**: Reduces dimensions by selecting the highest value in each 2×2 region, used in CNN to reduce computations.

- **Flatten**: Converts data into a one-dimensional vector, used in CNN to prepare data for the Dense layer.



Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 32, 64, 8, 1)	0
time_distributed (TimeDistributed)	(None, 32, 128)	281,088
bidirectional (Bidirectional)	(None, 32, 128)	98,816
time_distributed_1 (TimeDistributed)	(None, 32, 24)	3,096

Total params: 383,000 (1.46 MB)
Trainable params: 383,000 (1.46 MB)
Non-trainable params: 0 (0.00 B)

Figure 4.4: Model CNN+LSTM

4.5.2 Model Training:

After building the CNN-LSTM model, the training phase was conducted using the prepared dataset. The following settings were used during training:

- **Number of Epochs:** The model was trained for 10 epochs, which means it processed the entire training set ten times. This number strikes a balance between learning performance and avoiding overfitting.
- **Batch Size:** A batch size of 8 was selected. This means the model processes 8 images at a time. This choice is suitable for systems with limited resources and helps stabilize weight updates during training.
- **Loss Function:** The loss function used is `categorical_crossentropy`, which is appropriate for multi-class classification tasks. In this case, the model predicts one character out of several possible ones for each image slice.
- **Optimizer:** The Adam optimizer was used, which combines the benefits of both SGD and RMSprop. It is known for fast convergence and stable performance.
- **Evaluation Metric:** Accuracy was used to monitor the model's performance during training and validation.
- **Validation:** A portion of the dataset was reserved for validation. After each epoch, the model was evaluated on unseen data to monitor generalization and avoid overfitting on training data.

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
1	0.6907	1.3690	0.6991	1.2149
2	0.7115	1.0600	0.7038	1.0167
3	0.7260	0.8966	0.7133	0.9299
4	0.7389	0.7805	0.7230	0.8415
5	0.7532	0.6930	0.7312	0.7864
6	0.7766	0.6122	0.7543	0.6917
7	0.7949	0.5470	0.7592	0.6517
8	0.8113	0.4948	0.7742	0.6083
9	0.8280	0.4424	0.7973	0.5438
10	0.8400	0.4157	0.8011	0.5271

Table 4.1: Training and Validation Metrics for CNN-LSTM Model Across 10 Epochs

- **Analysis of the Model Training Results:**

  Analysis of the model training results over 10 epochs shows a continuous improvement: the accuracy increased from 69.07% to 84.00%, and the loss decreased from 1.3690 to 0.4157 on the training data. On the validation data, the accuracy increased from 69.91% to 80.11%, and the loss decreased from 1.2149 to 0.5271. The model demonstrates good performance, but there is a slight overfitting (the accuracy on the training data is higher than on the validation data).

4.5.2.1 Loss Curve Analysis:

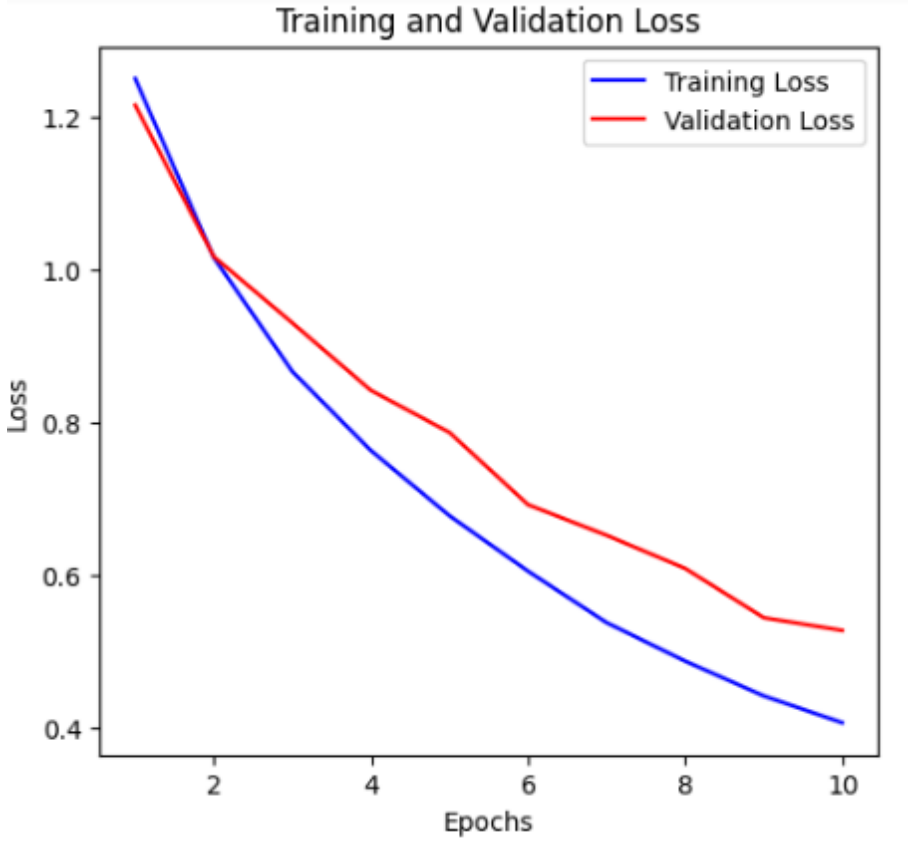


Figure 4.5: Training and Validation Loss Curve Over 10 Epochs

The figure illustrates the evolution of the loss value over 10 epochs for both the training and validation sets.

It can be observed that both the **Training Loss** and **Validation Loss** decrease progressively as training advances, indicating that the model is learning properly from the data.

At the beginning of training, the loss was high (greater than 1.2), but it gradually decreased to around 0.4 for the training set and 0.5 for the validation set.

The gap between the training and validation loss curves remained small, which indicates that the model did not suffer from overfitting and maintained good performance on unseen data.

4.5.2.2 Accuracy Curve Analysis

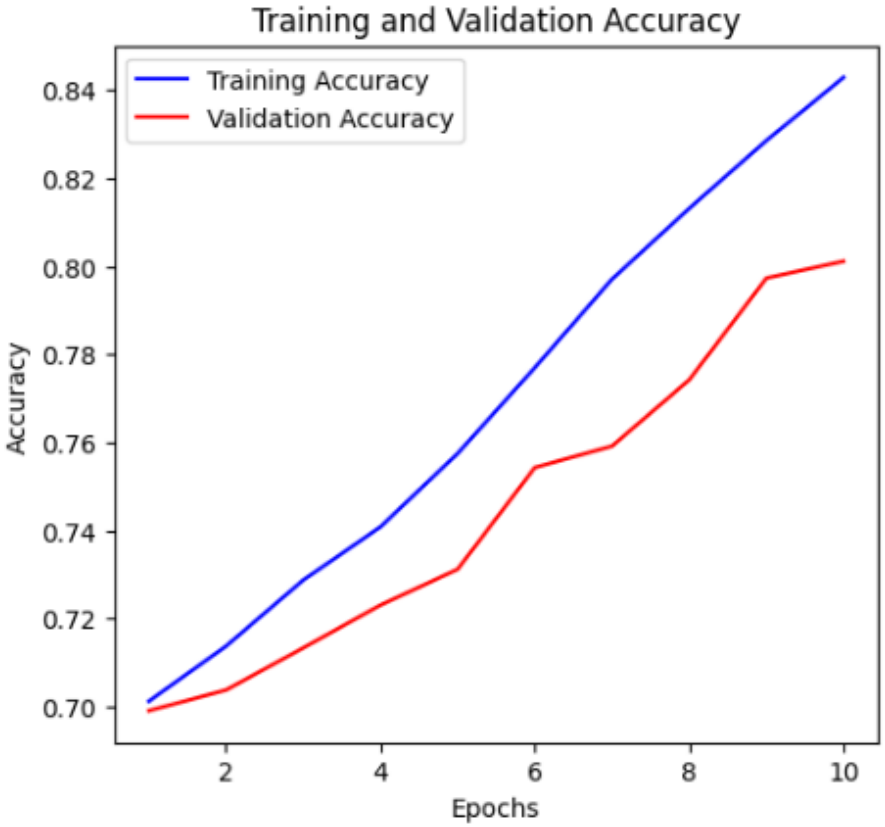


Figure 4.6: Training and Validation Accuracy Curve Over 10 Epochs

The figure illustrates the evolution of accuracy during the training process.

The accuracy of both the training and validation sets increases steadily with each epoch, which is a positive indicator of the model's learning progress.

Initially, the accuracy started at approximately 70%, and it gradually improved to reach:

- Around 84% on the training data
- Around 80% on the validation data

The small gap between the training and validation curves is expected and acceptable. It indicates that the model is performing well on new, unseen data.

Conclusion: The model shows significant improvement in accuracy over time and reaches good performance on both training and validation sets, reflecting the effectiveness of the chosen architecture.

4.5.3 Prediction

- The model generates a probability distribution for each slice, with shape (`batch_size`, 32, `num_classes`).
- The character with the highest probability per slice is selected using `np.argmax` and mapped to characters via `idx_to_char`.
- Duplicate and empty characters are filtered to form the predicted word.
- **Output**: A predicted word per image, representing the recognized text.

4.5.3.1 Word Correction Using N-gram and Levenshtein

Overview

The correction mechanism improves the accuracy of words predicted by the CNN+LSTM model by addressing errors such as missing or substituted characters, using an **N-gram model** to evaluate character sequence likelihood and **Levenshtein distance** to select valid dictionary words close to the predicted word.

Components and Process

- **Dictionary**: Loaded from `class_to_word.txt` (63 words) as a reference for correction.
- **N-gram Model**: Uses $n = 2$ to estimate character sequence probabilities, suitable for the small dictionary size to reduce complexity. Laplace smoothing adds 1 to frequency counts, and a 1×10^{-6} probability is assigned to unseen sequences for numerical stability.
- **Levenshtein Distance**: Measures edits (insertion, deletion, substitution) with a maximum distance of 1, fitting the small dictionary and efficiently handling simple errors.
- **Correction Process**: Retains the predicted word if in the dictionary; otherwise, selects a word within distance 1 based on the highest N-gram score, or keeps the original if no candidates exist.

4.6 Before and After N-gram Correction

4.6.0.1 Evaluation Metrics

The following metrics are used to evaluate the system's performance:

- **Exact Match Accuracy**: The proportion of words predicted exactly correctly. This is a stringent measure of word-level accuracy.
- **Character Error Rate (CER)**: The number of character insertions, deletions, and substitutions needed to correct the predicted text, normalized by the total number of characters in the ground truth. Lower CER indicates better performance.
- **Character Precision**: The proportion of correctly predicted characters out of all predicted characters.
- **Character Recall**: The proportion of correctly predicted characters out of all actual characters in the ground truth.
- **Character F1 Score**: The harmonic mean of Character Precision and Recall, balancing both measures.

In this section, we present a detailed analysis of the model's performance before and after applying the N-gram correction mechanism. The correction process was applied to the outputs of the base CNN+LSTM model, and the results are summarized below.

4.6.1 Quantitative Comparison of Results

Metric	**Before N-gram**	**After N-gram**
Total Number of Words	611	611
Correctly Predicted Words	92	237
Exact Match Accuracy	0.1506	0.3879
Character Error Rate (CER)	0.3492	0.3201
Character Precision	0.8142	0.7361
Character Recall	0.6741	0.7234
Character F1 Score	0.7376	0.7297

Table 4.2: Model performance comparison before and after applying N-gram correction.

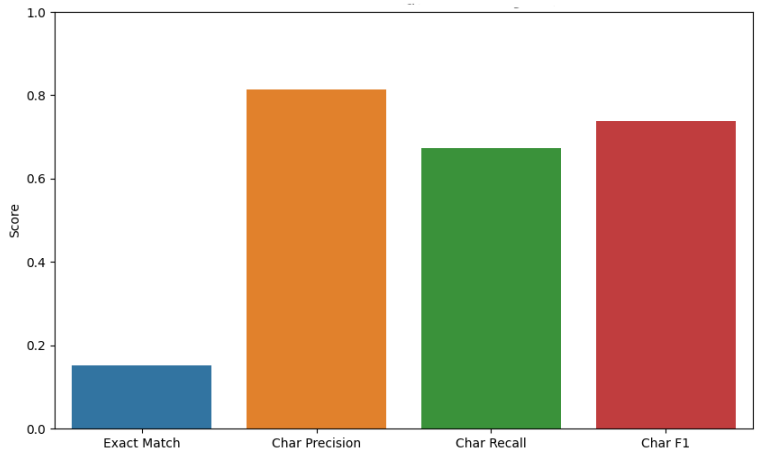


Figure 4.7: Model Evaluation Metrics Before N-gram Correction

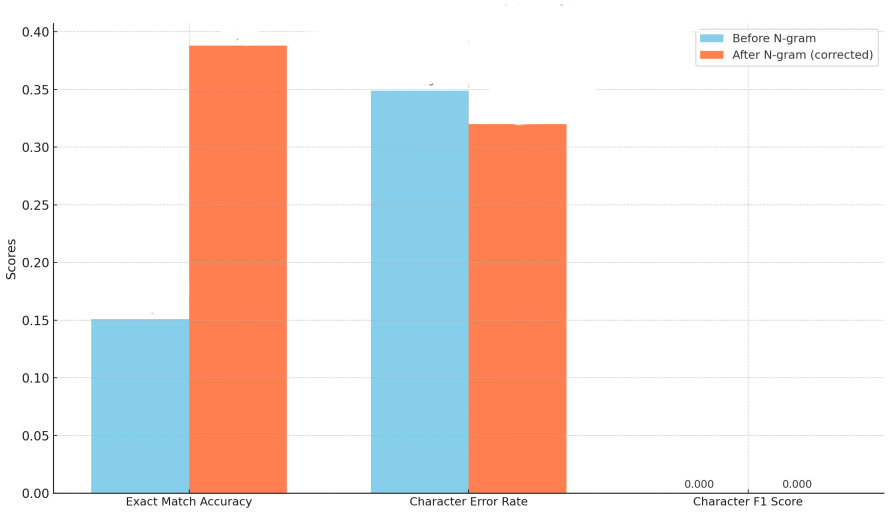


Figure 4.8: Evaluation Metrics Before and After N-gram Correction

4.6.2 Results Analysis

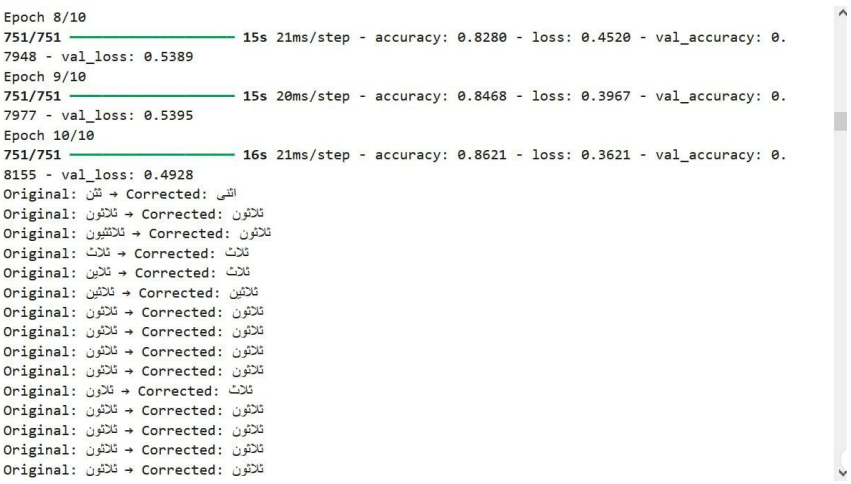
The application of N-gram correction led to a significant improvement in word-level accuracy:

- The number of correctly predicted words increased from 92 to 237, representing an improvement of approximately **157.6%**.
- Exact Match Accuracy rose from 15.06% to 38.79%, indicating better full-word predictions.
- The Character Error Rate (CER) decreased from 34.92% to 32.01%, showing that fewer character-level mistakes remained after correction.

- Although character precision slightly decreased (from 81.42% to 73.61%), character recall improved (from 67.41% to 72.34%), suggesting the corrected model captures more correct characters overall.
- The Character F1 Score remained relatively stable, reflecting a balanced performance.

4.6.3 Additional N-gram Impact Statistics

- Total number of words processed: **611**
- Correct words before N-gram: **92**
- Correct words after N-gram: **237**
- Number of words modified by N-gram: **484**
- Errors successfully corrected by N-gram: **145**
- Correct words turned incorrect: **0**
- Errors changed to other incorrect words: **339**



```
Original: عشرون → Corrected: عشرون
Original: مشين → Corrected: ستين
Original: عشرين → Corrected: عشرين
Original: عشيين → Corrected: عشرون
Original: خمشرين → Corrected: خمسين
Original: عشس → Corrected: عشر
Original: عشرين → Corrected: عشرين
Original: عشرون → Corrected: عشرون
Original: عشرين → Corrected: عشرين
Original: عشرة → Corrected: عشرة
Original: مرن → Corrected: مئة
Original: عشيين → Corrected: عشرون
Original: عشرون → Corrected: عشرون
Original: عشرين → Corrected: عشرين
Original: عشرين → Corrected: عشرين
Original: سئ → Corrected: ست
Original: عشرين → Corrected: عشرين
Original: مشرين → Corrected: عشرون
Original: عشو → Corrected: عشر
Original: عشرونن → Corrected: عشرون
Original: عشرين → Corrected: عشرين
Original: عشرون → Corrected: عشرون
Original: اثن → Corrected: احد
```

- Some test results of the model are presented, where "Original" represents the predictions of the CNN+LSTM model, while "Corrected" refers to the corrections made by the N-gram mechanism to the model's predictions.

4.6.4 Conclusion

The results clearly demonstrate that N-gram post-correction significantly enhances the model's performance at the word level. This improvement is especially valuable in the context of handwritten Arabic word recognition, where full-word accuracy is crucial. The correction module complements the CNN+LSTM model effectively, offering a substantial boost in practical usability.

General Conclusion:

At the conclusion of this thesis, it is important to emphasize that the primary goal of this work was to develop a **prototype model** for recognizing **Arabic handwritten words** using a hybrid framework that integrates Deep Learning (CNN-LSTM) and Natural Language Processing (NLP) techniques, with a correction mechanism based on N-gram models. The focus was deliberately placed on recognizing **words only**, without extending to full sentences or paragraphs, as part of an initial experimental effort to evaluate the effectiveness of the proposed approach.

The obtained results were **promising**, with the model achieving an accuracy of approximately 84%, demonstrating its potential despite the complex challenges of Arabic handwriting, such as connected letters, multiple letter forms, and diverse handwriting styles.

Although the model is still in its early stages, it represents a **first step** toward building a comprehensive system for Arabic handwritten text recognition. Throughout this study, several methods and configurations were explored in order to enhance performance, which provided deeper insights into the practical challenges of this domain.

There is significant room for **future improvement (amélioration future)**, including expanding the scope of the input data, enabling recognition of full sentences, and integrating more advanced language models (such as Transformers). Such developments could further improve the system's accuracy and contextual understanding, making it more suitable for real-world applications in areas such as education, digital archiving, and historical manuscript processing.

Bibliographie

[1] Y. Xu, X. Liu, X. Cao, *et al.*, "Artificial intelligence: A powerful paradigm for scientific research," *The Innovation*, vol. 2, no. 4, 2021.

[2] M. S. Kasem, M. Mahmoud, and H. S. Kang, "Advancements and challenges in arabic optical character recognition: A comprehensive survey," *arXiv preprint*, 2023. arXiv: 2312.11812.

[3] H. Li, "Deep learning for natural language processing: Advantages and challenges," *National Science Review*, vol. 5, no. 1, pp. 24–26, 2018.

[4] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[5] A. Sulaiman, K. Omar, and M. F. Nasrudin, "Two streams deep neural network for handwriting word recognition," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5473–5494, 2021.

[6] A. Bin Durayhim, A. Al-Ajlan, I. Al-Turaiki, and N. Altwaijry, "Towards accurate children's arabic handwriting recognition via deep learning," *Applied Sciences*, vol. 13, no. 3, p. 1692, 2023.

[7] P. Bojja, N. S. S. T. Velpuri, G. K. Pandala, S. D. L. Polavarapu, and P. R. Kumari, "Handwritten text recognition using machine learning techniques in application of nlp," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2019, ISSN: 2278-3075.

[8] M. G. Mahdi, A. Sleem, I. M. Elhenawy, and S. Safwat, "Enhancing the recognition of handwritten arabic characters through hybrid convolutional and bidirectional recurrent neural network models," *Sustainable Machine Intelligence Journal*, vol. 9, pp. 34–56, 2024.

[9] A. Abood, E. AbdElrazek, and A. M. Saad, "Recognizing handwritten arabic characters using deep learning techniques in educational platforms," *Journal of the Faculty of Specific Education, Damietta University*, vol. 2024, no. 9, pp. 324–344, 2024.

[10] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten arabic character recognition," *Cognitive Systems Research*, vol. 50, pp. 180–195, 2018.

[11] N. Alrobah and S. Albahli, "Arabic handwritten recognition using deep learning: A survey," *Arabian Journal for Science and Engineering*, vol. 47, no. 8, pp. 9943–9963, 2022.

[12] I. Ahmad and G. A. Fink, "Handwritten arabic text recognition using multi-stage sub-core-shape hmms," *International Journal on Document Analysis and Recognition (IJDAR)*, 2019, Sample handwritten (above) and machine-printed (below) Arabic texts [Figure].

[13] M. Eltay, A. Zidouri, and I. Ahmad, "Exploring deep learning approaches to recognize handwritten arabic texts," *IEEE Access*, vol. 8, pp. 89 882–89 898, 2020.

[14] F. H. Zawaideh, "Arabic hand written character recognition using modified multi-neural network," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1021–1026, 2012.

[15] M. E. Gumah, *Arabic handwriting recognition: Challenges and solutions*, Unpublished or institution report, 2008.

[16] A. Lawgali, A. Bouridane, M. Angelova, and Z. Ghassemlooy, "Automatic segmentation for arabic characters in handwriting documents," in *2011 18th IEEE International Conference on Image Processing*, IEEE, Sep. 2011, pp. 3529–3532.

[17] Instabase, *Handwriting recognition with ai: Challenges and solutions*, Available at https://instabase.com/blog/handwriting-recognition-ai/, n.d.

[18] Yasmine, *Everyday arabic handwriting*, Transparent Language Blog, Available at https://blogs.transparent.com/arabic/everyday-arabic-handwriting/, 2019.

[19] F. Biadsy, R. Saabni, and J. El-Sana, "Segmentation-free online arabic handwriting recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 07, pp. 1009–1033, 2011.

[20] P. Dreuw, S. Jonas, and H. Ney, "White-space models for offline arabic handwriting recognition," in *2008 19th International Conference on Pattern Recognition*, IEEE, Dec. 2008, pp. 1–4.

[21] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Computing and Applications*, vol. 33, no. 7, pp. 2249–2261, 2021.

[22] A. El-Sawy, M. Loey, and H. El-Bakry, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS Transactions on Computer Research*, vol. 5, no. 1, pp. 11–19, 2017.

[23] U. Pal, R. Jayadevan, and N. Sharma, "Handwriting recognition in indian regional scripts: A survey of offline techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 11, no. 1, pp. 1–35, 2012.

[24] P. Burrow, "Arabic handwriting recognition," Master's thesis, School of Informatics, University of Edinburgh, 2004.

[25] V. Babushkin, H. Alsuradi, M. O. Al-Khalil, and M. Eid, "Analyzing arabic handwriting style through hand kinematics," *Sensors*, vol. 24, no. 19, p. 6357, 2024.

[26] S. Djaghbellou, A. Attia, A. Bouziane, and Z. Akhtar, "Local features enhancement using deep auto-encoder scheme for the recognition of the proposed handwritten arabic-maghrebi characters database," *Multimedia Tools and Applications*, vol. 81, no. 22, pp. 31 553–31 571, 2022.

[27] B. N. Alshahrani and W. Y. Alghamdi, "A deep learning approach to convert handwritten arabic text to digital form," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 5, 2024.

[28] M. Cheriet, "Visual recognition of arabic handwriting: Challenges and new directions," in *Summit on Arabic and Chinese Handwriting Recognition*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–21.

[29] X. Kang, "Dissertation submitted in partial fulfillment of the requirement for the degree of master of science in international business," Master's thesis, 2008.

[30] B. Alabdulkader, "Development of an arabic continuous text near acuity chart," 2017.

[31] M. T. Parvez and S. A. Mahmoud, "Arabic handwriting recognition using structural and syntactic pattern attributes," *Pattern Recognition*, vol. 46, no. 1, pp. 141–154, 2013.

[32] U. Porwal, A. Fornes, and F. Shafait, "Advances in handwriting recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 25, no. 4, pp. 241–243, 2022.

[33] W. AlKendi, F. Gechter, L. Heyberger, and C. Guyeux, "Advancements and challenges in handwritten text recognition," *Journal of Imaging*, vol. 10, no. 1, p. 18, 2024.

[34] P. Yadav and N. Yadav, "Handwriting recognition system-a review," *International Journal of Computer Applications*, vol. 114, no. 19, pp. 36–40, 2015.

[35] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, vol. 21, 2008.

[36] R. Kala, H. Vazirani, A. Shukla, and R. Tiwari, "Offline handwriting recognition using genetic algorithm," *arXiv preprint*, 2010. arXiv: 1004.3257.

[37] X. Y. Zhang, Y. Bengio, and C. L. Liu, "Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark," *Pattern Recognition*, vol. 61, pp. 348–360, 2017.

[38] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 787–808, 1990.

[39] A. Al-Salman and H. Alyahya, "Arabic online handwriting recognition: A survey," in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, Oct. 2017, pp. 1–4.

[40] A. Fouillé, *Digital image processing: Introduction*, INPG - LIS - Jocelyn Chanussot, 42 pages, 2005.

[41] M. Sarfraz, "Introductory chapter: On digital image processing," in *Digital Image Processing*, M. Sarfraz, Ed., IntechOpen, 2020. DOI: 10.5772/intechopen.9228.

[42] I. T. Young, J. J. Gerbrands, and L. J. Van Vliet, *Fundamentals of image processing*. Delft: Delft University of Technology, 1998, vol. 841.

[43] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 4th. Pearson, 2008.

[44] C. Liu, "A review of digital image processing techniques and future prospects," *International Journal of Computer Science and Information Technology*, vol. 4, no. 3, 2024. DOI: 10.62051/ijesit.v4n3.22.

[45] P. Lakshmi Devi and N. Subash, *Digital image processing (r15a0426): Lecture notes*, B.Tech IV Year - I Sem, Department of Electronics and Communication Engineering, Malla Reddy College of Engineering & Technology, Telangana, India, 2019.

[46] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[47] R. Kundu, *Image processing: Techniques, types, & applications*, https://www.v7labs.com/blog/image-processing-guide, Aug. 2022.

[48] G. Kumar and P. K. Bhatia, "A detailed review of feature extraction in image processing systems," in *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, IEEE, Feb. 2014, pp. 5–12.

[49] A. C. Goswami, "Image processing techniques and challenges: A review paper," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 2, 2022.

[50] G. Saady, *Image processing (level 4)*, PDF, Uploaded by Bakhtyar Mahmood on July 10, 2024, 2024.

[51] J. Nelson, *Synergy between ai and deep learning*, 2024.

[52] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2014.

[53] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine learning and deep learning approaches for cybersecurity: A review," *IEEE Access*, vol. 10, pp. 19 572–19 585, 2022.

[54] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: An overview," in *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, 2021, pp. 599–608.

[55] A. I. Károly, R. Fullér, and P. Galambos, "Unsupervised clustering for deep learning: A tutorial survey," *Acta Polytechnica Hungarica*, vol. 15, no. 8, pp. 29–53, 2018.

[56] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *arXiv preprint*, 2020. arXiv: 2006.05278.

[57] K. Chowdhary and K. R. Chowdhary, "Natural language processing," in *Fundamentals of Artificial Intelligence*, 2020, pp. 603–649.

[58] E. D. Liddy, *Natural language processing*, 2001.

[59] I. Hafeez Sodhar and A. Hafeez Buller, "Natural language processing: Applications, techniques and challenges," in *Advances in Computer Science*, AkiNik Publications, 2020, pp. 1–25. DOI: 10.22271/ed.book.784.

[60] B. Akram, *Tokenization and text preprocessing in nlp*, https://www.linkedin.com/pulse/tokenization-text-, Jun. 2024.

[61] GeeksforGeeks, *Pos (parts-of-speech) tagging in nlp*, https://www.geeksforgeeks.org/nlp-part-of-speech-default-tagging, Jan. 2024.

[62] C. W. Schmidt, V. Reddy, H. Zhang, *et al.*, "Tokenization is more than compression," *arXiv preprint*, 2024. arXiv: 2402.18376.

[63] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," *arXiv preprint*, 2020. arXiv: 2003.00104.

[64] P. Bala, *Handwritten character recognition and evaluation*, Apr. 2025.

[65] R. P. Sunita, *Natural language processing (nlp) and understanding*, 2025.

[66] A. Lahreche, "Deep learning for arabic letters recognition," Master's thesis, University Kasdi Merbah Ouargla, Faculty of New Technologies of Information, Communication, Department of Computer Science, and Information Technologies, 2019.

[67] M. G. Mahdi, A. Sleem, I. M. Elhenawy, and S. Safwat, "Hybrid neutrosophic deep learning model for enhanced arabic handwriting recognition," *Neutrosophic Sets and Systems*, vol. 72, pp. 446–465, 2024.

[68] S. Hamida, O. El Gannour, B. Cherradi, H. Ouajji, and A. Raihani, "Efficient feature descriptor selection for improved arabic handwritten words recognition," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 5, 2022, ISSN: 2088-8708.

[69] M. Elsayed, A. Alnaggar, M. Abdeen, A. Wahdan, and W. Gomaa, *Arabic handwritten text recognition using advanced cnn-rnn architecture*, Unpublished or institution report, 2023.

[70] M. E. Mustafa and M. K. Elbashir, "A deep learning approach for handwritten arabic names recognition," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 678–682, 2020.

[71] M. El-Melegy, A. Abdelbaset, A. Abdel-Hakim, and G. El-Sayed, "Recognition of arabic handwritten literal amounts using deep convolutional neural networks," in *Pattern Recognition and Image Analysis: 9th Iberian Conference, IbPRIA 2019, Madrid, Spain, July 1–4, 2019, Proceedings, Part II*, Springer International Publishing, 2019, pp. 169–176.

[72] T. M. Ghanim, M. I. Khalil, and H. M. Abbas, "Comparative study on deep convolution neural networks dcnn-based offline arabic handwriting recognition," *IEEE Access*, vol. 8, pp. 95 465–95 482, 2020.

[73] M. Amrouch, M. Rabi, and Y. Es-Saady, "Convolutional feature learning and cnn based hmm for arabic handwriting recognition," in *Image and Signal Processing: 8th International Conference, ICISP 2018, Cherbourg, France, July 2-4, 2018, Proceedings*, Springer International Publishing, 2018, pp. 265–274.

[74] S. Khosravi and A. Chalechale, "Recognition of persian/arabic handwritten words using a combination of convolutional neural networks and autoencoder (aecnn)," *Mathematical Problems in Engineering*, vol. 2022, no. 1, p. 4 241 016, 2022.

[75] N. Zermi, M. Ramdani, and M. Bedda, "Arabic handwriting word recognition based on hybrid hmm/ann approach," *International Journal of Soft Computing*, vol. 2, no. 1, pp. 5–10, 2007.

[76] X. Li, "An effective approach to offline arabic handwriting recognition," *International Journal of Artificial Intelligence and Applications*, vol. 4, no. 6, 2013.

[77] A. Mars and G. Antoniadis, "Arabic online handwriting recognition using neural network," *International Journal of Artificial Intelligence and Applications*, vol. 7, no. 5, pp. 51–60, 2016.

[78] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey of the foundations, selected improvements, and some current applications," *arXiv preprint*, 2020. arXiv: 2011.12960.

[79] S. U. Masruroh, M. F. Syahid, F. Munthaha, A. T. Muharram, and R. A. Putri, "Deep convolutional neural networks transfer learning comparison on arabic handwriting recognition system," *JOIV: International Journal on Informatics Visualization*, vol. 7, no. 2, pp. 330–337, 2023.

[80] A. Benchabana, "Conception et mise en œuvre d'une plateforme de traitement d'images satellitaires et aériennes pour la mise à jour des objets spatiaux d'un système d'information géographique (sig) destiné au suivi des changements urbains," Doctoral dissertation, Université Chahid Hamma Lakhdar d'El-Oued, 2024.

[81] T. Höser, "Global dynamics of the offshore wind energy sector derived from earth observation data-deep learning based object detection optimised with synthetic training data for offshore wind energy infrastructure extraction from sentinel-1 imagery," Doctoral dissertation, Julius-Maximilians-Universität Würzburg, 2022.

[82] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[83] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019.

[84] S. Jiang, S. Qin, J. L. Pulsipher, and V. M. Zavala, "Convolutional neural networks: Basic concepts and applications in manufacturing," in *Artificial Intelligence in Manufacturing*, Academic Press, 2024, pp. 63–102.

[85] D. Chenna, "Evolution of convolutional neural network (cnn): Compute vs memory bandwidth for edge ai," *arXiv preprint*, 2023. arXiv: 2311.12816.

[86] P. Knap, K. Lalik, and P. Bałazy, "Boosted convolutional neural network algorithm for the classification of the bearing fault form 1-d raw sensor data," *Sensors*, vol. 23, no. 9, p. 4295, 2023.

[87] M. S. Ahmad, "Deep learning 101: Lesson 18: Image data in machine vision," *Medium*, Sep. 2024. [Online]. Available: https://medium.com/@muneebsahmad/deep-learning-101-lesson-18-image-data-in-machine-vision.

[88] D. Bhatt, C. Patel, H. Talsania, *et al.*, "Cnn variants for computer vision: History, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, p. 2470, 2021.

[89] A. M. AlShareef, M. Alnowaimi, and M. Siddig, "Cnn-based detection of welding crack defects in radiographic non-destructive testing," in *Saudi International Conference On Nuclear Power Engineering*, Cham: Springer Nature Switzerland, Nov. 2023, pp. 45–57.

[90] M. D. Rahman, "Deep learning neural networks-based edge detection in mechanical components," Master's thesis, Itä-Suomen yliopisto, 2021.

[91] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[92] K. Abdelouahab, "Reconfigurable hardware acceleration of cnns on fpga-based smart cameras," Doctoral dissertation, Université Clermont Auvergne, 2018.

[93] J. Ayeni, "Convolutional neural network (cnn): The architecture and applications," *Applied Journal of Physical Sciences*, vol. 4, pp. 42–50, 2022.

[94] C. R. A. Kumar, A. Akhil, and C. S. Kumar, "Ai & ml-based pet feeding system," 2025.

[95] M. T. Redjati and K. Rachid, "Diagnostic des défauts de roulement basé sur les réseaux de neurones bilstm," Master's thesis, Université Badji Mokhtar - Annaba Repository, 2023.

[96] H. Okut, "Deep learning for subtyping and prediction of diseases: Long-short term memory," in *Deep Learning Applications*, IntechOpen, 2021.

[97] M. E. H. Boukhatem and I. Makrerougrass, "Reconnaissance de l'écriture manuscrite avec les réseaux de neurones," Université Abdelhamid Ibn Badis, Mostaganem, Faculté des Sciences Exactes et d'Informatique, Département de Mathématiques et Informatique, Tech. Rep., 2022.

[98] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[99] Rajasekhar, "Deep learning-lstm rnn," *Medium*, 2025. [Online]. Available: https://medium.com/@rajasekhar.2319/deep-learning-lstm-rnn-bi-directional-rnn-6e9ac4413f19.

[100] F. Landi, L. Baraldi, M. Cornia, and R. Cucchiara, "Working memory connections for lstm," *Neural Networks*, vol. 144, pp. 334–341, 2021.

[101] F. Silfa, J. M. Arnau, and A. González, "Boosting lstm performance through dynamic precision selection," in *2020 IEEE 27th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, IEEE, pp. 323–333.

[102] C. B. Vennerød, A. Kjærran, and E. S. Bugge, "Long short-term memory rnn," *arXiv preprint arXiv:2105.06756*, 2021.

[103] J. Houvardas and E. Stamatatos, "N-gram feature selection for authorship identification," *Artificial Intelligence: Methodology, Systems, and Applications*, vol. 4183, pp. 77–86, 2006.

[104] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, vol. 161175, p. 14, 1994.

[105] D. E. Hamla, "Handwritten digit recognition using encryption methods," Master's thesis, University Larbi Tébessi - Tébessa, Faculty of Exact Sciences et al., 2022.

[106] GeeksforGeeks, *Levenshtein distance*, `https://www.geeksforgeeks.org/levenshtein-distance-in-python/`, 2024.