الجمهوريــة الجزائريــة الديمقراطيــة الشعبيــة
République Algérienne Démocratique et Populaire
و البـحث الـعلـمـي وزارة الـتـعلـيـم الـعـالي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université MUSTAPHA Stambouli
Mascara

جامعة مصطفى اسطمبولي
معسكر

Faculté des Sciences Exactes
Département de l'informatique
Laboratoire de physique quantique de la matière et modélisation mathématique

THESE de DOCTORAT

Spécialité : Modélisation et optimisation des systèmes

Intitulée

Contribution à la reconnaissance hors ligne de l'écriture arabe manuscrite

Présentée par **:** Chadli Mohamed Amine

Le 14/07/2025

Devant le jury :

Président	**BOUFERA Laila**	**Professeure**	**Université de Mascara**
Examinateur	**Rahmani El Hadi**	**MCA**	**Université de Saida**
Examinateur	**Sahraoui Mustapha**	**MCA**	**Université de Mascara**
Encadreur	**Bachir Bouiadjra Rochdi**	**MCA**	**Université de Mascara**
Co-Encadreur	**Fekir Abdelkader**	**MCB**	**Université de Mascara**

Année Universitaire : 2024-2025

Contribution to Offline Recognition of Arabic Handwritten Text

ملخص

يُعد التعرف الآلي غير المتصل على النصوص العربية المكتوبة بخط اليد (HTR) مجالًا بحثيًا مهمًا. وقد يكون للمساهمة في تطويره آثار كبيرة ومثيرة للعديد من الدول الناطقة بالعربية، حيث تُعد العربية اللغة الرسمية للمحادثة والكتابة.

لقد تناولت هذه الرسالة دراسة اللغة العربية المكتوبة بخط اليد، وخصائصها، والتحديات التي تواجه التعرف على خطها المكتوب. وقمنا بإجراء مراجعات أدبية شاملة للطرق الكلاسيكية والحديثة في التعلم العميق للتعرف على النصوص العربية. كما تعمقنا في دراسة الموارد المستخدمة لتدريب أنظمة التعرف على النصوص العربية المكتوبة بخط اليد، محاولين استخراج جميع قواعد البيانات المعروفة للنصوص العربية المكتوبة يدويًا. بالإضافة إلى ذلك، قمنا بالتحقيق في أساليب زيادة البيانات المستخدمة لإثراء قواعد البيانات للنصوص العربية المكتوبة بخط اليد ببيانات جديدة.

وفي مرحلة لاحقة، اقترحنا بنية جديدة للتعلم العميق تعتمد على الشبكة العصبية الالتفافية المتكررة (CRNN) للتعرف الآلي غير المتصل على النصوص العربية المكتوبة بخط اليد. وكمساهمة أخيرة، قمنا باقتراح تقنية جديدة لزيادة البيانات تعتمد على طريقة المربعات الصغرى المتحركة (MLS)، مصممة خصيصًا لتوليد صور جديدة للنصوص العربية المكتوبة بخط اليد لتكون بمثابة بيانات تدريب لأنظمة التعلم العميق.

Abstract

Offline Arabic Handwritten Text Recognition (HTR) is an important research area. Contributing to its development may have significant and exciting implications for many Arabic-speaking countries, where Arabic is the official spoken and written language.

This thesis has examined the Arabic handwritten language, its characteristics, and the challenges of recognizing its written script. We conducted thorough literature reviews on classical and modern deep learning methods for recognizing Arabic script. We delve deep into the resources used to train the Arabic HTR systems, trying to extract all the known databases of Arabic handwritten text. We also investigated the data augmentation approaches used to enrich the handwritten Arabic text databases with new data.

Later, we proposed a new deep learning architecture based on Convolutional Recurrent Neural Network (CRNN) to recognize offline Arabic scripts. As a final contribution, we proposed a new data augmentation technique based on Moving Least Squares (MLS), specifically designed to generate new images of Arabic handwritten text to serve as training data for deep learning systems.

Acknowledgment

"And my success is not but through Allah. Upon him I have relied, and to Him I return." Verse 88 from surah Hud.

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Bachir Bouiadjra Rochdi, for all his support and patience during my journey. He was such an incredible person. He supported me during my lowest points and never gave up on me when I was stuck, even when circumstances made him unavailable. I thank him for guiding me in getting a scholarship abroad. I would also like to extend my heartfelt thanks to my co-supervisor, Dr. Fekir Abdelkader. He is such an admirable person. He never let me down when I needed support. He was so patient with me. He tried his best to guide me to fulfill my full potential.

I thank both my parents for giving me all the support I needed. I appreciate their patience, and I will be in debt to them all my life. There are no words to express all my gratitude towards them for everything they gave me, and all the sacrifices they made for me to become the man I am. Hopefully, I wish for them to live a happy, peaceful, fruitful life, and I hope that I can live to pay just a fraction of what they gave to me.

I thank the University of Mustapha Stambouli and the University of Castilla–La Mancha for granting me a scholarship to pursue my studies and experience life abroad in Spain.

I thank Prof. José Antonio Gámez Martín for welcoming my stay in his research laboratory "Intelligent Systems and Data Mining" (S.i.M.D.), and for providing everything I needed, and facilitating my stay in the research lab. He truly made my life easy. He treated me like one of his students. I sincerely felt at home.

I will always remember Dr. Jesus Martínez-Gómez. I was fascinated by his approach to tackling new research problems. He is indeed a world-class researcher. His guidance truly marked a dramatic shift in my research. He made me realize how much space for improvement is available. I enjoyed reading his reviews. He truly inspired me.

A special thanks to my uncle, Prof. Djilali, for providing me with help whenever I needed it. He was truly a pillar of support for me. He stands out for me during my hardest times.

I will always remember my friends who stood by me in times of need. Their support meant the world to me.

Table of Contents

List of Figures

List of Tables

1. Chapter 01: Introduction

1.1. Text Recognition

The research field of pattern recognition was first motivated by the need to recognize texts present in images. Text recognition is the ability to transcribe texts present in images into a machine-readable format. Different languages have different characteristics. Each language is characterized by its writing direction. Some languages are written from left to right, like Latin languages, while other languages are written from right to left, like Arabic and Urdu, and some of the East Asian languages are traditionally written vertically in columns from top to bottom and ordered from right to left. The texts written in any language can be either printed using a machine or written by hand. In some languages, the writing style of printed texts is very different from their handwritten counterpart. Printed Latin texts are often noncursive, which means the letters corresponding to the same word are not connected, and there is a minor gap or space between each character. Latin texts written by hand follow a different rule of writing, where all characters of the same word are connected. This way of writing is named cursive writing, and it is used to make writing faster. The Arabic language is a semi-cursive language, which means that for each word, some characters are connected, and other characters are not. The semi-cursive nature of Arabic is consistent for both printed and handwritten text.

There are two main approaches to handwritten text recognition. The first, named offline recognition, and it deals with texts that were previously written and that have been captured and stored digitally. In this case, the text will be transformed into a digital format using a camera or a digital scanner and after that sent to a recognition system.

The second approach, on the other hand, consists of recognizing handwritten text in real time, at the very moment of writing. This method is generally used on digital devices such as smartphones, touchscreens, tablets, or devices specially designed for digital writing. The user writes either directly with their fingers, as on a smartphone, or with a digital pen. Real-time recognition requires optimized computing power and powerful recognition systems, especially when the device has limited computing capabilities.

The difficulty of the task of offline Arabic handwritten text recognition resides in the variability and diversity of writing styles among different writers. Writing styles can even vary from the writings of the same writer. We can observe a slight difference between copies of the same word written by the same writer.

1.2.History

The first attempts to develop text recognition systems date back to the 1960s, driven by the US military, which was seeking to automate the translation of scientific articles written in languages other than English, such as Russian, as mentioned in [1].

In that era, computers came with punchers, which are devices that punch holes in stiff papers with high precision on specific locations determined by a human who operates the machine by pressing some keys. This process of entering text was not only time-consuming but also expensive, requiring a lot of specialized operators. Faced with these limitations, the search for an automated method to read and transmit texts to the computer became necessary to facilitate machine translation while reducing costs and processing time.

By 1963, approximately 100 optical character reading machines were already in use in the commercial sector [2]. This number increased considerably, and by the 1970s, hundreds of these machines were operating successfully. However, they were primarily designed for a narrow and highly standardized set of characters.[5] The cost of these machines, ranging from several tens of thousands to $1.5 million, was a considerable sum even by today's standards.

The first attempts to recognize handwritten text started with the recognition of isolated characters, that was written by users in predefined places (locations) for each character, in a well-defined writing style, that is similar to printed letters, instead of a cursive writing style. This writing style, named handprinted writing, was designed to make all the characters uniform and easily distinguishable, consequently making the recognition phase easier than the recognition of unconstrained handwritten texts. Many OCR (optical character recognition) committees around the world have worked to create standards for handprinted characters, aiming to reduce variations in form to improve recognition rates [6]. An example of handprinted alphanumeric characters, developed in 1974 by the OCR committee of the American National Standards Institute (ANSI),

is illustrated in Figure 1.1. Writers of these characters are trained to reproduce the letters as closely as possible to the standard models provided, ideally using mechanical pencils with fine HB leads. In general, each country has its handprinted standard, but no standard for handprinted Arabic characters has been made public, and therefore, no research on handprinted Arabic character recognition exists in the literature.

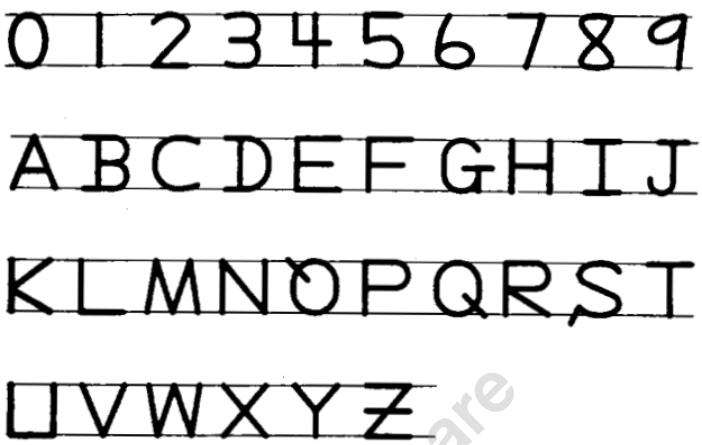


Figure 1.1: Alphanumeric characters as defined by the American handprint standard [6].

Optical readers of that era consisted of several key steps: character segmentation, preprocessing, feature extraction, and classification (identification). Feature extraction, in particular, was distinguished by the use of three broad categories of techniques: global features, distribution of points, and geometrical and topological features, according to [6].

All the above-mentioned advances illustrate the characteristics of the first era of text recognition, which can be summarized as follows: systems were designed to recognize isolated characters only, and exclusively in standardized printed or handprinted styles. Unable to handle a wide variety of styles and variations, these systems remained limited.

A new era started to take shape at the beginning of the 1990s, driven by intensive research in the field of text recognition. Throughout this period, Hidden Markov Models (HMMs) became the main research approach to develop text recognition systems [10], making the recognition of character sequences easier than ever before, and consequently, the recognition of entire words, and even lines of text. This progress made it possible to make the recognition of cursive handwritten text more efficient. Inspired

by the successes of automatic speech recognition, which, like text recognition, is based on sequences of data, the adoption of HMMs opened new perspectives.

Around the 2010s, text recognition reached a new milestone with the rise of neural networks [11]. Although the theoretical foundations of neural networks had been established long before, their widespread application had been hampered by a lack of computing power and the scarcity of data acquisition devices. With the explosion of the Internet, access to massive and varied data has stimulated this adoption. This context, coupled with the increased availability of powerful computing resources, has allowed neural networks to establish themselves as the new gold standard in text recognition.

One of the aspects that makes this approach interesting is its way of blending all the different steps of the process (preprocessing, feature extraction, segmentation, and classification) into just one single end-to-end neural network system, thus eliminating the complexities of passing through each step of the classical approaches.

1.3.Challenges of Text Recognition

Image quality is crucial for text recognition, whether printed or handwritten. Issues such as blurring or uneven lighting can seriously hamper the effectiveness of recognition systems. Indeed, when lighting varies within the same image, it causes disparities in pixel intensity, complicating the interpretation of the text.

Printed text can also be put on top of complex backgrounds (patters, textures), which makes the recognition even harder. This complexity increases in public spaces, where text can appear on marketing displays, road signs, and vehicle license plates. These complex scenes with diverse colors, objects and perspectives make the task very hard.

In addition, printed text is often oriented irregularly, with rotations at different angles. It can also adopt complex structures, such as texts nested inside containers with specific shapes or superimposed on other elements, typical of works of art, decorations, or advertising panels of marketing campaigns.

The use of various fonts within a single document, combined with stylistic variations such as bold, italics, or underlining, further complicates recognition.

These challenges related to the recognition of printed text are found in all languages, including Arabic.

The recognition of handwritten text is complicated by a multitude of factors, including the writing instrument used and the surface on which the text is written. In addition, the infinite variations in letterforms, due to factors such as the writing habit, personal style, level of education, region of origin, or the social environment of the author, add to the complexity. Added to this are more circumstantial factors, such as the mood or state of health of the writer at the time of writing, which also influence the form of the handwritten text [4].

1.4. Arabic Text Recognition

1.4.1. Characteristics of Arabic Script

Arabic is the official language in all Arab countries with 255 million speakers. Arabic world is the fifth most spoken language in the world. It is used in most writings and, orally, in official or informal situations (television news, administrative, religious, political). It should also be noted that Farsi (Persian), used mainly in Iran and Afghanistan, shares many characters with the Arabic script.

The Arabic language is technically distinguished from other languages in writing on several points. One of the most important differentiations comes from its semi-cursive in both its handwritten and printed form. A cursiveness that is manifested in the links between letters in the formation of words. This technical aspect of connection in writing is found in the same way in handwritten and printed writing. In addition, the notions of capital letters and lowercase letters do not exist in Arabic writing. The development of computer science has enriched printed Arabic, now, there are more than 450 different styles and fonts. This richness is also demonstrated in the change of the morphological characteristics of the Arabic character, from one font to another. In this respect, some interesting and detailed studies on the morphological characteristics of Arabic are carried out in these researches [7, 8, 9].

Like Syriac, Hebrew, N'Ko, Adlam, or ancient Hungarian, Arabic is written from right to left, in a way where the letters are mostly connected. Similarly, their spelling differs depending on whether they are preceded and/or followed by other letters or whether they are isolated (contextual variants). In Arabic writing, the characters align on what is called the baseline. The baseline is a horizontal virtual line that all the connected parts of a text are present on top of it. We can notice that the maximum number of black

pixels of text is present alongside the baseline. Characters of a word are compared to the French alphabet, Arabic has 2 more letters (28 letters), and it also owes its richness to the ability of its letters to change shape depending on whether they appear isolated, at the beginning, middle, or end of the word. Now, some letters take up to four different forms: for example, the letter (ع) (has four forms of appearance: isolated «ع», at the beginning «عـــ», in the middle «ـعـ», and at the end «ـــع». Table 1.1 summarizes the different forms of appearance in a word, that the 28 Arabic letters can take.

Table 1.1: The different shapes of Arabic characters in all word position (isolated, beginning, middle, and ending o the word).

Character	End form	Middle form	Beginning form	Isolated
Alif	ـا	ـا	ا	أ
Ba	ـب	ـبـ	بـ	ب
Ta	ـت	ـتـ	تـ	ت
Tha	ـث	ـثـ	ثـ	ث
Jeem	ـج	ـجـ	جـ	ج
Hha	ـخ	ـخـ	خـ	خ
Kha	ـح	ـحـ	حـ	ح
Dal	ـد	ـد	د	د
Thal	ـذ	ـذ	ذ	ذ
Ra	ـر	ـر	ر	ر
Zay	ـز	ـز	ز	ز
Seen	ـس	ـسـ	سـ	س
Sheen	ـش	ـشـ	شـ	ش
Ssad	ـص	ـصـ	صـ	ص
Dhad	ـض	ـضـ	ضـ	ض
Tta	ـط	ـطـ	طـ	ط
Ttha	ـظ	ـظـ	ظـ	ظ
Ain	ـع	ـعـ	عـ	ع
Ghain	ـغ	ـغـ	غـ	غ
Fa	ـف	ـفـ	فـ	ف
Qaf	ـق	ـقـ	قـ	ق
Kaf	ـك	ـكـ	كـ	ك
Lam	ـل	ـلـ	لـ	ل
Meem	ـم	ـمـ	مـ	م
Noon	ـن	ـنـ	نـ	ن
Ha	ـه	ـهـ	هـ	ه
Waw	ـو	ـو	و	و
Ya	ـي	ـيـ	يـ	ي

Letters that have just two forms of appearance cannot be linked to the following letter; their starting form is simply their isolated form, and their middle form is exactly the

ending form. But for most letters, the start/middle and ending/isolated forms are identical except for the ligature. The presence of a ligature with the preceding letter or with the following letter does not change the shape of the letter significantly. In Arabic, ligatures are always located at the level of the writing line (baseline), that is to say that there is no high-link letter like the 'o' or the 'v' in the Latin alphabet. It is noted that six letters of the Arabic alphabet are never attached to their successor: (ا, د, ذ, ر, ز, و). These letters introduce a break in the word, hence the notion of sub-words. The sub-word is made up of one or multiple connected characters., Making the words made up of one or several sub-words. A sub-word is then a connected component of black pixels grouping one or more letters (see Figure 1.2). An example of a word made up of multiple sub-words is the word elaalam (العالم) (see Figure 1.3), which consists of three sub-words: alif (ا), which is made of just one isolated character, laa (لعا), which is made up of three connected characters, and lam (لم), which is made up of two connected characters. Indeed, the distinction between word and sub-word is made by the length of the inter-word space, which is generally greater than the intra-word space.

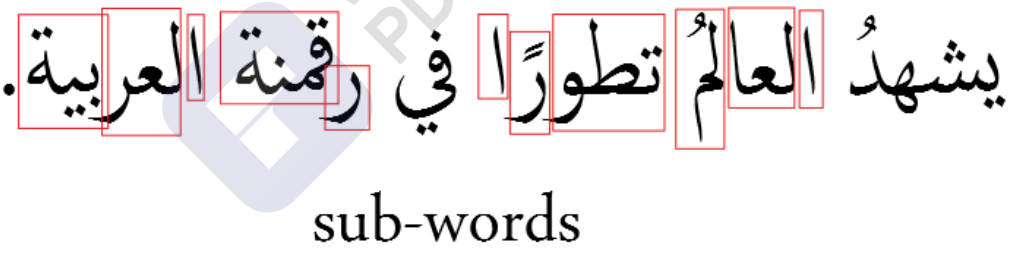


Figure 1.2: A sample of Arabic words featuring sub-words

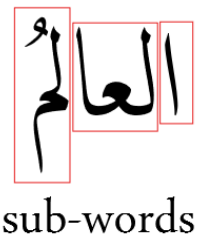


Figure 1.3: An example of a word composed of three sub-words

A technical problem arises when it comes to handwriting. Now, the spacing between the different pseudo-words of the same word is not necessarily systematically greater

than the spacing between two different words, which sometimes poses problems in segmenting the text. This degree of complexity is somewhat alleviated by the fact that when one of the 22 letters (28 minus the 6 that do not connect with the next one) appears in its “end of word” or “isolated” form, this necessarily means that we are reaching the end of a word.

Another feature of Arabic is that the characters of a word sometimes overlap vertically without touching each other (See Figure 1.4). This vertical character overlap happens to some characters that come immediately after each other.

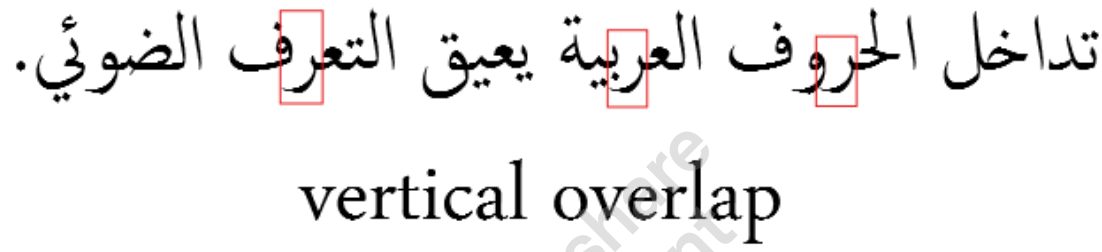


Figure 1.4: Examples of vertical character overlap without touching.

Sometimes, at the beginning of a text line, the first characters of the first word are stacked vertically on top of each other to form what is called a ligature. Ligature is often found in typeset and in handwritten texts (See Figure 1.5).

Figure 1.5: Examples of Arabic ligatures formed through various character combinations.

Diacritics:

Diacritical marks are auxiliary signs that are added to Arabic letters to indicate either that it is a vowel, or dots to distinguish letters, or other signs like (a-chadda, al-madda, al-hamaza) to indicate a change in pronunciation of the letter.

a) Dots Diacritics:

Figure 1.6: An illustration of various dot diacritic shapes.

The Arabic alphabet is composed of 28 letters, where 15 letters have dots to distinguish between the letters. Ten letters have only one dot, three letters have two dots, and only two letters have three dots. The dots can be either put above the letter (ex: ف ,ش ,ت), below the letter (ex: ي ,ب), or in the middle of the letter (ex: ج). If it weren't for these dots, a lot of letters would have the same shape (Figure 1.6 presents the different dot diacritic shapes, while Figure 1.7 shows examples of dot diacritics on their corresponding characters).

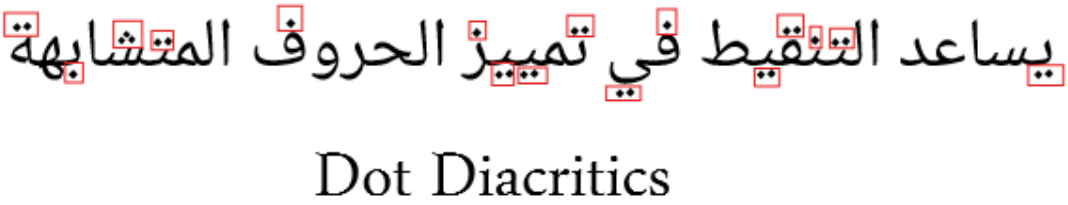


Figure 1.7: Examples of Arabic diacritics formed by dots.

Furthermore, in Arabic handwritten text, the two dots can be written as two connected dots like a line or a stroke. The three dots can be written in different ways, like one dot plus a line or a stroke, two connected strokes in a shape of a triangle, or it can be written as a curve (See Table 1.2).

Table 1.2: Examples of Arabic letters with varying dot shapes for the same character.

Character	Different dot shapes
ت	ت ـتـ ـت
ش	ش ـشـ ـشـ ـش

b) Vowel Diacritics:

Figure 1.8: An illustration of all short vowel diacritics.

Arabic writing is distinguished from the Latin alphabet by its systematic absence of vowels. To compensate for this absence and ensure accurate reading, Arabic scripts use diacritical marks. These marks, such as al-fatha, al-dammah, and al-kasrah, indicate the nature of the vowels and thus modify the pronunciation and meaning of words. All Arabic letters, except the alif, are consonants. The waw and ya have a phonetic particularity, being able to function as both consonants and vowels. The absence of systematic vocalization can generate homographic polysemy. The same assembly of consonants can correspond to several distinct words, depending on the vocalization. Diacritical marks, placed above like al-fathah (ـَ), al-dammah (ـُ), and al-sukun (ـْ) or below the letters like al-kasrah (ـِ), help to remove these ambiguities and clarify the pronunciation. The tanwīn (تنوين) is an Arabic diacritic that indicates nasalization at the end of a word. It occurs in three forms, each corresponding to a nasalized vowel: the tanwīn al-fatḥa (ـً), which indicates nasalization on the vowel a, the tanwīn al-ḍamma (ـٌ), which indicates nasalization on the vowel o, and the tanwīn al-kasra (ـٍ), which indicates nasalization on the vowel e (Figure 1.8 illustrates the shapes of short vowel diacritics, while Figure 1.9 provides examples of these diacritics on characters). Although diacritical marks are essential for rigorous phonetic transcription, they are not systematically used in all texts. Arabic readers, accustomed to this practice, infer the

missing vowels from the context. Thus, two types of texts are distinguished: those entirely vocalized, such as the Quran or educational works, and those partially or not vocalized, such as newspapers and contemporary publications. In current Arabic, writing focuses mainly on consonants and long vowels, leaving it to the reader to reproduce the short vowels.

Figure 1.9: Examples of short vowel diacritics in Arabic characters

c) Other Diacritics:

Other diacritical marks include al-hamza, a-chadda, and al-madda (See Figure 1.10 for an illustration of these diacritics). A-chadda, placed above a consonant, indicates a gemination, that is, a double consonant, comparable to the accentuation of certain consonants in French. Al-hamza, for its part, represents a glottal, a sound produced at the root of the throat. Its use is governed by complex morphological and syntactic rules, linked in particular to the root of the words and their phonetic environment. Finally, al-madda is a lengthening of the preceding vowel and is often associated with the letter alif. It plays an important role in Arabic metrics.

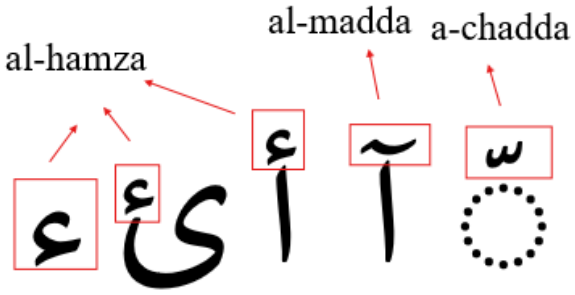


Figure 1.10: An illustration of the al-hamza, a-chadda, and al-madda diacritics.

Ligature:

The Arabic script doesn't have an ascending link between the adjacent characters like the Latin script. However, ligatures in Arabic form unique vertical links between the adjacent letters. These ligatures, composed of two to four characters, form complex shapes that make their segmentation into individual characters very difficult. Additionally, the characters forming the ligature overlap vertically, which complicates the segmentation even more.

1.4.2. Challenges of Offline Handwritten Arabic Text Recognition

When writing in Arabic, writers often make mistakes and don't write some letters correctly. Or poorly connect the characters, by connecting characters that are not supposed to be connected, or not connecting characters that are supposed to be connected. Or sometimes forgetting to add some diacritics, or even putting the wrong diacritics by mistake. The writers can also poorly space between the inter-words and intra-words. Where the inter-word space is supposed to be always greater than the intra-word space. But writers of Arabic often poorly space between them. When handwritten texts are poorly connected and poorly spaced, as explained previously, it leads to some serious segmentation challenges. Identifying the boundaries between words becomes much harder. It is already hard to segment words in Arabic because of the semi-cursive nature of the language, even without introducing the challenges of handwritten texts. Even humans, who are theoretically the best readers, will make around 4% mistakes when reading texts in the absence of context [3]. The writers also sometimes introduce some unintentional noise to the text, such as blots or smudges or even tears, which can interfere with the recognition.

1.5. Applications of Text Recognition Systems

Text recognition has many use cases and applications. Some of these use cases are purely academic, others have potential commercial or industrial use cases, and other use cases are found in the educational sector. Some use cases have social welfare applications. Others have a direct relation to the security.

In the academic sector, one of the well-known use cases of text recognition is the digitization of historical documents. Millions of ancient documents and books are written and preserved throughout the history of mankind in various languages, such are Latin and Arabic. These ancient documents are often written in much different writing style and shape than the modern writing styles, thus they are hard to read and interpret, which make very few expert trained people able to read and interpret them, thus it comes the necessity to digitize those documents automatically to make them accessible for researchers to research them.

In the banking sector, banks and similar entities deals with high throughput volumes of data found in the format of checks and invoices, which necessitate the automatic reading of these checks and invoices, which led to the availability of countless commercial Optica Character Recognition (OCR) systems to perform these tasks.

In offices, the digitalization of documents to make them searchable and indexable is one use case, the automatic extraction and processing of forms filled by hand by customers is another use case.

In a post office, some systems help in process automation by automatically sorting the post mails after automatically reading the address or the zip code found on top of the mail post or the package.

In the education sector, children can participate in learning sessions to learn writing and spelling and so on using digital devices.

Social welfare applications to aid the visually impaired individuals cope with their disabilities through reading machines that can be integrated with smartphones or with augmented reality glasses.

At last, but not least, the governments can have law enforcement applications such as automatically reading vehicle plate numbers to keep the streets safe, either by tracking outlaws or simply keeping track of the movements of the vehicles.

1.6. Thesis Structure Overview

This thesis structure is organized in four main chapters, each one of these chapters explore and study one main aspect of the offline recognition of Arabic handwritten text.

- **Chapter 01: Introduction**
  In the introduction we introduce the domain of research where we explain text recognition in general, and we dive into the history of text recognition, and later we explain the challenges of text recognition, emphasizing on the specific challenges of the recognition of Arabic handwritten texts, but first we describe the characteristics of Arabic scripts, and we finally list the potential applications and use cases of Arabic handwritten text recognition systems.
- **Chapter 02: Literature Review**
  This chapter is composed of three parts: 1) the Data Collection, Augmentation, and Evaluation part; 2) the Classical Text Recognition Approaches part; and 3) the Neural Networks Approaches to Text Recognition part. In the first part, we explain a process we created to create new Arabic handwritten text databases. We show the evaluation metrics used in the field of text recognition. We have also presented comprehensive reviews of data augmentation techniques for Arabic handwritten texts and the available Arabic handwritten text databases. In the second part, we have explained the process of text recognition using the classical approaches alongside all the different techniques and methods used in the literature in each step of the process. In the third part, we studied all the recent research to extract all the deep learning approaches of offline Arabic handwriting text recognition.
- **Chapter 03: Offline Arabic Handwritten Text Recognition for Unsegmented Words using Convolutional Recurrent Neural Network**
  In this chapter, we present a new deep learning architecture that recognizes offline Arabic handwriting texts, where we describe the methodology in detail and show and discuss the experimental results.

- **Chapter 04: Data Augmentation for Offline Arabic Handwritten Text Recognition Using Moving Least Squares**

  In this chapter, we present a new data augmentation technique for Arabic handwritten texts, developed to help enrich databases with more training data. The newly proposed technique is extensively tested and compared to show its effectiveness. This chapter represents the core contribution of this thesis.

1.7.Contributions

The main contributions of this thesis are as follows:

1. A comprehensive review of all available data augmentation techniques that are used to augment handwritten Arabic text images is listed in Section 2.1.3.
2. A comprehensive review of all the available Arabic handwritten text image databases listed in Section 2.1.4.
3. A comprehensive review of neural network approaches to Arabic text recognition in Section 2.3.
4. The design of a new deep learning offline Arabic Handwriting Text Recognition architecture in Section 3.
5. The development of a new data augmentation technique for Arabic Handwritten Text images in Section 4.

2. Chapter 02: Literature Review

2.1.Data Collection, Augmentation, and Evaluation

2.1.1. Dataset Creation Process

The creation of datasets is pivotal for developing machine learning models, particularly for tasks involving handwritten text recognition. This section outlines a streamlined process encompassing data collection, form design, acquisition, labeling, verification, preprocessing, and storage. These steps are crucial for ensuring dataset diversity and relevance, addressing the complexities of scripts like Arabic. The aim is to provide a systematic approach that enhances the accuracy and reliability of recognition systems, thereby advancing the field.

1) Data Collection and Sampling Strategies:

The choice of the text to be written by writers to create a new dataset, and the choice of the writers themselves is essential to the dataset creation because it determines the quality and the representativeness of the dataset, as more diverse writers from as many regions with as many educational and cultural backgrounds writing text lines or text words of as many topics maximizing the pool of words present in the dataset can significantly make the dataset appealing and statistically representative for both the writing style diversity and the words diversity making the recognition systems relying on this dataset more robust.

a) Writer Sampling Strategies:

There are many factors to sample candidates to participate in the writing of a dataset, and there are some considerations, like whether the dataset is intended for general-purpose use cases or domain-specific use cases. For instance, if the need is to create general-purpose text recognition systems, factors such as age range, education level, and cultural backgrounds must be maximally diversified, but if the need is to create text recognition systems for children-related applications, a sampling pool of individuals with younger age ranges is more appropriate.

b) Text and Words Sampling Strategies:

As in the author sampling step, the selection of the text lines or the words to be written by the authors to create a new dataset is crucial. And again, it also depends on the use

case, for either it is for general-purpose use cases or domain-specific use cases. As in the former case, it is advised to include common words from as many topics as possible, and in the latter case, include as many words from the topic of interest as possible. Another thing to consider is the length of the text line if the dataset intends to create recognition systems that specialize in the recognition of text lines, whether to use long text lines or short text lines. One more thing to consider is the intended audience, for which there are many different dialects for each region, and each dialect is characterized by its own set of common words and spellings. For this reason, the selection of the dialects to sample words from is essential for a good design of a dataset.

2) Form Design (Layout Design):

When creating new datasets of handwritten texts or words, we must find a way to tell the potential authors, what they must write, and where to write it. As the location of the writing is important for later steps. For this, a good layout form design is crucial for a successful data collection operation. As it will serve as an interface between the designers of the dataset and the authors. Some good criteria to conduct a good data collection operation are to make the form as clear, simple, self-explanatory, and self-descriptive as possible. The form needs to make the act of writing on it smooth with minimal intervention of the people in charge of the operation. Since often hundreds or thousands of authors are involved in the data collection operation, and some organizers make the data collection operation remote by sending the forms by post to the authors, there is no one physically available to ask about the form. The design of the layout often must have three blocks to satisfy the criteria mentioned earlier, it must have a guideline block, a block containing the text or the word to be written, and an empty block in which the writing should happen.

3) Data Acquisition and Retrieval:

After the writing is done by the authors, a way is needed to digitize these writings. A common way to digitize them is by scanning them with digital scanners. Although, in our times at the moment of writing this thesis, scanning can be made by smartphones due to the advances in the speed and quality of the photography of the smartphones, but digitizing in this way is only advised when creating small-scale datasets. A professional digital scanner is quicker and can handle a large number of documents when the need is to create a large-scale dataset. Digitizing the handwritten documents is not the final

step. A further step is to extract the relevant text lines or words from the scanned document. As often, a form contains several writing blocks to write many different text lines or words. The extraction can either be done manually using tools like Paint or Photoshop, or automatically by determining beforehand the block location coordinates of the handwritten text line or word by identifying the form with its associated identification number, knowing that each identification number determines the layout design and the content of each form. Finally, each extracted text line or word is saved separately in a digital format with its relevant details.

4) Labelling:

Labelling (or annotation) is a major step in the dataset creation process. When creating handwritten text datasets, we typically assign ground truth information details to each text image. This step is mandatory because labelled text images are needed in both the training and the evaluation of text recognition systems. The information details that need to be associated to each text image usually include the sequence of characters or words present in the image, and in some cases, additional information such as the coordinates of individual characters (for segmentation tasks) or writer statistical demographic information (e.g., gender, age, education level).

The design of the form tremendously influences the labelling step, because the layout of the form determines the ease of filling the form for the participants, and also determines how effective and accurate the process of extracting the labels from the form. This means that a form that was properly designed with clear guidelines that indicate the target text and where to write it, notably makes the labelling task easier. For instance, knowing the exact locations of the writing blocks makes the process of extracting the text images automatic and precise.

Labelling can be performed either manually or automatically. In the manual approach, data entry specialists transcribe the handwritten text present in the filled forms into a digital format. Although this method is very accurate, it is time-consuming and labor-intensive, making it not suitable for large-scale datasets. On the other hand, the labelling process can be automated using dedicated software. For the automation to be possible, the form design must be aligned with the labelling software. By taking advantage of the predefined structure of the form, the software can extract sub-images from specific locations in the form and associate these text images with their corresponding ground

truth text/ labels. The automation of the labelling process significantly reduces the time and effort needed to label the text images compared to the manual labeling process.

5) Data verification:

When creating new datasets of handwritten text recognition, it is necessary to establish a vigorous data verification process to guarantee that the dataset is as reliable as possible. The verification process is split into three phases, each phase tackles a different attribute of the data.

The first phase concerns form level verification, where we check for indications of poor quality or incompleteness of the form. It often consists of examining the form to see if there are some cropped parts or some missing paragraphs, and evaluating the readability of the handwritten texts. After the examination, we classify the filled forms into three categories: readable, challenging, and unreadable. We discard the unreadable forms, and only keep the readable and challenging forms to be included in the dataset.

The second phase is responsible of checking if the extracted paragraph and line images truly corresponds to their respective numbers, as each paragraph and line image has a predefined number as described previously.

The third phase of the verification process deals with the verification of the ground truth labels, where a human operator manually checks the written text against the ground truth labels to make sure that the dataset is reliable.

This step is made possible through the involvement of multiple reviewers to ensure the integrity of the dataset.

6) Pre-processing:

Preprocessing is another important step for the creation of handwritten text datasets, as we need standardized datasets, and we can achieve such standardized datasets by applying some preprocessing techniques such binarization, to convert grayscale images into binary images, smoothing techniques such as Gaussian filtering, and noise removal.

Image normalization is also very common when creating handwritten text datasets, because we often need images with consistent size and resolution such as 128x128 or 128x32 pixels. It is also common to apply skew correction to correct the baseline of a

skewed image due to scanning misalignment or handwrites habits of tilting the text when writing, making the extracted text lines horizontal and ready to be recognized or segmented.

Sometimes some kind of segmentation can be applied on the extracted text depending on the use case of the dataset. For example, if the extracted text is a paragraph, a segmenting it to text lines is very common, and in some other cases such as in Arabic, it is sometimes necessary to segment the text lines into sub-words, which is needed in some text recognition applications, where the aim is only to recognize sub-words.

Additionally, contrast enhancing techniques may be applied to make all the extracted images uniform. Such light differences can happen because of the different scanning conditions and scanning tools used to scan the forms.

7) Data storage:

Datasets are stored in a digital format using standards formats such as TIFF, BMP, an PNG. Ground truth data is often stored in an XML format, which allow the storage of annotations in a structured way.

Datasets are also organized into subdirectories (categories), to facilitate the retrieval of the data. There is often a folder for training, validation, and testing data.

Metadata is also stored alongside the data that includes information about the writer such as his identity, gender, age, education level, geographic information and so on.

2.1.2. Evaluation Metrics

Evaluating the performance of the text recognition systems is critical. Thus, a proper evaluation scheme is important. There are two main evaluation schemes in text recognition. The first evaluation scheme is for the text recognition systems that recognize isolated characters or isolated words among multiple characters or words given the image of interest. Thus, it's a multi-class classification task. The second evaluation scheme is for the text recognition systems that recognize sequences of characters.

2.1.2.1. Isolated Characters/ Words Evaluation Metrics

In the context of character or isolated word recognition, the problem is a multiclass classification, where each character or word represents a distinct class. The system

performance is evaluated using a confusion matrix, a C×C table (where C is the number of classes) that crosses the model predictions with the real annotations.

- **True Positives (TP):** Number of characters/words correctly identified as belonging to their target class.
- **True Negatives (TN):** Number of characters/words correctly excluded from a given class (i.e., belonging to other classes and not predicted as this one).
- **False Positives (FP):** Number of characters/words incorrectly assigned to a class (i.e., belonging to other classes but predicted as this one).
- **False Negatives (FN):** Number of characters/words missed by the model (i.e., belonging to one class but predicted as another).

Confusion matrix:

(C = number of character classes/words)

	Predicted: Class 1	Predicted: Class 2	…	Predicted: Class C	Actual Total
Actual: Class 1	TP_1	$FP_1 \rightarrow 2$	…	$FP_1 \rightarrow C$	$\sum row\ 1$
Actual: Class 2	$FP_2 \rightarrow 1$	TP_2	…	$FP_2 \rightarrow C$	$\sum row\ 2$
…	…	…	…	…	…
Actual: Class C	$FP_C \rightarrow 1$	$FP_C \rightarrow 2$	…	TP_C	$\sum row\ C$
Predicted Total	$\sum column\ 1$	$\sum column\ 2$	…	$\sum column\ C$	N

Where:

- $\boldsymbol{TP_a}$ (True Positives): Number of instances correctly recognized as Class a.
- $\boldsymbol{FP_a \rightarrow b}$ (False Predictions): Instances of Class a incorrectly predicted as Class b (where a ≠ b).
- $\boldsymbol{\sum row}$: Total number of actual instances for a class.

- $\sum \boldsymbol{column}$: Total number of predicted instances for a class.
- **N**: Total number of samples.

And:

- **Diagonal** (TP_1 to TP_C): Correct recognitions.
- **Off-diagonal** ($FP_a \rightarrow b$): Classification errors (confusions between classes).
- **False Negatives** (FN_a) for a class a: Sum of $FP_a \rightarrow b$ in row a.
- **False Positives** (FP_a) for a class a: Sum of $FP_a \rightarrow b$ in column a.

The following metrics are then calculated:

1. **Accuracy:** Ratio of correct predictions (TP + TN) to the total number of samples. It measures the overall performance of the system:

   $$Accuracy = \frac{\sum_{i=1}^{C} TP_i + TN_i}{\sum_{i=1}^{C} (TP_i + TN_i + FP_i + FN_i)} \quad (1)$$

2. **Precision** per class**:** The model's ability to avoid false detection errors for a specific class:

   $$Precision_i = \frac{TP_i}{TP_i + FP_i} \quad (2)$$

3. **Recall** per class**:** The model's ability to correctly identify all instances of a class:

   $$Recall_i = \frac{TP_i}{TP_i + FN_i} \quad (3)$$

4. **F1-Score** per class: Harmonic mean of precision and recall, balancing the two metrics:

   $$F1 - Score_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Reall_i} \quad (4)$$

These metrics are calculated for each class and then aggregated. The accuracy show the general correctness of the system, and precision, recall, and F1-score provide additional insight on the quality and quantity of the relevant predictions for each class.

2.1.2.2. Sequence-based Evaluation Metrics

In the context of sequence-based text recognition, whether character sequences or words, it is essential to have quantitative methods to evaluate the performance of recognition systems. We use Levenshtein (edit distance) to calculate the distance

between the ground truth text and the predicted text. We can create four evaluation metrics based on this edit distance:

1. **Character Error Rate (CER):** CER measures the percentage of errors (insertions, substitutions, and deletions) compared to the total number of characters in the target text. When the CER is low, the recognition is more accurate:

   $$CER = \frac{I+S+D}{N} \times 100 \qquad (5)$$

2. **Word Error Rate (WER):** WER measures the percentage of errors at the word level rather than at the character level. It measures the performance of whole-word recognition systems:

   $$WER = \frac{I+S+D}{N} \times 100 \qquad (6)$$

3. **Character Recognition Rate (CRR):** CRR, also called **Character Accuracy Rate (CAR)**, represents the percentage of characters that are correctly recognized:

   $$CRR = 1 - CER \qquad (7)$$

4. **Word Recognition Rate (WRR):** WRR, also known as **Word Accuracy Rate (WAR)**, measures the percentage of words that are correctly recognized. It is used when we want to evaluate the performance of word recognition systems:

   $$WRR = 1 - WER \qquad (8)$$

Where I, S, and D represent the minimum number of edits (insertions, substitutions, and deletions) that change a ground truth text string into a predicted text string, and N is the total number of characters or words in the ground truth text. These metrics provide a comprehensive and nuanced assessment of the performance of text recognition systems.

To calculate the average metrics (CER, WER, CRR, and WRR), each metric must be evaluated for each pair of source and target texts and then averaged over all pairs. After measuring the errors (insertions, substitutions, deletions) and applying the formulas, the results are summed and divided by the total number of pairs.

2.1.3. Data Augmentation

Data augmentation plays a significant role in improving the performance of Arabic HTR systems when there is a scarcity of large annotated databases. Small databases are not enough to represent much of the writing styles found in the written Arabic language. Thus, data augmentation presents itself as a promising approach to address this limitation and help with the development of robust models that have better generalization. The diversity of training data can be increased by incorporating data augmented with traditional image processing techniques and morphological operations. This allows for the creation of new data that looks like natural variations in handwriting styles. Techniques such as rotation, shifting, zooming, shearing, erosion, dilation, and noise addition are widely used to create a more comprehensive and realistic training set. Moreover, advanced techniques such as Generative Adversarial Networks (GANs) have the potential to generate synthetic data that looks like real-world data. These data augmentation techniques can deal with limited amounts of data and to deal with uneven class representation. In this subsection, we present a comprehensive review of all the available data augmentation techniques that were used to augment Arabic handwritten text. We have found that most of the research has focused only on traditional augmentation techniques, with few exceptions, where they developed complex data augmentation techniques.

In the context of offline Arabic handwritten text recognition, the authors in this work [12] generated a synthetic dataset comprising 500,000 images of printed Arabic text lines, to which they applied various image processing techniques such as shearing, rotation, distortion, erosion, and compression, to increase the diversity and realism of the images. The models were first trained with the synthetic dataset generated by these data augmentation techniques to improve their generalization. Later, they were fine-tuned on the KHATT dataset. The results showed that pretraining the models with these augmented printed Arabic texts improved their performance, with a 0.93% decrease in the character error rate for the Transformer Transducer, and 0.85% for the cross-attention Transformer. This suggests that combining such transfer learning techniques with the proposed data augmentation methods can be an effective strategy to improve the performance of Arabic HTR systems.

Ashiquzzaman et al [13] proposed a data augmentation strategy for handwritten Arabic numeral recognition. They applied several techniques, such as ZCA-whitening, random rotation up to 10 degrees, horizontal and vertical shift up to 20% of the image dimensions, and random zoom up to 10%. They applied this strategy to augment the images of the CMATERDB dataset [18], which consists of 3000 handwritten Arabic digit images. This strategy helped increase the robustness of their CNN model, and it achieved an accuracy of 99.4%, outperforming the previous models that did not use data augmentation.

Balaha et al. [14] used a public database named "HMDB" [118] that is composed of 54,115 images of isolated handwritten Arabic characters in different word positions. The images in this database have a fixed size of 32x32 pixels. The total number of classes in this database is 115, representing the different Arabic characters in different word positions. The authors used simple augmentation techniques to augment the data of this database to improve the performance of their CNN models. They specifically applied rotation, shifting, zooming, and shearing. The authors trained 14 different CNN models with different architectures, and they selected the best-performing model with an accuracy of 91.96%. Their experiments showed that these augmentation techniques significantly improved the performance of their models compared to the models that were trained without data augmentation.

Wagaa et al. [15] proposed a new CNN architecture to recognize isolated handwritten Arabic characters. They used two Arabic databases to train and test their model, AHCD and Hijja databases. AHCD database [119] is composed of isolated handwritten Arabic letters in the isolated position of the word, and the Hijja database [120] is composed of isolated handwritten Arabic characters in different positions of the word. They applied some augmentation techniques to improve the performance of the model by taking advantage of image processing techniques, such as rotating the images, shifting, flipping, zooming, and injecting noise into them, including Gaussian noise, Salt-and-pepper noise, and Speckle noise. They observed that the performance would be significantly improved if they rotated the images by 10 degrees and shifted them just by one pixel. An accuracy of 98.48% was reached for the AHCD database, and 91.24% on the Hijja database.

Rabi et al. [16] proposed a data augmentation approach to augment the IFN/ ENIT database based on a Generative Adversarial Neural Network (GAN). This GAN was developed using a ScrabbleGAN model, and it generates highly realistic and diverse Arabic handwritten text images similar to the text images present in the IFN/ ENIT database. They tested the performance of this technique by incorporating the augmented images into the training data of a CNN-BLSTM-based Arabic HTR model. The augmented data helped the model to surpass the baseline model by 3.54%, which is a significant improvement, demonstrating the ability of GAN networks to help increase the performance of Arabic HTR systems.

Ahamed et al. [17] applied some data augmentation techniques to augment the data of the CMATERdb 3.3.1 dataset [18], which is composed of images of Arabic digits. They applied morphological operations, specifically dilation, erosion, opening, and closing. The database went from 3,000 original samples to 72,000 samples. This augmentation technique helped increase the performance of the CNN model, which achieved 99.76% accuracy.

Mohamed Eltay et al. [19] presented an innovative way to augment handwritten Arabic text images. They developed a Generative Adversarial Network (GAN) that smartly augments the text images. Their GAN takes into consideration the dataset imbalances found in the Arabic handwritten text databases, where they used an adaptive strategy to generate text images that are well balanced in terms of character frequency, where their technique generates more images for the less represented characters from the databases they used for training and augmentation. They experimented with their method with the IFN/ENIT and AHDB databases. The results showed significant improvement in the recognition accuracy of the systems that are trained with these balanced augmented data when compared to the recognition systems that are trained with standard methods.

Mohamed Eltay et al. [20], worked again on the class imbalances problem found in all Arabic handwritten text databases. They developed a novel adaptive data augmentation algorithm that produces augmented images based on word weights, in which the algorithm calculates and assigns weights to each word of the database based on the average probability of each class in the word. This algorithm promotes class diversity and helps produce more balanced databases. The experiments were carried out on the IFN/ENIT and AHDB databases, and the results achieved state-of-the-art performance.

Sulaiman et al. [21] introduced two novel data augmentation strategies. Their strategies are inspired by human writing habits. In the first strategy, they rotate subsections of the image, trying to emulate some human writing habits, where the writers sometimes tend to deviate from the baseline. In the second strategy, they apply different thickness stress on various parts of the image, trying again to emulate some human writing habits, where some writers apply different stress on the letters when writing, making the thickness of writing differ from one letter to another in the same word or text line. The proposed strategies were applied and tested in several languages, including Arabic.

2.1.4. Available Arabic databases

The recognition of handwritten Arabic text is very challenging because of the nature of the language and how it is written in different regions and by different writers, and because of the extensive use of ligatures, which makes the recognition even more difficult. To design and develop robust recognition systems for Arabic handwritten text, good-quality Arabic handwritten text databases must be used. For this reason, we present in this section a comprehensive review of all the available databases of Arabic handwritten text/ characters. We examine various quantitative and qualitative aspects of each database, such as the diversity of words/ characters, the diversity of writers, the demographics of writers, and the number of samples. We discuss famous databases, such as IFN/ENIT, KHATT, HACDB, and others, to assess their quality, where they are potentially lacking, and what their impact is on the ongoing research. In this review, we try to provide a comprehensive understanding of the current state of Arabic handwritten databases and point toward potential limitations and gaps that may be affecting the generalization and the performance of the recognition systems developed on top of these databases.

1) The IFN/ENIT database:

The IFN/ENIT database [22] is a well-known Arabic handwritten text database, most used for offline recognition of Arabic handwritten text. This database is developed by the German Institute for Communications Technology (IFN) and the Tunisian school Ecole Nationale d'Ingénieurs de Tunis (ENIT). It is composed of 26,459 images of handwritten words of Tunisian town/ village names written in Arabic by 411 writers. The images are stored alongside their ground truth information, including the sequence

of characters constituting the words present in the image, baseline estimation, and some writers metadata.

The participants filled out forms with predefined town names and their corresponding postcodes. They were scanned at 300 dpi, with automatic word extraction, automatic baseline estimation, and automatic pre-labeling. They then manually verified the labels of the extracted words to ensure consistency between the filled forms and their electronic version. The designers of this database also made it publicly available for research purposes upon request.

This database also includes 10 ligatures that are frequent in handwritten Arabic texts. These ligatures cannot be found in the printed Arabic texts. The pool of writers who participated in writing the town names is mostly limited to students of the Tunisian ENIT.

This database represents a great contribution to the field of offline recognition of Arabic handwritten text, thanks to its robust character-level annotations, even though it is limited to Tunisian town names.

This database is publicly available upon request for research purposes.

2) The KHAAT Database:

The KHATT database is one of the best resources for Arabic handwritten text. 1000 distinct writers participated in writing the forms of this database. Each one of them writes and fills out a form. The demographics of the writers are very diverse, with variations in age, gender, education level, and geographical origin. The forms are scanned with multiple scanning resolutions (200, 300, and 600 dpi). The forms contain three types of text: the first text is a fixed minimal text that covers all Arabic character shapes in all positions of the word inspired by HUSNI A. et al [121], the second text is randomly selected from a large corpus, unique to each form, and the third text is up to the writer to write anything about open subjects freely. The design complexity of the KHATT database makes it suitable for various research applications, such as text recognition, writer identification, and form analysis.

The KHATT database is an open vocabulary resource, in contrast to other databases, such as the IFN/ENIT database, where it contains a limited domain-specific vocabulary.

making it best suited for research that involves real-world scenarios where text variability is high.

The database is divided into training, validating, and testing sets (70%, 15%, and 15%, respectively). A verification step to ensure data integrity was performed on the form, paragraph, and line levels.

This database is publicly available worldwide for researchers.

3) The HACDB Database:

Lawgali et al. introduced a database named the Handwritten Arabic Characters Database (HACDB) [24], which is a database specifically composed of the 28 isolated Arabic letters in the different shapes of the various positions of a word (isolated, beginning, middle, and end). This database also includes some overlapping characters.

The HACDB database was developed to play the role of a standardized dataset for training and testing Arabic handwritten character recognition systems. The database covers 52 basic shapes of Arabic characters and other overlapping characters. The characters of this database don't contain any dots or diacritics, as they were removed. Around 50 writers of different ages helped write this database, with each writer filling out two forms, each form containing 66 different character shapes. The database is composed of 6,600 character images.

The forms were scanned at 300-dpi resolution, and they used software tools to automatically extract the character shapes and create an image for each character shape. The authors provided two versions of this database, the version one for training and testing Arabic handwritten character recognition systems with each other, and the second one is used for recognizing words after character segmentation.

The database was standardized by applying some preprocessing techniques, such as binarization, noise removal, and resizing to 128x128 image pixel size.

This database is available for academic purposes.

4) Jawad Alkhateeb Database:

Jawad H AIKhateeb et al. presented an Arabic handwritten database for isolated letters [25]. This database is composed of 28,000 images of Arabic letters. This database contains only isolated Arabic letters in their isolated shape. Around 100 writers from

various backgrounds helped write the letters for this database, including academic staff, university students, and high school students. Each writer was given a form to write each letter of the 28 letters 10 times.

The images were preprocessed by applying normalization, median filtering, noise removal, and resizing them. The database is organized into folders, each containing the images of a character. It was divided into training and testing sets (80% for training and 20% for testing).

This database was intended to be freely available.

5) The AHDB Database:

The AHDB (Arabic Handwritten Data Base) database developed by Somaya et al. [26] is a database written by 100 writers. It is composed of three types of text images: text images of isolated words, including text images of sixty-seven handwritten words of numbers that are used in cheque writing, and text images of the twenty-nine most common Arabic words. The second type of text is sentences, including text images of sentences of words, numbers, and quantities that can be written on cheques. The third type of text images is texts written freely by the choice of the writer on any subject. The total number of text images contained in this database is not revealed.

All the text images of this database have been through a preprocessing step, where slope correction, slant correction, thinning, and width normalization were applied to these text images to normalize them.

A high-resolution Hewlett-Packard 6350 scanner was used to scan all the forms at a 600 dpi resolution, and the images were stored in TIFF file format.

This database doesn't seem to be publicly available.

6) The NDAHR Database:

The NDAHR database developed by Abdalkafor et al. [27] is very rich and interesting. It covers Arabic isolated characters in their isolated form, it covers the words of the days of the week, it covers numbers from 1 to 10 written in their letter form, it covers currency words, it covers words of the unit of measurements, it covers Arabic digits, Indian digits, it covers special symbols, it covers mathematical symbols and operations, it covers Arabic diacritics, it covers all sorts of brackets and punctuation marks. This

database is composed of a total of 123 different shapes. It was written by 100 distinct writers. Two groups of writers have participated in writing, including academic staff and university students, government employees, and primary and secondary schools. Each character/ word or symbol is written 5 times by each writer, making the database a total number of samples around 61500 images.

There is no sign of public availability of this database.

7) Nawwaf Database:

Nawwaf et al. [28] developed a database written by five hundred randomly selected students from Al-Isra University in Amman, Jordan. This database is composed of a curated list of words that carefully covers all Arabic characters in all their word positions (beginning, middle, end), and numbers written in both Magharibi (North African) and Mashriqi (Middle Eastern) styles, as well as a sentence written freely by each writer that covers short diacritics (tashkeel), and punctuation marks, and it also includes the writer's signature.

The database has a total of 37,000-word images, 10,000-digit images, 2.500 signature images, each writing his signature five times, and 500 images of freely written sentences. A final thinking preprocessing step is applied to the database.

The database is fully available for Canadian researchers and partially available for the rest of the world.

8) The IFHCDB Database:

The IFHCDB (Isolated Farsi Handwritten Character Database) was developed by Mozaffari et al. [29]. This database was collected from Iranian school entrance exam forms. It is a database composed of Farsi-isolated characters and digit numerals. Farsi characters are made of 28 Arabic characters, with an additional four characters. A total of 52,380 character images and 17,740 numeral images made up the database. The frequency of the characters is non-uniform, making some characters have a larger number of image samples compared to other characters, following the natural distribution of the character usage in real life.

Each image of this database has additional information attached to it as metadata, containing information about the writer's name, gender, city name, and birth date.

The database is freely available for academic use.

9) The IESK-ArDB Database:

The IESK-arDB [30] is an offline Arabic handwritten text database comprising 4,000-word images and 6,000 segmented character images. This database covers most of the Arabic parts of speech, country/ city names, security terms, and the words commonly used when writing banking cheques. Twenty-two writers from many Arabic-speaking countries have participated in filling out the forms to create this database. The writers were told to follow the Naskh writing style as much as possible, as it is the most common writing style. This database is very interesting compared to other databases as it contains the annotation of the sub-words and letter segmentation points coordinates. Allowing it to be used for sub-words and letter segmentation purposes. It also contains the label sequence of characters contained in each word image in plain English. The baseline coordinates for each word image are also provided. The annotation is stored in an XML format, holding all information about the word images, such as segmentation point coordinates, and label sequence annotations.

This database is publicly available.

10) The ALTID Database:

The ALTID (Arabic/Latin Text Images Database) [31] is a versatile text database designed to be used for document analysis and text recognition research. This database has multiple scripts, multiple fonts, and multiple text sizes, in which it includes text images in Arabic and Latin Scripts in the printed format and the handwritten one. Making it suitable for various research purposes, such as script identification, font identification, writer identification, and word segmentation.

For the printed text, they used multiple fonts and multiple sizes to create sample images of the Latin/ Arabic text.

For the handwritten text, 17 writers participated in writing on the forms to create the database, of which 3 of them participated in writing both Arabic and Latin scripts. The writers are randomly selected from Tunisia with varying backgrounds.

The database is composed of 731 printed text pages: 387 pages from the APTID/MF database 344 Latin text page images, and 1042 handwritten text-blocks: 460 Arabic and 582 Latin text-blocks.

The database is divided into four groups: the Arabic Printed Pages (APPage), the Latin Printed Pages (LPPage), the Arabic Handwritten Pages (AHPage), and the Latin Handwritten Pages (LHPage) datasets.

The ground truth information of the database is stored in an XML file format, storing text-block-level information and line-level information.

This database is publicly available.

11) The DBAHCL Database:

The DBAHCL (Database for Arabic Handwritten Characters and Ligatures) [32] is a database consisting of two folders: the first folder "DB CHARACTERS" contains Arabic handwritten characters in all the different word positions (beginning, middle, and end of the word), and the second folder "DB LIGATURES" contains ligatures. This database covers all the ligature forms found in handwritten Arabic writing. Fifty writers have participated in writing this database, with 9900 ligatures and 5500 characters.

This database is available for research purposes.

12) The AHTD Database:

The AHTD (Arabic Handwritten Text Database) [33] is a well-designed database with rich content. It is composed of Arabic handwritten paragraphs, with a total of 6,000 paragraphs written by around 300 writers, mostly from Saudi Arabia. The writer's details are included in the database, such as the gender, country of upbringing, left or right-handed, and education level. Each writer has written six paragraphs, in which one paragraph is a summarized paragraph that contains all Arabic characters in all their word positions (beginning, middle, isolated, and end of the word). This paragraph is written twice by the writer to enable research about writer identification and verification. This paragraph is written by all the writers. Each writer writes two distinct paragraphs that are unique for each writer and are randomly selected from an Arabic corpus. Each writer freely writes two paragraphs on any topic they like.

The database is stored in two forms: the first form is as paragraph images with their ground truth data to be used for line segmentation, and writer verification purposes, and the second form is as text line images alongside their ground truth data to be used for word, sub-word segmentation, and text recognition purposes.

This database will be freely made available worldwide for research purposes.

13) The AHTID/MW Database:

The AHTID/MW (Arabic Handwritten Text Images Database written by Multiple Writers) [34] is a database created by gathering 3,710 Arabic handwritten text lines from 53 individuals with varying backgrounds, leading to 3,710 text-line images and later segmented into 22,896-word images. These images come with their ground truth data in XML files, including information on the sequence of words for each text line, the sequence of PAWs (Piece of Arabic Word) for each word, and the sequence of characters for each PAW. This database can be used for word spotting, word recognition, sentence recognition, and writer identification.

This database will be made publicly available for researchers.

14) The AHDB/FTR Database:

The Arabic Handwriting Data Base for Text Recognition (AHDB/FTR) [35] is a database consisting of 497 images of Libyan Towns' names written by 5 different scholars. Each image is associated with ground truth labels consisting of the words present in the image in plain Arabic, the sequence of characters constituting the words present in the image, and other statistical information. The Libyan towns' names consist of utmost three words and many sub-words.

This database can be used for word recognition, word spotting, and writer identification.

The AHDB/FTR database will be freely available for researchers.

15) The Hijja Database

The Hijja database [120] is a database consisting of isolated handwritten Arabic letters in all the writing forms (beginning, middle, ending, isolated). Written by 591 children from Riyadh, Saudi Arabia. These children are of age between 7 and 12. The children wrote a total of 47,434 characters.

This database is a free, publicly available dataset.

2.2.Classical Text Recognition Approaches

2.2.1. Offline Handwriting Text Recognition Process

In this section, we will dive deep into the details of the processes of offline handwritten text recognition for the classical approaches. Often, the processes of offline Arabic handwritten text recognition are similar to the processes of offline Latin handwritten text recognition. The only difference is the direction of recognition. In Latin languages, the direction of recognition is from left to right, and in Arabic, the direction of recognition is in the opposite direction, from right to left. Which almost makes the recognition systems exchangeable between these languages. For this reason, in this thesis, we will talk a lot about the processes of Latin handwritten text recognition alongside those of Arabic.

We can see a pattern in classical recognition systems, they are often divided into the following steps: Preprocessing, Segmentation, Image Representation (feature extraction), and Classification. And in some cases, there is also another step, which is post-processing. The steps are not always ordered in the way we presented. Sometimes the segmentation comes first before the preprocessing. Other times, the feature extraction comes before the segmentation. It depends on the use case. Generally, when the segmentation step comes first this happen when we are trying to apply the recognition on some kind of document, and this is often referred to as documents analysis which is a separate field of research that specialize in the analysis of documents to detect and extract key component in documents such as titles, sub-titles, paragraphs, sentences, lines of texts, images, and so on. In some systems, some steps are either merged or omitted, so we observe a lot of variety in the processes of handwritten text recognition systems.

2.2.2. Preprocessing

To simplify the task of recognition of handwritten texts, the images containing the texts to be recognized pass through several image enhancements and correction techniques Some of these techniques aim to reduce the noise in the image, others aim to correct some errors that happened during the data acquisition step, and some techniques are

designed to normalize the data found in the images to further make the recognition easier.

a) Noise reduction:

Due to the scanning device used to capture the image or the writing instrument, the lighting can be uneven on the image or the writing can be too thick. Filtering is a common technique to reduce the noise in the image by applying some predefined filter convolutions on the image for either smoothing, sharpening, thresholding, removing unnecessary simple backgrounds, adjusting the contrast.

b) Normalization:

Normalization is the process of making images of texts and their content standard by trying to remove the variations in the writing styles with respect to the baseline orientation between different images like the inclinations of the text lines, the tilting of the text, the size of characters.

c) Baseline detection and Skew correction:

As described in previous sections, the Latine and Arabic languages have what we call the baseline, which is a virtual horizontal line that all the writing should happen upon. The baseline skew often happens during the scanning of the image when the document to be scanned is misaligned or can happen because of a writing habit of the writer to skew the baseline of the text. In classical approaches, the detection of the skew and consequently correcting it is necessary for better recognition results. The skewed baseline is also referred to as the slope, and the correction of it is called slope removal. You can check Figure 2.1 to see an illustration of a skewed baseline.

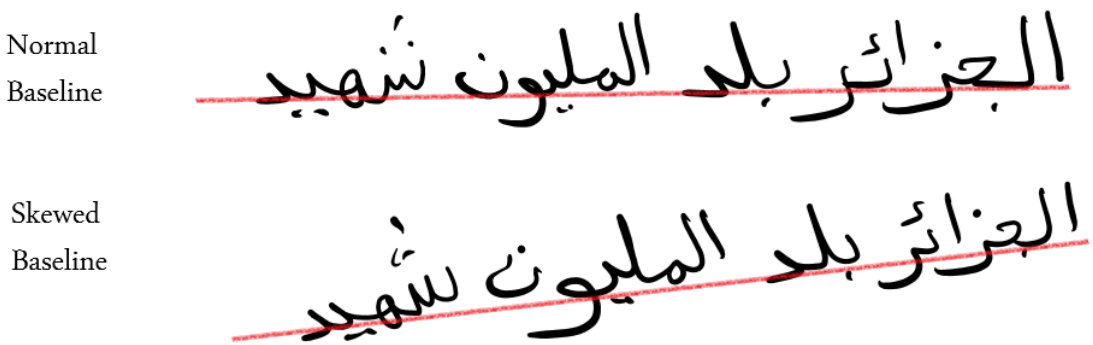
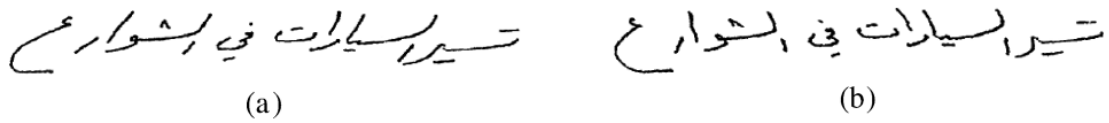


Figure 2.1: An illustration depicting both a standard (horizontal) baseline and a skewed (angled) baseline.

d) Slant angle and correction:

In Arabic and Latin languages, a big part of the writing strokes is vertical and forms a 90-degree angle with respect to the horizontal baseline. Some writers have a habit of slightly tilting those strokes where they will no longer form a 90-degree angle with the baseline, and this significantly diversifies the writings between writers, with each writer tending to have slightly tilted text in any direction. Some techniques are designed specifically to deal with this tilting by calculating the tilting angle, often referred to as the slant angle, and correcting it. Figure 2.2 shows an example of a tilted text.



(a) (b)

Figure 2.2: An illustration showing (a) tilted text on the left and (b) normal text on the right [8].

e) Size normalization:

After the skew baseline and slant angle are corrected, a size normalization step is crucial, where the aim of this step is to make the characters have the same size because much of the techniques that are used later in the process to extract the features are size sensitive, which means that a standard size is mandatory for most recognition systems. Images containing different sizes of the characters will completely ruin the recognition accuracy.

Other normalization techniques are also often applied, such as binarizing the image, where some other techniques in other steps of the process of recognition need the images to be in a binary format, not in a grayscale format. Thinning is also used a lot for the same reason as binarization, and it is also some kind of compression.

2.2.3. Segmentation

Segmentation is the process of segmenting the document into its sub-components: paragraphs, text lines, sentences, or words, and segmenting the words into their sub-components: characters or strokes. The first category of segmentation is called external segmentation, which is typical for document analysis. The second category of segmentation is called internal segmentation, which is the part that concern us the most

in our use case. These two categories are described in more details in this review paper [36].

The segmentation step is very important because without proper segmentation and distinction between the characters in the internal segmentation, the recognition process will fail drastically. In our thesis, we are concerned with the internal segmentation because the external segmentation is part of another research topic.

Internal segmentation techniques are divided into two strategies: 1) explicit segmentation, 2) implicit segmentation. Explicit segmentation aims at cutting the image into multiple sub-images, where each sub-image has a high likelihood of containing a valid character. Common methods used to apply explicit segmentation are vertical projection analysis [37, 38], white space and pitch [39], and connected component analysis [40, 41].

While implicit segmentation combines the character segmentation step and the recognition step into one step, it performs the segmentation while recognizing the letters. Two classes of techniques are employed to perform implicit segmentation: 1) search process-based techniques, 2) image representation-based techniques. Search process-based techniques use a mobile window of variable lengths to provide segmentation suggestions to the recognition system to confirm [36]. Image representation-based technique is further divided into two categories: 1) HMM-based approaches [42], 2) and non-Markovian-based approaches. This class of implicit segmentation uses spatial features extracted from the image to make the segmentation and the recognition. Non-Markovian approaches are inspired by the machine vision concepts of recognizing occluded objects [43].

The segmentation of offline Arabic handwritten words using implicit segmentation techniques such as HMM-based approaches is similar to the Latin language; there is no fundamental difference. On the other hand, segmenting the words into their sub-units explicitly is quite challenging due to the semi-cursive nature of the language. Most segmentation techniques that aim to segment Arabic words into their sub-units tend to segment the word into segments of graphemes instead of letters, according to this review paper [44], where each grapheme is either one letter, part of a letter, or represents more than one letter. This choice of segmenting the words into graphemes instead of

letters (characters) is made because of the difficulty of separating the cursive word directly into characters [45].

Four main categories of explicit segmentation techniques for Arabic can be identified [44]:

a) Segmentation by curve analysis:

All techniques in this category segment the Arabic word into strokes. Almualim and Yamaguchi developed an algorithm that segments Arabic words into strokes [45]. Zahour et al. developed an algorithm that extracts two types of primitives: loops and branches [46].

b) Segmentation by contour analysis:

Miled et al. developed a multi-level system to segment and recognize Arabic words. The first level detects the contours of each sub-word and uses them to extract portions of characters. The second level analyses these extracted graphemes and corrects the failed segmentations [47].

Cheriet et al. extracts visual indices of the word by using contour following to decompose the word into two parts: the main body, and the diacritics. The main body part is further divided into a middle zone, and a prominent zone. The middle zone contains the majority of the information about the word, the prominent zone contains the information about the ascenders, descenders, and tanks [48].

Sari et al. developed a segmentation technique based on morphological rules thar are constructed at the feature extraction stage. This technique gives perfectly separated characters [49].

c) Segmentation by stroke thickness analysis:

Romeo-Pakker et al. developed a technique that uses both horizontal/ vertical projections and contour-following to segment the Latin/ Arabic texts into lines, words, and characters [50].

d) Segmentation techniques based on singularities and regularities:

Motawa et al. used singularities and regularities to detect characters [51].

2.2.4. Image Representation (Feature Extraction)

Image representation in the context of handwriting text recognition is the process of making the images more suitable for the recognition techniques to make sense of them and use them to make a decision. In other words, converting the images into a form and shape that decision algorithms can interpret. Image representation techniques much often called feature extraction techniques, are used to extract relevant features and compact characteristics representation of the image. Feature extractions, though, are a more general term, and image representation is a more specific term used when extracting features from images.

There are three main categories of image representation techniques according to [36]: 1) Global Transformations and Series Expansion representation, 2) Statistical representation, 3) Geometrical and Topological representation.

2.2.4.1. Global Transformations and Series Expansion Representation

The image often contains more information than is needed for classification purposes. One way to create simple, effective, compact representations of the image is by using Global Transformation and Series Expansion. One of the main advantages of using these kinds of transformations is their ability to produce representations invariant to translation, rotation, and scaling. One of the characteristics of Series Expansion features is the ability to reconstruct the original images from these Series Expansion features representation.

a) Fourier Transform:

Fourier Transform has been applied in many research works about text recognition [52, 53]. The way it is used is often by extracting the magnitude spectrum and using it as features for the classification phase. The resulting magnitude spectrum is invariant to translation and rotation.

b) Gabor Transformation:

Gabor filters are used to extract relevant features in [53]. Gabor filters are similar to the early stages of the human visual system, as claimed by many researchers.

c) Wavelets:

The Wavelet Transformation is used as a Series Expansion technique to extract scale-invariant features from images as a series of Wavelet coefficients. They are also called multiresolution features, where each Wavelet coefficient represents a location in the image at a different scale. Lee et al. and Shioyama et al. both used Wavelet transform to extract scale-invariant features for character recognition [54, 55].

d) Moments:

Moments are a Series Expansion feature representation for the image. Different types of moments are used to extract features, normalized central moments are translation and scale invariant, Zernike moments are rotation invariant, and Hu moments are proven to be invariant to translation, rotation, and scale. Multiple works used different combinations of moments to extract features [56, 57].

e) Karhunen–Loeve Expansion:

This transform is common in face recognition applications, and it optimally reduces the dimensionality of images, so it is also an image compression technique.

2.2.4.2. Statistical Feature Extractors

Statistical distributions of pixels in images are used as features. The original image can't be reconstructed from these statistical features. Major techniques used to extract such features are:

a) Zoning:

The image to be recognized is divided into overlapped or non-overlapped zones. Each zone provides regional information about the image. Zoning methods have been found to deal well with different writing styles [58].

b) Crossings and distances:

Characters can be represented by Extracting of statistical features from image contours. These methods consist of analyzing contour crossings, coding pixel transitions, and measuring distances from predefined boundaries. The objective is to characterize the ascending and descending parts of the characters, in order to be able to distinguish and

recognize them. These techniques exploit the geometric properties of the contours to extract relevant information for character recognition [59, 60, 61, 62].

c) Projections:

Projection features are derived from the vertical, horizontal projections of histograms of the black pixels in certain areas of the image [63, 64, 65, 66].

2.2.4.3. Morphological Feature Extractors

Many global and local properties that define a character can be used as structural features. These structural features use the geometrical and topological properties of characters as their foundation. These topological and geometrical features describe the characters in many different ways depending on the language, and they are made of many different primitive structures, and they can be as simple as strokes, lines, and arcs in Latine languages, and as complex as zigzags, dots, loops, end points, intersection points, strokes in many directions, curves, splines, number of dots and their position with respect to the baseline in the Arabic language. These predefined structures are searched in the image, and their count and relative positions make up the extracted features [67, 68, 69, 70, 71]. Another popular method to extract structural features is the use of coding schema such as Freeman's chain code [72, 73, 74, 75, 76]. The coding is obtained by mapping the character strokes into a 2-D parameter space as shown in Figure 2.3. An example of coding a character using Freeman's chain code is illustrated in Figure 2.4.

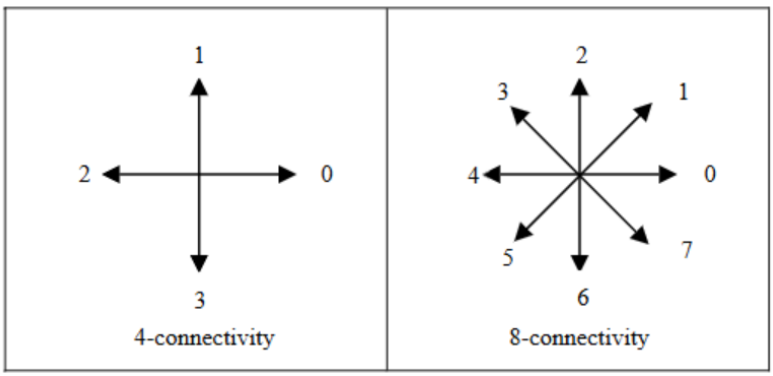


Figure 2.3: An illustration of the 2D parameter space with 4-connectivity and 8-connectivity in the Freeman chain code. [73]

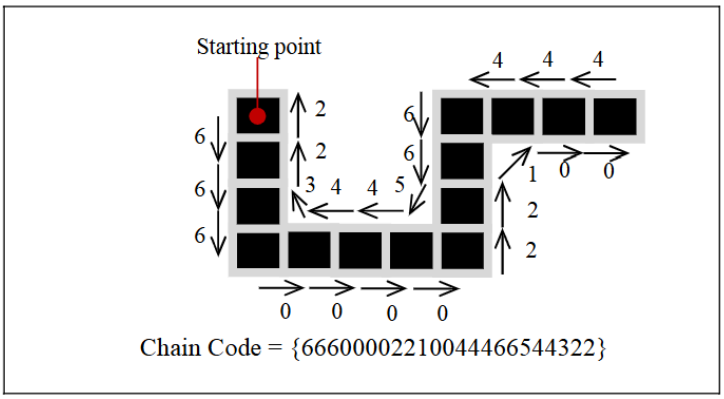


Figure 2.4: An illustration of a character represented using Freeman chain codes [73].

2.2.5. Training and Recognition

Text recognition systems extensively adopt many pattern recognition methods. These methods assign classes to unknown samples. In this case, they assign a character or a word to an image or a segment of an image. The recognition is generally made by a classification algorithm. These algorithms are usually trained before the recognition phase. Their training data consists of databases of labelled text images. These classification algorithms use the features extracted from the databases of text images in the training phase to group the characters that belong to the same class together intra-classes, and make the differences between the inter-classes much bigger to differentiate between the characters. The classification algorithms determine whether the word is present in the image or the characters, depending on the recognition system. There are multiple ways of using the extracted features for recognition. It can be as simple as using template matching methods to recognize prototypes stored of handprinted characters or words [36, 77, 78], or as complex as using Markovian models to recognize sequences of characters [10]. Several systems for recognizing sequences of handwritten Arabic texts with hidden Markov models approaches have been proposed [79, 80, 81, 82, 83].

2.3.Neural Network Approaches to Text Recognition

2.3.1. Introduction

In the previous section, we reviewed classical approaches to handwritten text recognition (HTR), with a particular focus on methods applied to Arabic script. These traditional approaches, although seminal, often present limitations in terms of flexibility, robustness, and performance, particularly in the face of the complexity of

Arabic script, characterized by its cursive nature, contextual variations, and ligatures. In this section, we turn to modern approaches based on neural networks, which have revolutionized the field of HTR thanks to their ability to learn hierarchical representations and model complex relationships in data.

Neural networks, particularly deep architectures, have enabled significant advances in handwritten text recognition, including for Arabic. These methods leverage machine learning models that can process raw data (such as text images) and extract relevant features from it without requiring manual preprocessing steps or explicit character segmentation. This capability is particularly crucial for Arabic, where character segmentation is often challenging due to the connectivity of letters and shape variations across their positions in the word.

In this section, we focus specifically on research conducted in the field of Arabic handwritten text recognition (HTR) using neural networks. We will classify these approaches into three main categories, based on the granularity of the text being processed and the specific challenges associated with them:

1. *Isolated character recognition:* These approaches focus on recognizing individual characters, often extracted from words or lines of text. They are particularly useful for tasks where characters can be easily segmented, but they must cope with the shape and style variations inherent in Arabic handwriting.
2. *Single Word Recognition:* These methods treat the word as a whole entity, thus avoiding the difficulties associated with character segmentation. They are particularly suited to the recognition of handwritten or printed words in Arabic, where the connectivity of letters makes segmentation complex.
3. *Text Sequence Recognition:* These approaches aim to recognize lines of text or sequences of characters of arbitrary length. They are the most complex due to the need to model long-term contextual dependencies and to handle significant variations in handwriting.

Each of these categories presents specific challenges and innovative solutions, often based on deep neural network architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memories (LSTMs), and

attention mechanisms. We will explore these approaches in detail, highlighting the key contributions, the architectures used, and the performances obtained on Arabic datasets.

In addition, we will discuss emerging trends in this field, such as transfer learning, the use of advanced attention mechanisms, and the integration of hybrid models combining multiple architectures. These innovations pave the way for more robust, accurate, and adaptive Arabic handwritten text recognition systems that can accommodate a variety of writing styles and application contexts.

2.3.2. Isolated Character Recognition Approaches

The recognition of isolated Arabic characters is one of the first approaches explored in the field of optical character recognition (OCR). This method has been the subject of extensive research, mainly due to the availability of Arabic databases containing isolated characters in various forms and positions. Indeed, many researchers have contributed to the enrichment of these resources by designing their databases dedicated to isolated Arabic characters. This trend is explained by the relative simplicity of creating and exploiting such databases, compared to more complex approaches involving words or entire sentences. Thus, isolated character recognition represents the most direct and widespread method for addressing the problems related to the recognition of Arabic script. Consequently, research works based on this approach are particularly numerous.

Among the most commonly used techniques, convolutional neural networks (CNNs) stand out as automatic feature extractors, often combined with fully connected layers for prediction. Researchers have explored various optimization avenues, including testing different CNN architectures, comparing shallow and deep models, proposing hybrid models, exploiting transfer learning, fine-tuning hyperparameters, and using data augmentation techniques. To illustrate this approach, we will focus on three significant studies, representative of the work conducted in this field.

For example, Jbrail et al. [84] studied the recognition of Arabic handwritten characters using convolutional neural networks. Their research is based on the Hijja dataset, comprising 47,434 handwritten characters produced by children aged 7 to 12 years. The authors proposed three distinct CNN models: Model A, composed of 3 hidden layers, achieves an accuracy of 81%; Model B, also with 3 hidden layers but using a combination of ReLU and Softmax activation functions, achieves an accuracy of 85%;

finally, Model C, more complex with 9 hidden layers, achieves a remarkable accuracy of 99.3%. The latter model integrates optimization techniques such as Adam, as well as an architecture combining Conv2D, MaxPool, Dropout, Flatten, and Dense layers, allowing to obtain competitive performance compared to previous studies.

In another study, Younis et al. [85] proposed a deep convolutional neural network approach for Arabic handwritten character recognition. Their model integrates regularization techniques, such as batch normalization and dropout, to improve performance and avoid overfitting. The CNN architecture used consists of three convolutional layers followed by a fully connected layer, with variable-size filters and ReLU activation functions. The experiments were conducted on two main datasets: AIA9k and AHCD. Through hyperparameter optimization, including increasing the number of filters and applying data augmentation techniques, their model achieved accuracies of 94.8% on AIA9k and 97.6% on AHCD. The authors also analyzed the classification errors, finding that confusions mainly occur between characters with similar shapes. They suggest using a two-stage classification scheme to further improve the results.

Finally, H. M. Balaha et al. [86] developed a method for recognizing Arabic handwritten characters based on convolutional neural networks. Their model was evaluated on the Hijja dataset. The proposed CNN architecture consists of multiple convolutional layers followed by fully connected layers, with ReLU and Softmax activation functions. The authors also employed regularization techniques, such as Dropout, to reduce overfitting. By optimizing the hyperparameters and increasing the network size, their model achieved an accuracy of 97% on the Hijja dataset.

For a comprehensive overview of the research in the field of isolated character recognition using deep learning, we refer the reader to the extensive review article by El Khayati et al. [87].

2.3.3. Isolated Word Recognition Approaches

The second category of approaches in Arabic script recognition concerns deep learning methods designed to recognize isolated words. Unlike single-character recognition approaches, these methods treat the word as a global entity, without explicit segmentation of its component characters. These approaches, often referred to as holistic, are distinguished by their ability to capture the global characteristics of the

word, which makes them particularly suitable for recognizing handwritten or printed Arabic texts, where character segmentation can be complex due to the cursive nature of the script.

Although it may seem intuitive to assume that these approaches rely mainly on convolutional neural networks (CNNs) similar to those used for isolated character recognition, the work conducted in this area is more diverse and innovative, although less numerous. For example, Lamtougui et al. [88] explored transfer learning using four CNN architectures pre-trained on the ImageNet database (VGG16, ResNet50, AlexNet, and InceptionV3), which they then fine-tuned on the IFN/ENIT Arabic handwritten database. These models were applied to word recognition, demonstrating the effectiveness of transfer learning in this context.

In another study, AL-Shatwani et al. [89] proposed a hybrid approach combining a classical feature extraction method, the stationary wavelet transform (SWT), with an artificial neural network (ANN) classifier. Their model, trained using Bayesian regularization (BR) and Levenberg-Marquardt (LM) methods, was applied to word recognition on the IFN/ENIT dataset, providing an interesting alternative to purely deep learning-based approaches.

Furthermore, Haboubi et al. [90] and Hassen et al. [93] developed similar architectures for Arabic text recognition, although their work focused on distinct datasets. Haboubi et al. [90] designed an end-to-end system to recognize Arabic city names from the Tunisian IFN/ENIT dataset. Their architecture integrates a CNN block for feature extraction, followed by a bidirectional recurrent gate unit (BGRU), and ends with dense layers with a softmax output to predict a word. Similarly, Hassen et al. [93] proposed a system for recognizing Arabic words from historical documents, using a similar structure composed of a CNN block, a BGRU block, and a softmax output layer. Their system was applied to two historical datasets: the IBN SINA dataset [94] and the VML-HD dataset [95]. These works illustrate the effectiveness of architectures combining CNN and BGRU to process Arabic texts, whether modern or historical.

A particularly interesting contribution in this area is that of Awni et al. [91], who conducted extensive transfer learning experiments using a ResNet-18 CNN architecture pre-trained on the ImageNet dataset. They proposed a two-phase strategy: in the first phase, the ResNet-18 model was retrained on the Arabic AlexU-W dataset [92]; in the

second phase, the fine-tuned model was reused and retrained on the IFN/ENIT dataset. This approach allowed exploring three types of transfer learning: fine-tuning, progressive layer freezing, and keeping all layers fixed, thus providing a comparative performance analysis.

Another innovative approach was proposed by Sulaiman et al. [21], who developed a two-stream deep neural network model for language-independent handwritten word recognition. This architecture combines two complementary information levels: a first stream processes weakly separated segments of the image by exploiting convolutional LSTM layers to capture sequential dependencies and local features; a second stream, inspired by the U-Net architecture, uses convolutional and max-pooling layers to extract global shape and structure features of the word. The features of the two streams are then merged and processed by fully connected layers to predict words. This approach, evaluated on the IFN/ENIT and AlexU-W datasets, avoids the challenges of explicit character segmentation, making it particularly suitable for cursive scripts such as Arabic, while remaining applicable to other languages.

Finally, Al-Taee et al. [96] proposed a novel approach based on the Faster R-CNN architecture for Arabic handwritten word detection and recognition. By integrating pre-trained convolutional neural networks (VGG16 and ResNet50), they efficiently exploited the IFN/ENIT and KHATT datasets to train and test their model, demonstrating the potential of object detection architectures tailored for word recognition.

2.3.4. Sequence-based Text Recognition Approaches

Finally, the third category of approaches in Arabic handwriting recognition concerns deep learning methods designed to recognize character sequences of arbitrary length. This approach is considered the most complex due to the challenges inherent in the cursive and contextual nature of Arabic handwriting, as mentioned in the previous sections. These methods aim to recognize text lines or words of varying lengths without requiring explicit character segmentation, which distinguishes them from previous approaches.

Among the dominant techniques in this field, Connectionist Temporal Classification (CTC) [97] occupies a central place. It is used in the majority of research on sequence-based Arabic handwriting recognition (HTR). CTC allows modeling character

sequences of varying lengths by automatically aligning the model outputs with the target sequences, without requiring prior segmentation. However, some notable exceptions exist, such as the CALText system proposed by Anjum and Khan [98], which is based on an encoder-decoder architecture with an integrated attention mechanism. In this system, the encoder is based on a DenseNet, a variant of convolutional neural networks (CNNs), while the decoder uses gated recurrent units (GRUs) with a contextual attention mechanism. This mechanism allows focusing attention on local regions of the input image, thereby improving recognition accuracy. Their model has been successfully tested on handwritten Arabic and Urdu texts.

However, most research in this area relies on architectures combining convolutional neural networks (CNNs) with bidirectional LSTMs (BLSTMs) and a CTC output layer. This common architecture, often referred to as CNN-BLSTM-CTC, has been widely adopted to model the sequential nature of languages. For example, several studies [99, 100, 101, 102] have used this approach for Arabic handwritten text recognition. Jemni et al. [100] experimented with author-dependent and author-independent training strategies to improve the performance of their system. In particular, they implemented a two-stage knowledge transfer strategy: first, the model is trained on the source dataset (KHATT), then the weights of the intermediate layers are released for retraining on the target dataset. In a second step, all the model weights are released for a final retraining on the source dataset.

In the same context, Noubigh et al. [102] explored a transfer learning strategy by initially training their CNN-BLSTM-CTC model on a printed Arabic dataset (P-KHATT) and then retraining it on the handwritten KHATT dataset. This approach has allowed to evaluate the efficiency of knowledge transfer between similar but distinct domains. Furthermore, Elsayed et al. [103] introduced a variation of this architecture by replacing the VGG16 network with EfficientNetB3 as the feature extraction block, combined with a bidirectional LSTM and a two-headed attention mechanism. Although the details of the attention mechanism are not clearly explained, this approach has shown promising results.

A major contribution in this field is that of Alex Graves, the author of the CTC technique [97], who developed an innovative architecture called the multidimensional recurrent neural network (MDRNN) [104]. This model uses a hierarchical structure based on

multidimensional long-short-term memories (MDLSTM) to capture context in all directions, not just sequentially. The MDLSTM extracts features progressively, starting with simple local patterns and ending up with complex global structures, before predicting the text sequence via a CTC output layer. This architecture won an award in the offline Arabic handwriting recognition competition at the International Conference on Document Analysis and Recognition (ICDAR) in 2009 [105], using the IFN/ENIT database as a reference.

Finally, Maalej and Kherallah [106] experimented with variants of the MDLSTM architecture by integrating modern techniques such as Dropout, ReLU and Maxout activation functions. These modifications improved the robustness and performance of the model, demonstrating the flexibility and potential of MDLSTM-based architectures for Arabic handwritten text recognition.

3. Chapter 03: Offline Arabic Handwritten Text Recognition for Unsegmented Words using Convolutional Recurrent Neural Network

3.1. Introduction

This chapter presents a novel approach to offline Arabic handwriting text recognition using a Convolutional Recurrent Neural Network (CRNN). The proposed system is designed as a three-part deep learning model: feature extraction using convolutional layers, sequence modeling using recurrent layers, and transcription via Connectionist Temporal Classification (CTC). The model is trained and evaluated on the IFN/ENIT dataset, demonstrating promising performance compared to state-of-the-art techniques.

3.2. Methodology

The deep learning architecture that we have designed is a CRNN-based architecture, which consists of a convolutional block, a recurrent layer block, and a final transcription layer made up of a CTC layer, as illustrated in Figure 3.1.

We can see in Figure 3.1 that the network starts with an input layer that accepts a text image as input, and this image will be first processed by multiple convolutional layers stacked on top of each other in a specific way to automatically extract visual features. The features extracted from this step have a sequential nature due to the specificity of the design of the architecture, which preserves the spatial information of the original image. These features will be the input of the next module in the CRNN network, the recurrent layers, which by design they natively process sequential data. They process the sequence of features of the previous step, and they produce a first preliminary prediction of a sequence of characters. The prediction of these characters is jointly done with a final CTC layer that takes the responsibility of both training the network with a CTC loss function and producing the final character prediction estimates. All the previously described processes of training are automatically done end-to-end, as this is one main advantage of using such a network design, which gets rid of performing multiple training steps for separate systems, as is done in other systems.

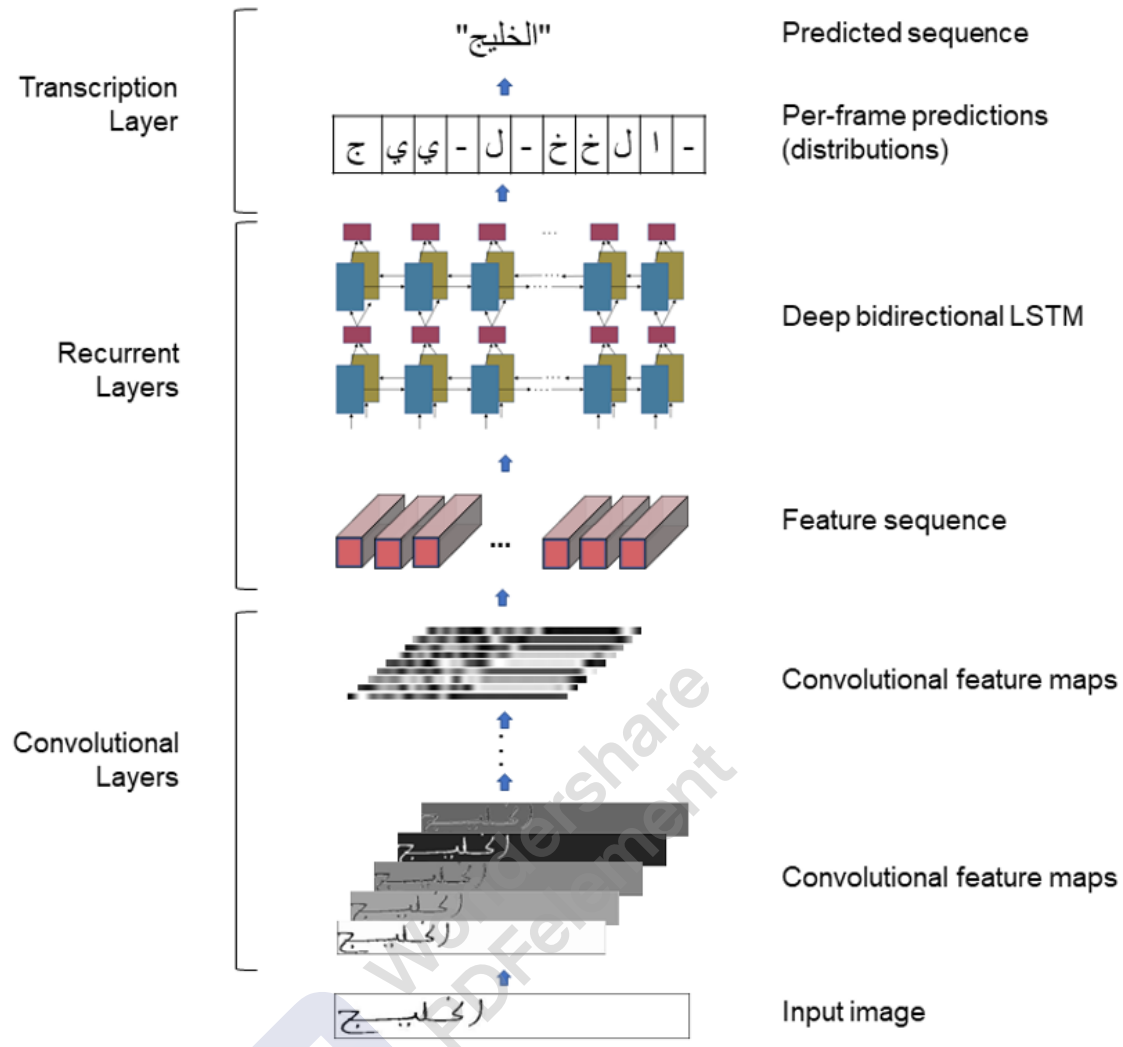


Figure 3.1: The model architecture composed of three parts: 1) a CNN block; 2) a recurrent block; 3) a CTC transcription layer.

3.2.1. Feature Extraction

The first block of the CRNN architecture is a Convolutional Neural Network CNN block. The network we designed is expecting an input image of a fixed size. Therefore, all the images that are used to train the network or the ones that are used during inference need to be scaled to the appropriate input size. Traditionally speaking, CNNs were first designed to extract a holistic representation of the given image. They were best suited for simple classification tasks, such as assigning a class to the image of interest. But such design doesn't suit our use case, where we want to get a variable sequence of classification for each region of the image, where each region of the image is supposed to hold a character, as we know that a word or a text line is composed of a sequence of characters neighboring each other. For this reason, instead of using a traditional CNN architecture that produce or generate a wholistic representation of the

image as a whole as a feature maps, we designed a different CNN architecture that produce a fixed sequence of vectors of feature maps, where each vector of feature maps can be mapped to a specific region in the image of input. Another interesting characteristic of this design is that neighboring vectors of feature maps directly represent neighboring regions in the input image. These regions are often called the receptive fields, as illustrated in Figure 3.2. We must note that because of the way CNN is implemented, the generation of these vectors of feature maps is done from left to right, and the Arabic language is written from right to left. This way of generating feature maps will not affect the generated feature maps, as it would make no difference if they were implemented to generate feature maps from right to left; they will always produce the same feature maps.

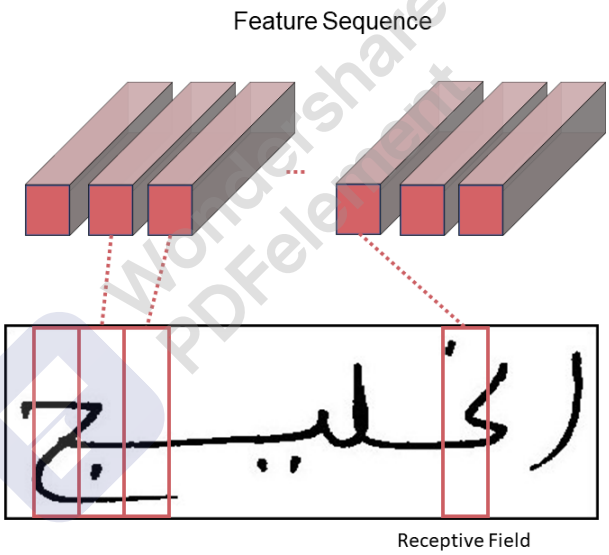


Figure 3.2: Sequence of Feature vectors representing specific regions in the original image.

3.2.2. Label Prediction

The second block of the CRNN architecture is a Recurrent layer block, specifically a Deep Bidirectional Recurrent Neural Network (DBRNN) block. This Recurrent Neural Network (RNN) block is stacked on top of the CNN block. It receives the outputted sequence of vectors of feature maps produced by the CNN block, and processes it to produce a new sequence of vectors of feature maps. The reason why we perform this extra step of processing the feature maps generated by the CNN block by the RNN block is to add the context information to the feature maps sequence of vectors. As each

vector of feature maps produced by the CNN block only represents a specific region from the input image, it is not aware of the neighboring regions. The problem that arises with designing a neural network without an RNN block is that the network will not have the ability to recognize characters that cover multiple neighboring receptive fields in the input image, as a character often takes up many neighboring spaces in the input image. Thus, incorporating a block such as an RNN block to capture contextual information is beneficial. Figure 3.2 illustrates a character that takes up many neighboring receptive fields in the image of the input.

We used Long Short-Term Memory (LSTM) cells to capture longer contextual information rather than using normal vanilla recurrent cells. We used bidirectional LSTM layers to capture context coming from the left as well as to capture context coming from the right to maximize the context captured from the data. We stacked two bidirectional LSTM layers on top of each other, which makes the RNN network deep and can extract complex contextual patterns. A deep bidirectional LSTM network is shown in Figure 3.3. The RNN block takes a sequence of feature vectors $X = x_1, \dots, x_t$ from the previous CNN block, process it and produce a new sequence of contextual feature vectors $Y = y, \dots, y_t$. The RNN block is then followed by a fully connected layer to allow for the prediction of label distributions. Labels, in this case, are the Arabic characters. Each vector y_i will be used to predict a character in the Arabic language. This fully connected layer is necessary during training and inference as it is directly used with the next CTC layer. Finally, the RNN block is trained with the algorithm of backpropagation through time, which is the way to go when training RNN networks.

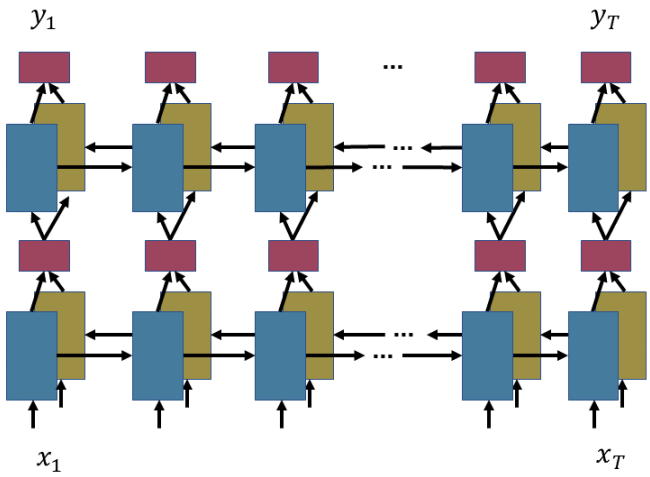


Figure 3.3: a deep bidirectional LSTM network.

3.2.3. Transcription

The predictions made by the recurrent layers are not yet ready to be interpreted. They don't usually form an authentic label prediction. Thus, they must be converted into an authentic label prediction through an operation called transcription. Transcription, in this case, is the process of converting the sequence of characters predicted by the recurrent layers into a true label prediction using the CTC layer. Alex Graves et al. developed the CTC method [97] for the recurrent layers to recognize speech. This method is used in both the training and the inference of the model. It excels at training the models with a sequential nature, such as speech recognition and text recognition. As they both prohibit a sequential nature, where the aim is to recognize a sequence of phonemes or a sequence of characters. The transcription can be either made with the help of a lexicon or without it. When the transcription is made with the help of a lexicon (lexicon-based), we take the transcription with the highest probable label sequence over a set of label sequences. This way of transcribing is computationally expensive as it requires a lot of computation, which is a scarce resource. The other way of doing the transcription is without the help of the lexicon (lexicon-free). We used the Lexicon-free transcription, although it yields worse transcription results compared to the lexicon-based transcription, but we compensate for this loss of precision with the use of a BK-tree data structure. We efficiently search over a set of lexicons after the transcription phase, and we take the closest label sequence as a prediction. When we use the BK-tree

data structure, we are typically narrowing the predictions to a set of predetermined labels. Therefore, the prediction is constrained to a set of predetermined labels, but when we perform the predictions without the use of a BK-tree data structure, the prediction is free and unconstrained, and we can predict any label freely without relying on any set of predefined labels. The choice of either incorporating a BK-tree data structure or not is dependent on the use case.

3.3.Experimental Setup

Our CRNN model was trained on a standard Arabic handwritten text database, "IFN/ENIT database". A lot of experiments have been done on this database. Thus, making it an ideal database to test our results and compare them with other research. In this section, we will present the IFN/ ENIT database, we will explain the training procedure we followed to train the model, and will present the results and discuss them.

3.3.1. Dataset

The database we chose to train and test our model on is the IFN/ENIT database developed by Mario Pechwitz et al. [22]. The developers of the database indicate that 411 writers from Tunisia have participated in the writing of the database. The database is composed of images that contain handwritten names of towns and villages of Tunisian cities. The texts are written in Arabic. The developers of the database have developed several versions of it, with the first version consisting of four sets (a, b, c, and d) containing around 26000 words with an overall 210000 characters. Later, they enriched the database with an additional three sets (e, f, and s). Set e and f are also written by Tunisian writers, while Set s is written by writers from the UAE to add more diversity and challenge to the database. It is common for researchers to use the first four sets for training and the last three sets for validation and testing.

3.3.2. Training Procedure

Table 3.1 describes the network architecture of our CRNN model. The architecture is read in the table from bottom to top, where the bottom layers represent the start of the network, and the top layers represent the final layers of the network. All the images used in training and testing need to be scaled to the fixed size of (64, 512), which is the input size we chose for the network to keep as much information from the images as we can. Normalization is applied to these images after scaling them. The first layer is a

convolutional layer followed by a Rectified Linear Unit (relu) activation function. Another convolutional layer is added, followed by another relu activation function. These first two convolutional layers are not followed by a BatchNormalization layer. But all later convolutional layers are immediately followed by the BatchNormalization layer before adding the relu activation function to speed up training. The network we designed is inspired by the VGG convolutional neural network. We used a convolutional kernel of 3 × 3. We started by feature maps of 64 for the first initial convolutional layer, and we gradually increased the size of feature maps to finally reach the size of 512 feature maps. The size of the feature maps was doubled gradually as follows: (64, 128, 256, 512). MaxPooling layers of 2 × 2 are also applied after the convolutional layers to reduce the dimensions of the feature maps. This MaxPooling configuration of 2 × 2 is used for all MaxPooling layers except for the last MaxPooling layer, where we used a MaxPooling of 2 × 1 and a stride of 2 × 1. This adjustment for the last MaxPooling layer is necessary to allow the network to be able to predict a longer sequence of characters at the final layer. This design of the CNN block makes its output a sequence of 31 vectors of feature maps. Thus, allowing for the prediction of Arabic words or text lines of a maximum length of 31 characters. It's enough to predict all words of the Arabic language because no Arabic word is composed of more than 31 characters. To forward the sequence of feature maps produced by the CNN block to the recurrent layers, a special layer (Map-To-Sequence layer) is used to convert the output of the CNN to the appropriate input shape needed by the recurrent layers block. Two consecutive layers of bidirectional LSTM were used to create the recurrent layers block. Dropout of 0.2 was applied to these two bi-LSTM layers. The recurrent layers were finally followed by a fully connected layer (dense layer) with a "softmax" activation function to predict the sequence of characters. L2 regularization was also applied to help reduce overfitting.

We implemented the model with Python 3 and the TensorFlow v2 framework. We created custom implementations of the "Map-To-Sequence" layer, the transcription layer, and the BK-Tree data structure. The model was trained using the 16GB of RAM Tesla P100 GPU and the 16GB of RAM Tesla T4 GPU. The early stopping was activated during training to assess the convergence of the model. The training of the model converged and automatically stopped training at epoch 53. The model was trained with a pair of images and ground truth labels of a batch size of 256. The

optimization algorithm of Stochastic Gradient Descent SGD was used to train the model with a momentum of 0.9 and a learning rate of 0.01. Approximately two hours were needed for the training to be completed, with an average of 126 seconds for each epoch.

Table 3.1: Model architecture. The first row is the final output layer. 'k', 's', 'p' and 'd' stand for kernel size, strides, padding size and dropout respectively. "chars" is the number of characters in the language of the dataset.

Type	Configurations
Transcription	-
Fully connected layers	#units: chars + 1, activation: softmax, regularization: L2
Bi-LSTM	#hidden units: 128, d: 0.2
Bi-LSTM	#hidden units: 128, d: 0.2
Map-To-Sequence	-
Activation	relu
BatchNormalization	-
Convolution	#maps: 512, K: 2 × 2, S: 1, P: Valid
MaxPooling	Window: 2 × 1, S: 2 × 1
Activation	relu
BatchNormalization	-
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	relu
BatchNormalization	-
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
Activation	relu
BatchNormalization	-
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	relu
BatchNormalization	-
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
Activation	relu
BatchNormalization	-
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	relu
BatchNormalization	-
Convolution	#maps: 128, K: 3 × 3, S: 1, P: Same
Activation	relu
BatchNormalization	-
Convolution	#maps: 128, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	relu
Convolution	#maps: 64, K: 3 × 3, S: 1, P: Same
Activation	relu
Convolution	#maps: 64, K: 3 × 3, S: 1, P: Same
Input	64 × 512 the size of a black and white image

3.3.3. Results and Discussion

Our CRNN model was trained on sets A, B, C, and D of the IFN/ ENIT dataset. It was validated on set E and tested on set F. To assess the performance of our model, we used the Word Accuracy Rate (WAR) as a metric. Our model predicts a sequence of characters, and with the help of the BK-tree data structure, we convert the predicted sequence to a true label prediction from the predefined set of labels (city names) in this case, and we use the WAR to calculate the accuracy and we show the results in Table 3.2. Figure 3.4 shows a graph plot of the training and validation losses as they change through the training epoch. From the graph, the training and validation losses start converging at epoch number 40.

Table 3.2: Word recognition accuracy of training, validation and testing the CRNN model on the IFN/ENIT database.

Database version: IFN/ENIT dataset v2.0p1e		
	Sets	Word Accuracy %
Training	A, B, C, and D	99
Validation	E	80
Testing	F	79

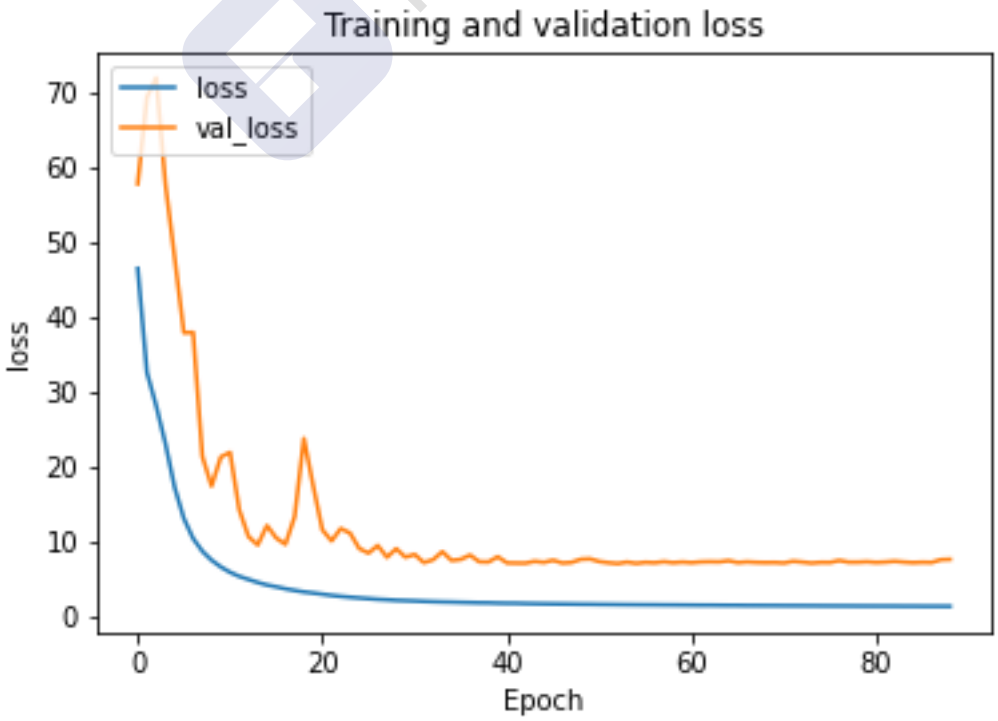


Figure 3.4: The training loss and the validation loss plotted against the number of epochs.

In Table 3.3 we compared the results we obtained with other systems. Many researchers have proposed many approaches to recognize Arabic handwritten text, and have conducted many experiments to assess the performance of their systems. The experiments conducted on the IFN/ ENIT dataset are done in many ways. Not all teams of researchers have agreed on a standard way of researching the Tunisian database. Some researchers [79, 107] trained and tested their systems on the a, b, c, and d sets with the cross-validation technique. One team of researchers [108] took a different training approach where they trained on sets a, b, and c, and tested on both d and e sets, and then they averaged the accuracy of both the test sets (d and e). But, most researchers [109, 110, 111, 112, 113, 114, 115] trained their systems using a straightforward strategy of training where they trained on sets a, b, c, and d, and tested on set e.

Our system exhibited promising results and indicates that deep learning has huge potential to develop the Arabic handwritten recognition systems. Although little research has been conducted to develop Arabic HTR systems with deep learning, it is already in many cases comparable to classical research that has been around for decades by now.

Table 3.3: Word recognition accuracy of several systems that have trained on Sets: a, b, c, d and tested on set E.

System	Training sets	Testing sets	Word Accuracy %
CNN-HMM Mustapha Amrouch et al. 2018 [109]	a, b, c, d	e	89.23
Mohammad Tanvir Parvez et al. 2013 [111]	a, b, c, d	e	79.58
El Moubtahij Hicham et al. 2017 [110]	a, b, c, d	e	78.95
Jawad H. Alkhateeb et al. 2011 [112]	a, b, c, d	e	DBN (66.56) HMM (82.3)
Hamdani et al. 2009 [113]	a, b, c, d	e	81.93
Kessentini et al. 2012 [114]	a, b, c, d	e	79.6
DBN (Jayech et al. 2016) [115]	a, b, c, d	e	78.5
CRNN our system	a, b, c, d	e	80

3.4.Conclusion

In this research, a new variation of the Convolutional Recurrent Neural Network (CRNN) to recognize offline Arabic handwritten texts is defined, and it is specially designed to suit large input size images. Experiments were conducted on a standard database, the IFN/ ENIT database. The system exhibited promising results and showed the significance of using deep learning systems to create Arabic handwritten text recognition systems. The results were compared with other systems to assess the performance of the system. One of the benefits of these kinds of deep learning systems is the ability to design fully end-to-end systems without the need to create separate

systems that work with each other, further simplifying the process of designing Arabic HTR systems. One more advantage of using such deep learning systems is the ability of the automatic feature extraction from visual image data with the convolutional neural networks (CNNs) instead of manually extracting handcrafted features with classical approaches. For future research, deep learning based HTR systems will allow a new kind of research to be done, such as investigating transfer learning strategies and domain adaptation to benefit from knowledge gained from other similar fields. Also, the use of pretrained CNN blocks such as Inception, ResNet, and MobileNet networks will allow the extraction of more robust visual features from the data. Incorporating attention mechanisms into the recurrent block will also be a topic of interest in future research to capture better contextual information from the data.

4. Chapter 04: Data Augmentation for Offline Arabic Handwritten Text Recognition Using Moving Least Squares

4.1. Introduction

In this chapter, we present a new data augmentation technique to augment Arabic text images. This augmentation technique is based on Moving Least Squares (MLS) [117], and it is used to deform the images of interest in a way that makes the text present in the newly augmented images look different, like it was written differently, or written by another person. This new technique was tested on the CRNN model. It was trained with the IFN/ ENIT database and later compared with other traditional augmentation techniques. This augmentation technique, when used to enrich the training data of the CRNN model, led to better performance of the model and outperformed the models trained without augmentation or with images augmented with traditional techniques.

4.2. Methodology

In the field of Arabic HTR, researchers have investigated many different approaches to improving the performance of their Arabic HTR systems. Among these approaches, data augmentation is a potential candidate to improve performance when data is scarce, and it is indeed scarce in the field of Arabic HTR because of the difficulty of gathering such data.

In this research, we have designed a new data augmentation technique based on the MLS technique most known for deforming images. This augmentation technique will be used to create new augmented images, and these newly created images will be used to train the deep learning models. To test the performance of this new augmentation technique, we first designed a new variation of a CRNN model, an HTR deep learning model known in the literature. We used this newly designed CRNN model as a baseline for performing the experiments. We compared the performance of this newly designed CRNN model with the CRNN model previously developed for Arabic HTR in the previous chapter. This newly designed CRNN model has distinct characteristics, where it is characterized by its fewer parameters, and its capability to take small-sized images as input, compared to the model mentioned in the previous chapter, which is

characterized by its capability to take large-sized images as input. From now on, in this study, we will refer to the newly designed model as the small input-size model, and the model mentioned in the previous chapter will be referred to as the large input-size model. An illustration of the CRNN model we used in this study is shown in Figure 3.1.

In this study, the experiments were designed around assessing the performance of incorporating the images augmented with the MLS technique into the training data of the baseline model. For that, the baseline model (small input-size model), was trained three times: the first time, without incorporating any augmented images into the training data, the second time, with incorporating images augmented with traditional augmentation techniques into the training data, and the third time, with incorporating images augmented with the MLS-based augmentation technique into the training data.

In the following subsections, the details of the baseline model (small input-size model), the details of the traditional augmentation techniques, and the details of the MLS-based augmentation technique are discussed.

4.2.1. Baseline Model (Small Input-Size Model)

The network architecture of the model that will be used as a baseline, which was designed in this chapter, differs from the model designed in the previous chapter in the size of the input image, the number of layers, and the kernel and stride sizes of some of the layers. The network architecture of the small input-size model is described in detail in Table 4.1. The table presents the network layers, starting from the bottom row as the input layer to the top row as the output layer and the final layer of the network. The input images first need to be normalized and scaled to the standard image size of (32, 128). The first two layers are convolutional layers, followed by a ReLU activation function. BatchNormalization is used to speed up the training, and it is applied to all convolutional layers, except for the first layers. This model like the model designed in the previous chapter is inspired by the VGG CNN architecture, where the 3 x 3 size is used for the kernels of the convolutional layers, and the number of feature maps starts from 64 feature map for the initial layers, and gradually increase to reach 512 feature maps for the last convolutional layers. The last convolutional layer of this model uses a 2 × 2 kernel size. Like regular VGG CNN architectures, the convolutional layers are followed by MaxPooling layers to reduce the size of the feature maps. 2 × 2 MaxPooling layers are applied after the convolutional layers. The last two MaxPooling layers use a

2 × 1 MaxPooling window and a 2 × 1 stride. These adjustments are necessary to allow the CNN block to produce wide feature maps that will eventually help predict all available Arabic words. The network architecture of the CNN block we just described allows for outputting a sequence of 31 feature vectors. The architecture was designed in a way that makes an input image of size (32, 128), and produces a sequence of 31 feature vectors, to be later used to predict a sequence of 31 Arabic characters. A specially designed layer named "MapToSequence" is added after the CNN block to convert the output of the CNN block to the appropriate input shape of the next recurrent block. The sequence of feature vectors $X = [x_1, x_2, \dots, x_t]$ produced by the CNN block is fed to the recurrent layer block, where the recurrent block will produce another sequence of contextual feature vectors $Y = [y_1, y_2, \dots, y_t]$. The recurrent layers block is later connected to a fully connected layer to predict a label distribution (a character) for each y_t. CTC loss function uses this sequence of label distributions to calculate the loss. The recurrent layers block used a deep Bidirectional Long Short-Term Memory (Bi-LSTM) to capture previous and later context.

Table 4.1: Network configuration summary of Small Input-Size Model. The uppermost row denotes the top layer. Within this context, 's', 'k', 'd', and 'p' correspond to strides, Kernel size, dropout, and padding size, in that order. The variable "chars" represents the count of characters within the language constituting the dataset

Type	Configurations
Transcription	-
Dense	#units: chars + 1, activation: softmax, regularization: L2
Bi-LSTM	#hidden units: 128, d: 0.2
Bi-LSTM	#hidden units: 128, d: 0.2
Map-To-Sequence	-
Convolution	#maps: 512, K: 2 × 2, S: 1, P: Valid
MaxPooling	Window: 2 × 1, S: 2 × 1
BatchNormalization	-
Activation	relu
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
BatchNormalization	-
Activation	relu
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 1, S: 2 × 1
Activation	relu
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
Activation	relu
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	relu
Convolution	#maps: 128, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	relu
Convolution	#maps: 64, K: 3 × 3, S: 1, P: Same
Input	32 × 128 the size of a black and white image

4.2.2. Traditional augmentation techniques

Four main traditional augmentation strategies are used to augment the images of the database we used to train the baseline model. These augmentation strategies are easy to implement because they use simple image-processing techniques. We applied rotation, translation, and dilation, where the images of the database are rotated around the center of the image in a 4-degree counter-clockwise direction, and they are also rotated by another 4 degrees, but this time in a clockwise direction. The text images are also thickened by dilating the images to make the text present in the images thicker. The images are finally translated by adding white pixels to the left of the images, allowing the text to move to the right of the image. These four traditional data augmentation

strategies were chosen because they suit the case of Arabic handwritten text, and they don't fundamentally change the nature of the text present in the images. An example of these augmentations is shown in Figure 4.1.

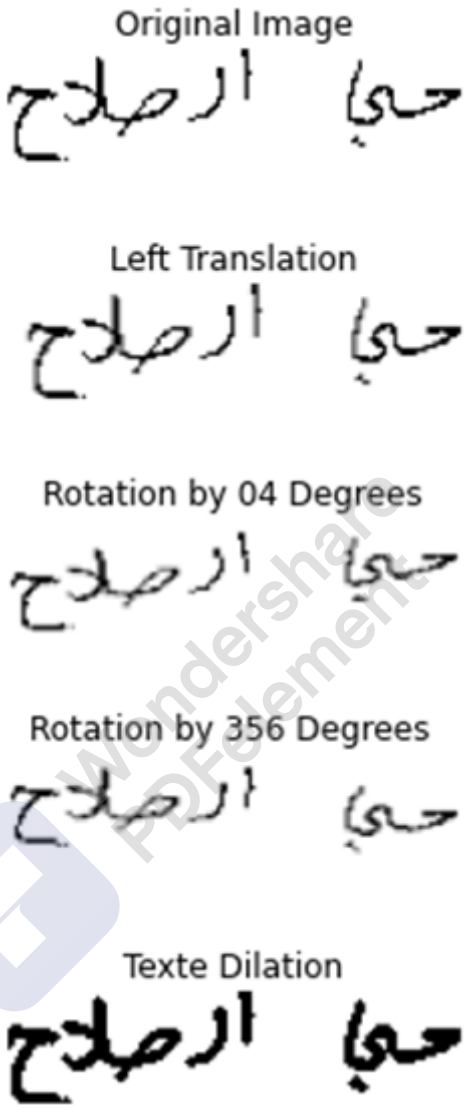


Figure 4.1: An example of the four traditional augmentation techniques: Original image, left translation, rotation by 04 degrees, rotation by 356 degrees, and text dilation

4.2.3. The proposed data augmentation method

To enrich the training data of the Arabic HTR deep learning system, we used an image deformation technique named Moving Least Squares (MLS), first described in [117]. The authors of the MLS paper have created many deformation transformations, and we chose to use the similarity deformation transformation because it suits our Arabic case very well. We use the MLS technique to create new Arabic handwritten text images that can be very similar to the original images, or significantly different from them, and this

is based on the parameters of the MLS deformation that are applied to the images, as different parameters may yield very different results. The parameters of the deformation transformation are a set of fiducial points. For each transformation, we need a set of fiducial points of the starting position of the transformation, and a set of fiducial points of the new positions of these fiducial points, where these fiducial points will move from the starting position to the new position.

The length of the sequence of characters L present in each image determines the number of fiducial points that will be used to deform the image. For each character present in the image, we randomly select 4 fiducial points, resulting in a total of $4L$ number of fiducial points for each image. We randomly select the coordinates of the fiducial points p by randomly sampling the coordinates from a discrete uniform distribution made of the size of the image. For example, the image input size of the baseline model in this chapter is (32, 128), where 32 is the height, 128 is the width of the image, and the coordinates of one of the fiducial points will be in the range ([0, height of the image], [0, width of the image]), making it in the range ([0, 32], [0, 128]). After that, we augment the images by randomly moving the fiducial points p to the new coordinates q within a random radius of R. To get the new augmented image, we apply the similarity deformation based on Moving Least Squares (MLS) on an input image. Given a point v from the input image, the transformation for this v based on the fiducial points p when moved to the new positions q is

$$T_v(q) = (v - p_*)M + q_* \qquad (9)$$

where, M is a linear transformation matrix of a constant size of 2×2, and it is constrained to have the property of $M^T M = \lambda^2 I$ for some scalar λ. The p_* and q_* are the weighted centroids of the initialized fiducial points p and q respectively

$$p_* = \frac{\sum_{i=1}^{4L} w_i p_i}{\sum_{i=1}^{4L} w_i}, \; q_* = \frac{\sum_{i=1}^{4L} w_i q_i}{\sum_{i=1}^{4L} w_i}, \qquad (10)$$

In the above equations, p_i and q_i are row vectors and the weights w_i for the point v have the form

$$w_i = \frac{1}{|p_i - v|^{2a}}, \; p_i \neq v \qquad (11)$$

We can observe that when v approaches p_i , w_i increases significantly, which means that $T_v(p_i) = q_i$. And it also means that the points close to p_i will be also close to the

new point q_i in the deformed image. Note that, if $p_i = q_i$, then each $T_v(v) = v$ for all v and, thus, T is the identity transformation function. We obtain the best transformation $T_v(v)$ by minimizing

$$\sum_{i=1}^{4L} w_i |T_v(p_i) - q_i|^2 \qquad (12)$$

4.3.Experiments

In this section, we will mention the dataset used for training and augmentation, we will discuss the training details of the baseline model, we will discuss the training details of the baseline model with images augmented with traditional data augmentation techniques, we will discuss the training details of the baseline model with images augmented using the MLS-based augmentation technique, and we will finally discuss the results and see the role of employing such augmentation techniques to improve the performance of the baseline model.

4.3.1. Dataset details.

The experiments were carried out using the IFN/ENIT database [22].

4.3.2. Training details of baseline model (small input-size model)

Most images in the IFN/ENIT database are of large dimensions; in fact, they often surpass 200 pixels in height and 1000 pixels in width. The CRNN model that was designed in the previous chapter was specifically designed to deal with large-sized images, typically images of size (64, 512). The motivation behind making the model accept such big-sized images is to conserve as much information from the images as possible. However, in this chapter, we have designed a new variation of the CRNN model to be used as a baseline with an image input size of (32, 128). We wanted to test whether reducing the size of the images would drastically change the performance of the model.

The network architecture was implemented in Python 3 and TensorFlow v2. The large input-size CRNN model of the previous chapter has a much bigger number of parameters than the small input-size CRNN model developed in this chapter due to the number of layers found in the former model, as more layers often yield a much bigger number of parameters. The baseline model in this chapter has 6,634,617 parameters, while the model of the previous chapter has 9,186,105, a reduction of 2,551,488

parameters. One of the gains of using the small input-size model (baseline model) instead of the other model appears in training, where the large input-size model needs 08 times more memory space for the RAM and the GPU RAM when training.

The baseline model (small input-size model) was trained with a batch size of 256 using the GPU: Tesla P100 with 16GB of RAM and Tesla T4 with 16GB of RAM. The training was stopped when reaching convergence at epoch number 19. The Adam optimizer with a learning rate of 0.001 was used. The training lasted 36 minutes, with 114 seconds for each epoch.

4.3.3. Training details of the CRNN model with traditional augmented images

The baseline model (small input-size model) was trained with the IFN/ ENIT database alongside all the images of the database augmented four times with the traditional augmentation techniques, the rotation in the two directions, the translation, and dilation of the images. During training, the model will receive the original image and the four augmented images consecutively. Thus, the model will train with five times more data than the training without augmented data.

4.3.4. Training details of the CRNN model with MLS augmented images

The MLS technique is applied to the images of the IFN/ ENIT database to obtain three new images for each image of the database. First, the control points p are randomly selected for each image that is to be augmented. Then we randomly select the destination points q within a radius R from the original control points p. The distance (radius R) between the control points and the destination points determines how much deformation will be applied to the image. The farther the points are from each other, the more deformation will be applied. Where the radius is too big, the resulting images will be unrecognizable from the original image. For that, we manually experimented with many values for the radius R and visually observed the resulting images to choose the best radius R values for our augmentation case. The images will be deformed three times with the MLS technique to obtain three new versions of the images of the database. We chose three radius R values that will be used to generate three new images that still keep the text present in the images readable. The first deformation will be soft, and it will not change much of the text present in the original image. The second

deformation will be a little bit harder than the previous one, and it will result in an image with text much different than the original image. The third deformation is the one with the most change to the original image, and it will result in a distorted text when compared to the original image. The three deformations are named based on their intensity, respectively, soft deformation, medium deformation, and hard deformation. Some examples of images augmented with these deformations are shown in Figure 4.2.

Figure 4.2: Three examples of augmented images using the MLS augmentation technique with: Soft deformation, medium deformation, and hard deformation

We trained the baseline CRNN model with the original images alongside the three versions of the images obtained from the augmentation with the soft deformation, medium deformation, and the hard deformation. The model was trained until convergence.

In Figure 4.3 we respectively plot the training loss, and the validation loss of our models (Large Input Size Model, Small Input Size Model, Small Input Size Model with Traditional Augmented Images, and Small Input Size Model with MLS Augmented Images) as a function of the number of epochs.

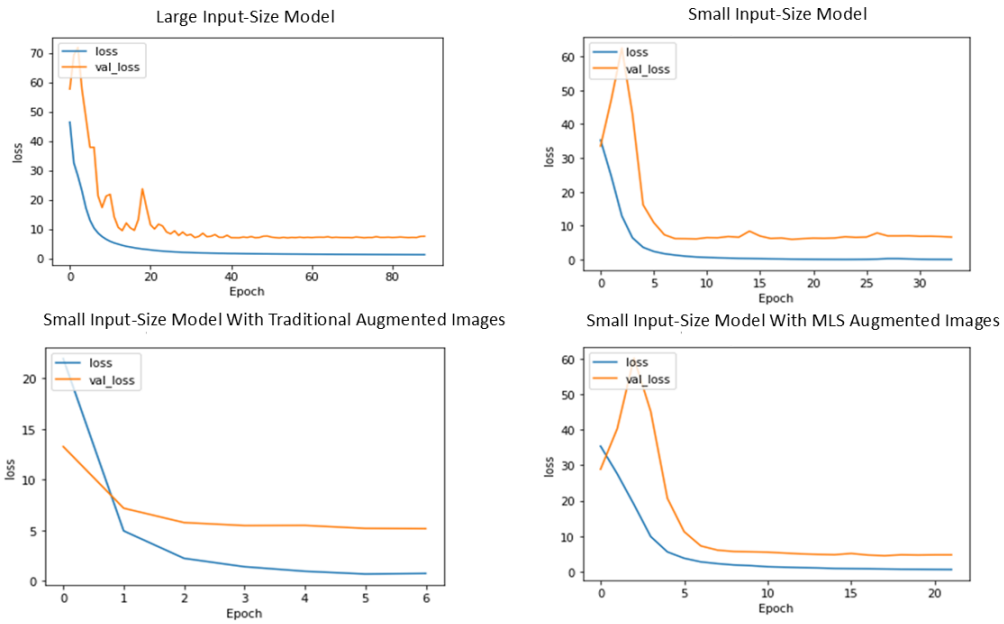


Figure 4.3: Losses during training and validation as a function of epoch count

4.3.5. Results and discussion

We carried out all of our experiments on the IFN/ENIT database with a popular training scenario, where we trained with sets a, b, c, and d, validated on set e, and tested on set f. Figure 4.2 shows all the experiments. The Word Accuracy Rate (WAR) metric was used to assess the performance of the models. The sequences of characters predicted by the models were converted to the closest Tunisian city names found in the IFN/ENIT database using a search with a BK-Tree data structure over the list of Tunisian city names. The WAR metric was calculated after converting all the predicted sequences of characters to their corresponding truth labels using the BK-Tree data structure.

Table 4.2: Accuracy of word recognition in training, validation, and testing conducted on the IFN/ENIT database for all experiments

IFN/ENIT Dataset Version: v2.0p1e					
		Word Accuracy %			
	Sets	Large Input-Size Model	Small Input-Size Model	Small Input-Size Model with Traditional Augmented Images	Small Input-Size Model with MLS Augmented Images
Training	A, B, C, and D	99.9	99.9	99.9	99.9
Validation	E	80.97	81.02	**81.17**	80.82
Testing	F	79.95	81.54	80.59	**81.81**

The CRNN model of the previous chapter with large input-size delivered the weakest performance on the test set f, especially when comparing it with the baseline model (small input-size model), where the latter model surpassed the former model by 1.59%. Demonstrating that there is no need to use bigger Arabic HTR models with larger input sizes, as smaller models with smaller input sizes will suffice. Additionally, the experiment conducted with the CRNN model trained with data augmented with the MLS technique yielded better results on the test set f compared with the other experiments, surpassing the model with the closest result by 0.27%. The results are compared with other systems in Figure 4.3.

Table 4.3: Accuracy of word recognition across many systems that were evaluated on set e and trained on sets a, b, c, and d

System	Word Accuracy %
Parvez and Mahmoud (2013) [111]	79.58
CNN-HMM (Amrouch et al. 2018 [109])	89.23
Alkhateeb et al. (2011) [112]	DBN (66.65) and HMM (82.3)
El Moubtahij et al. (2017) [110]	78.95
Hamdani et al. (2009) [113]	81.93
Kessentini et al. (2012) [114]	79.6
DBN (Jayech et al. 2016) [115]	78.5
Our model with MLS Augmented Images	**81.17**

4.4.Conclusion

In this work, we explored the effects of augmented data on the training procedure of Arabic deep learning Handwritten Text Recognition (HTR) systems. We developed a new data augmentation technique based on the Moving Least Squares (MLS) image processing method to generate new images containing authentic Arabic handwritten text. Our augmentation method exhibited promising results when compared with the experiments carried out with other augmentation techniques, and without the employment of any augmentation techniques. Moreover, we designed a new variation of the CRNN model with a smaller size and smaller image input size. The new small CRNN model outperformed the bigger CRNN model, and was more efficient during training, where it cut the memory use by a factor of 08 compared to the big CRNN model, and converged 04 times faster, and reduced the number of parameters by 27%, making it more suitable for use cases where faster recognition is more desirable. Further

research must be conducted on the MLS technique for data augmentation purposes to get a robust understanding of all the capabilities and potential limitations of MLS-based augmentation techniques for Arabic HTR.

5. Conclusion and Future Work

This thesis presented a novel data augmentation technique based on Moving Least Squares (MLS) to augment Arabic handwritten texts. Before that, a new Convolutional Recurrent Neural Network (CRNN) architecture was developed and employed as a baseline for experimental evaluation. The proposed data augmentation method showed promising results in improving the performance of Arabic Handwriting Text Recognition systems when the augmented samples were included in the training data of the recognition systems. The development of this augmentation technique was motivated by two main reasons: the lack of Arabic handwritten text databases containing an extensive volume of samples to be used for training the recognition systems, and the limited number of data augmentation techniques specifically designed for Arabic handwritten text. We first conducted a comprehensive review of all the available Arabic handwritten text databases to identify the gap. Our research shows that many Arabic handwritten text databases exist. However, they are dispersed across various sources, and all contain too few samples to enable deep learning-based recognition systems to generalize beyond their training distributions. Thus, making them best suited for narrow, domain-specific text recognition tasks. Another thing about these databases is that they are designed in a way that makes them unable to statistically represent all the character shapes present in the Arabic language. As a result, the databases are severely imbalanced. For the previously mentioned reasons, data augmentation research to augment Arabic handwritten text was conducted. We explored existing Arabic handwritten text augmentation techniques through a comprehensive review. We found that the field of Arabic handwritten text data augmentation remains relatively underexplored. Notably, most research on Arabic handwritten text data augmentation fails to consider the statistical representation of Arabic character shapes. Thus, leaving much space for potential research in future work. Equally important, future work on designing better Arabic handwritten text databases with improved statistical representation of the character shapes could potentially lead to better recognition systems.

6. References:

1. Kellogg, D. A. (1960). Modern Trends in Character Recognition Systems. In *Proceedings of the National Symposium on Machine Translation*.
2. Lindgren, N. (1965). Machine recognition of human language Part III-Cursive script recognition. *IEEE spectrum*, *2*(5), 104-116.
3. Suen, C. Y., Shinghal, R., & Kwan, C. C. (1977, September). Dispersion factor: A quantitative measurement of the quality of handprinted characters. In *International Conference of cybernetics and Society* (pp. 681-685).
4. Suen, C. Y. (1973, November). Factors affecting the recognition of handprinted characters. In *Proc. Int. Conf. Cybernetics and Society* (pp. 174-175).
5. Stevens, M. E. (1970). Introduction to the special issue on optical character recognition (OCR). *Pattern Recognition*, *2*(3), 147-150.
6. Suen, C. Y., Berthod, M., & Mori, S. (1980). Automatic recognition of handprinted characters—the state of the art. *Proceedings of the IEEE*, *68*(4), 469-487.
7. Al-Badr, B., & Mahmoud, S. A. (1995). Survey and bibliography of Arabic optical text recognition. *Signal processing*, *41*(1), 49-77.
8. Parvez, M. T., & Mahmoud, S. A. (2013). Offline Arabic handwritten text recognition: a survey. *ACM Computing Surveys (CSUR)*, *45*(2), 1-35.
9. Khorsheed, M. S. (2002). Off-line Arabic character recognition–a review. *Pattern analysis & applications*, *5*, 31-45.
10. Plötz, T., & Fink, G. A. (2009). Markov models for offline handwriting recognition: a survey. *International Journal on Document Analysis and Recognition (IJDAR)*, *12*, 269-298.
11. Arica, N., & Yarman-Vural, F. T. (2001). An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *31*(2), 216-233.

12. Momeni, S., & BabaAli, B. (2024). A transformer-based approach for Arabic offline handwritten text recognition. Signal, Image and Video Processing, 18(4), 3053-3062.
13. Ashiquzzaman, A., Tushar, A. K., Rahman, A., & Mohsin, F. (2019). An efficient recognition method for handwritten arabic numerals using CNN with data augmentation and dropout. In Data Management, Analytics and Innovation: Proceedings of ICDMAI 2018, Volume 1 (pp. 299-309). Springer Singapore.
14. Balaha, H. M., Ali, H. A., Youssef, E. K., Elsayed, A. E., Samak, R. A., Abdelhaleem, M. S., ... & Mohammed, M. M. (2021). Recognizing arabic handwritten characters using deep learning and genetic algorithms. Multimedia Tools and Applications, 80, 32473-32509.
15. Wagaa, N., Kallel, H., & Mellouli, N. (2022). Improved Arabic alphabet characters classification using convolutional neural networks (CNN). Computational Intelligence and Neuroscience, 2022(1), 9965426.
16. Rabi, M., & Amrouche, M. (2024). Enhancing Arabic Handwritten Recognition System-Based CNN-BLSTM Using Generative Adversarial Networks. European Journal of Artificial Intelligence and Machine Learning, 3(1), 10-17.
17. Ahamed, P., Kundu, S., Khan, T., Bhateja, V., Sarkar, R., & Mollah, A. F. (2020). Handwritten Arabic numerals recognition using convolutional neural network. Journal of Ambient Intelligence and Humanized Computing, 11, 5445-5457.
18. Das, N., Reddy, J. M., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., & Basu, D. K. (2012). A statistical–topological feature combination for recognition of handwritten numerals. Applied Soft Computing, 12(8), 2486-2495.
19. Eltay, M., Zidouri, A., Ahmad, I., & Elarian, Y. (2022). Generative adversarial network based adaptive data augmentation for handwritten Arabic text recognition. PeerJ Computer Science, 8, e861.
20. Eltay, M., Zidouri, A., & Ahmad, I. (2020). Exploring deep learning approaches to recognize handwritten arabic texts. *IEEE Access*, *8*, 89882-89898.

21. Sulaiman, A., Omar, K., & Nasrudin, M. F. (2021). Two streams deep neural network for handwriting word recognition. Multimedia Tools and Applications, 80(4), 5473-5494.
22. Pechwitz, M., Maddouri, S. S., Märgner, V., Ellouze, N., & Amiri, H. (2002, October). IFN/ENIT-database of handwritten Arabic words. In *Proc. of CIFED* (Vol. 2, pp. 127-136). Citeseer.
23. Mahmoud, S. A., Ahmad, I., Al-Khatib, W. G., Alshayeb, M., Parvez, M. T., Märgner, V., & Fink, G. A. (2014). KHATT: An open Arabic offline handwritten text database. *Pattern Recognition*, *47*(3), 1096-1112.
24. Lawgali, A., Angelova, M., & Bouridane, A. (2013, June). HACDB: Handwritten Arabic characters database for automatic character recognition. In *European workshop on visual information processing (EUVIP)* (pp. 255-259). IEEE.
25. AlKhateeb, J. H. (2015). A database for Arabic handwritten character recognition. *Procedia Computer Science*, *65*, 556-561.
26. Al-Ma'adeed, S., Elliman, D., & Higgins, C. A. (2002, August). A data base for Arabic handwritten text recognition research. In *Proceedings eighth international workshop on frontiers in handwriting recognition* (pp. 485-489). IEEE.
27. Abdalkafor, A. S., Allawi, E. T., Al-Ani, K. W., & Nassar, A. M. (2019, May). A novel database for Arabic handwritten recognition (ndahr) system. In *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)* (pp. 1-6). IEEE.
28. Kharma, N., Ahmed, M., & Ward, R. (1999, May). A new comprehensive database of handwritten Arabic words, numbers, and signatures used for OCR testing. In *Engineering Solutions for the Next Millennium. 1999 IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No. 99TH8411)* (Vol. 2, pp. 766-768). IEEE.
29. Mozaffari, S., Faez, K., Faradji, F., Ziaratban, M., & Golzan, S. M. (2006, October). A comprehensive isolated Farsi/Arabic character database for handwritten OCR research. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft.

30. Elzobi, M., Al-Hamadi, A., Al Aghbari, Z., & Dings, L. (2013). IESK-ArDB: a database for handwritten Arabic and an optimized topological segmentation approach. *International Journal on Document Analysis and Recognition (IJDAR)*, *16*, 295-308.
31. Chtourou, I., Rouhou, A. C., Jaiem, F. K., & Kanoun, S. (2015, August). ALTID: Arabic/Latin text images database for recognition research. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)* (pp. 836-840). IEEE.
32. Lamghari, N., & Raghay, S. (2017). DBAHCL: database for Arabic handwritten characters and ligatures. *International Journal of Multimedia Information Retrieval*, *6*, 263-269.
33. Mahmoud, S. A., Ahmad, I., Alshayeb, M., & Al-Khatib, W. G. (2011). A database for offline Arabic handwritten text recognition. In *Image Analysis and Recognition: 8th International Conference, ICIAR 2011, Burnaby, BC, Canada, June 22-24, 2011. Proceedings, Part II 8* (pp. 397-406). Springer Berlin Heidelberg.
34. Mezghani, A., Kanoun, S., Khemakhem, M., & El Abed, H. (2012, September). A database for arabic handwritten text image recognition and writer identification. In *2012 international conference on frontiers in handwriting recognition* (pp. 399-402). IEEE.
35. Ramdan, J., Omar, K., Faidzul, M., & Mady, A. (2013). Arabic handwriting data base for text recognition. *Procedia Technology*, *11*, 580-584.
36. Arica, N., & Yarman-Vural, F. T. (2001). An overview of character recognition focused on off-line handwriting. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 31(2), 216-233.
37. Tsujimoto, S., & Asada, H. (1992). Major components of a complete text reading system. Proceedings of the IEEE, 80(7), 1133-1149.
38. Amara, M., & Zaghdoud, R. (2022). Optical Character Recognition using Deep Learning: An enhanced Approach. IJCSNS, 22(5), 545.
39. Hennis, R. B. (1968). The IBM 1975 optical page reader Part I: System design. IBM Journal of Research and Development, 12(5), 346-353.

40. Wang, J., & Jean, J. (1993, March). Segmentation of merged characters by neural networks and shortest-path. In Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice (pp. 762-769).
41. Boraik, O. A., Ravikumar, M., & Saif, M. A. N. (2022). Characters segmentation from Arabic handwritten document images: hybrid approach. International Journal of Advanced Computer Science and Applications, 13(4).
42. Gilloux, M. (1994). Hidden Markov models in handwriting recognition. In Fundamentals in Handwriting Recognition (pp. 264-288). Springer Berlin Heidelberg.
43. Chen, C. H., & DeCurtins, J. L. (1993, October). Word recognition in a segmentation-free approach to OCR. In Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93) (pp. 573-576). IEEE.
44. Sari, T., & Sellami, M. (2007). Overview of Some Algorithms of Off-Line Arabic Handwriting Segmentation. Int. Arab J. Inf. Technol., 4(4), 289-300.
45. Almuallim, H., & Yamaguchi, S. (1987). A method of recognition of Arabic cursive handwriting. IEEE transactions on pattern analysis and machine intelligence, (5), 715-722.
46. Zahour, A., Taconet, B., & Faure, A. Une méthode de reconnaissance structurelle de l'arabe écrit. Proceedings Actes RFIA, 87, 1521-1530.
47. Olivier, G., Miled, H., Romeo, K., & Lecourtier, Y. (1996, August). Segmentation and coding of Arabic handwritten words. In Proceedings of 13th International Conference on Pattern Recognition (Vol. 3, pp. 264-268). IEEE.
48. Cheriet, M., Miled, H., Olivier, C., & Lecourtier, Y. (1998). Visual aspect of cursive Arabic handwriting recognition. Proceedings of VI, 99, 263-270.
49. Sari, T., Souici, L., & Sellami, M. (2002, August). Off-line handwritten Arabic character segmentation algorithm: ACSA. In Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition (pp. 452-457). IEEE.

50. Romeo-Pakker, K., Miled, H., & Lecourtier, Y. (1995, August). A new approach for Latin/Arabic character segmentation. In Proceedings of 3rd International Conference on Document Analysis and Recognition (Vol. 2, pp. 874-877). IEEE.
51. Motawa, D., Amin, A., & Sabourin, R. (1997, August). Segmentation of Arabic cursive script. In Proceedings of the fourth international conference on document analysis and recognition (Vol. 2, pp. 625-628). IEEE.
52. Wang, S. S., Chen, P. C., & Lin, W. G. (1994). Invariant pattern recognition by moment Fourier descriptor. Pattern Recognition, 27(12), 1735-1742.
53. Hamamoto, Y., Uchimura, S., Masamizu, K., & Tomita, S. (1995, August). Recognition of handprinted Chinese characters using Gabor features. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 2, pp. 819-823). IEEE.
54. Lee, S. W., & Kim, Y. J. (1995, August). Multiresolution recognition of handwritten numerals with wavelet transform and multilayer cluster neural network. In *Proceedings of 3rd International Conference on Document Analysis and Recognition* (Vol. 2, pp. 1010-1013). IEEE.
55. Shioyama, T., Wu, H. Y., & Nojima, T. (1998, August). Recognition algorithm based on wavelet transform for handprinted Chinese characters. In *Proceedings. Fourteenth international conference on pattern recognition (Cat. No. 98EX170)* (Vol. 1, pp. 229-232). IEEE.
56. Chim, Y. C., Kassim, A. A., & Ibrahim, Y. (1999). Character recognition using statistical moments. Image and vision computing, 17(3-4), 299-307.
57. Kim, W. Y., & Yuan, P. (1994, June). A practical pattern recognition system for translation, scale and rotation invariance. In 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (pp. 391-396). IEEE.
58. Impedovo, S., Pirlo, G., Modugno, R., & Ferrante, A. (2010, November). Zoning methods for hand-written character recognition: An overview. In 2010 12th International Conference on Frontiers in Handwriting Recognition (pp. 329-334). IEEE.

59. Siddharth, K. S. (2011). HANDWRITTEN GURMUKHI CHARACTER RECOGNITION (Doctoral dissertation, NATIONAL INSTITUTE OF TECHNOLOGY JALANDHAR).
60. Mohamed, M., & Gader, P. (1996). Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. IEEE transactions on pattern analysis and machine intelligence, 18(5), 548-554.
61. Arica, N., & Yarman-Vural, F. T. (2000). One-dimensional representation of two-dimensional information for HMM based handwriting recognition. Pattern recognition letters, 21(6-7), 583-592.
62. Brown, M. K., & Ganapathy, S. (1983). Preprocessing techniques for cursive script word recognition. Pattern Recognition, 16(5), 447-458.
63. Heutte, L., Paquet, T., Moreau, J. V., Lecourtier, Y., & Olivier, C. (1998). A structural/statistical feature based vector for handwritten character recognition. Pattern recognition letters, 19(7), 629-641.
64. Naz, S., Umar, A. I., Ahmed, S. B., Ahmad, R., Shirazi, S. H., Razzak, M. I., & Zaman, A. (2018). STATISTICAL FEATURES EXTRACTION FOR CHARACTER RECOGNITION USING RECURRENT NEURAL NETWORK. Pakistan Journal of Statistics, 34(1).
65. Tao, Y., & Tang, Y. Y. (1999, September). The feature extraction of chinese character based on contour information. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)* (pp. 637-640). IEEE.
66. Kai, W., Yang, Y. Y., & Suen, C. Y. (1988, January). Multi-layer projections for the classification of similar Chinese characters. In 9th International Conference on Pattern Recognition (pp. 842-843). IEEE Computer Society.
67. Guillevic, D., & Suen, C. Y. (1998). Recognition of legal amounts on bank cheques. Pattern Analysis and Applications, 1, 28-41.
68. Madhvanath, S., Kim, G., & Govindaraju, V. (1999). Chaincode contour processing for handwritten word recognition. IEEE transactions on pattern analysis and machine intelligence, 21(9), 928-932.

69. Madhvanath, S., & Govindaraju, V. (1999). Reference lines for holistic recognition of handwritten words. Pattern Recognition, 32(12), 2021-2028.
70. Abdelazim, H., & Hashish, M. (1989, March). Automatic recognition of handwritten Hindi numerals. In Proceedings of the 11th National Computer Conference, Dhahran, Saudi Arabia (pp. 287-298).
71. Amin, A. (1998, April). Recognition of printed Arabic text using machine learning. In Document Recognition V (Vol. 3305, pp. 62-71). SPIE.
72. Guillevic, D., & Suen, C. Y. (1998, August). HMM-KNN word recognition engine for bank cheque processing. In Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170) (Vol. 2, pp. 1526-1529). IEEE.
73. Althobaiti, H., & Lu, C. (2017, March). A survey on Arabic optical character recognition and an isolated handwritten Arabic character recognition algorithm using encoded freeman chain code. In 2017 51st Annual conference on information sciences and systems (CISS) (pp. 1-6). IEEE.
74. Althobaiti, H., Shah, K., & Lu, C. (2017, September). Isolated handwritten arabic character recognition using freeman chain code and tangent line. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems (pp. 79-84).
75. Abed, M. A. (2012). Freeman chain code contour processing for handwritten isolated Arabic characters recognition. Alyrmook University Magazine, Baghdad.
76. Fethi, M. R., Farhaoui, O., Zeroual, I., & Allaoui, A. E. (2024, April). An Advanced Modified Freeman Chain Code Algorithm for Enhancing Arabic Character Recognition. In The International Workshop on Big Data and Business Intelligence (pp. 438-444). Cham: Springer Nature Switzerland.
77. Govindan, V. K., & Shivaprasad, A. P. (1990). Character recognition—a review. *Pattern recognition*, *23*(7), 671-683.
78. Beigi, H. S. (1993, July). An overview of handwriting recognition. In Proceedings of the 1st annual conference on technological

advancements in developing countries, Columbia University, New York (pp. 30-46).

79. El-Hajj, R., Likforman-Sulem, L., & Mokbel, C. (2005, August). Arabic handwriting recognition using baseline dependant features and hidden Markov modeling. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)* (pp. 893-897). IEEE.
80. AlKhateeb, J. H., Ren, J., Jiang, J., & Al-Muhtaseb, H. (2011). Offline handwritten Arabic cursive text recognition using Hidden Markov Models and re-ranking. *Pattern Recognition Letters*, *32*(8), 1081-1088.
81. Rabi, M., Amrouch, M., & Mahani, Z. (2018). Recognition of cursive Arabic handwritten text using embedded training based on hidden Markov models. *International journal of pattern recognition and Artificial Intelligence*, *32*(01), 1860007.
82. Alma'adeed, S., Higgens, C., & Elliman, D. (2002, August). Recognition of off-line handwritten Arabic words using hidden Markov model approach. In *2002 International Conference on Pattern Recognition* (Vol. 3, pp. 481-484). IEEE.
83. Azeem, S. A., & Ahmed, H. (2013). Effective technique for the recognition of offline Arabic handwritten words using hidden Markov models. *International Journal on Document Analysis and Recognition (IJDAR)*, *16*, 399-412.
84. Jbrail, M. W., & Tenekeci, M. E. (2022). Character recognition of Arabic handwritten characters using deep learning. Journal of Studies in Science and Engineering, 2(1), 32-40.
85. Younis, K. S. (2017). Arabic hand-written character recognition based on deep convolutional neural networks. Jordanian Journal of Computers and Information Technology, 3(3).
86. Balaha, H. M., Ali, H. A., Youssef, E. K., Elsayed, A. E., Samak, R. A., Abdelhaleem, M. S., ... & Mohammed, M. M. (2021). Recognizing arabic handwritten characters using deep learning and genetic algorithms. Multimedia Tools and Applications, 80, 32473-32509.
87. El Khayati, M., Kich, I., & Taouil, Y. (2024). CNN-based Methods for Offline Arabic Handwriting Recognition: A Review. Neural Processing Letters, 56(2), 115.

88. Lamtougui, H., El Moubtahij, H., Fouadi, H., & Satori, K. (2024). Improving Arabic handwritten text recognition through transfer learning with convolutional neural network-based models. Bulletin of Electrical Engineering and Informatics, 13(6), 4294-4305.
89. Al-Shatnawi, A. M., Al-Saqqar, F., & Souri, A. (2021). Arabic handwritten word recognition based on stationary wavelet transform technique using machine learning. Transactions on Asian and Low-Resource Language Information Processing, 21(3), 1-21.
90. Haboubi, S., Guesmi, T., Alshammari, B. M., Alqunun, K., Alshammari, A. S., Alsaif, H., & Amiri, H. (2022). Improving CNN-BGRU Hybrid Network for Arabic Handwritten Text Recognition. Computers, Materials & Continua, 73(3).
91. Awni, M., Khalil, M. I., & Abbas, H. M. (2022). Offline Arabic handwritten word recognition: A transfer learning approach. Journal of King Saud University-Computer and Information Sciences, 34(10), 9654-9661.
92. Hussein, M. E., Torki, M., Elsallamy, A., & Fayyaz, M. (2014). Alexu-word: a new dataset for isolated-word closed-vocabulary offline arabic handwriting recognition. arXiv preprint arXiv:1411.4670.
93. Hassen, H., Al-Madeed, S., & Bouridane, A. (2021). Subword Recognition in Historical Arabic Documents using C-GRUs. TEM Journal, 10(4).
94. Farrahi Moghaddam, R., Cheriet, M., Adankon, M. M., Filonenko, K., & Wisnovsky, R. (2010, June). IBN SINA: a database for research on processing and understanding of Arabic manuscripts images. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems (pp. 11-18).
95. Kassis, M., Abdalhaleem, A., Droby, A., Alaasam, R., & El-Sana, J. (2017, April). Vml-hd: The historical arabic documents dataset for recognition systems. In 2017 1st international workshop on Arabic script analysis and recognition (ASAR) (pp. 11-14). IEEE.
96. AL-Taee, M. M., Neji, S. B. H., Frikha, M., & Allawi, S. T. (2024). Using Faster R-CNN to Detect and Recognize Arabic Handwritten

Words. International Journal of Intelligent Engineering & Systems, 17(3).

97. Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006, June). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning (pp. 369-376).
98. Anjum, T., & Khan, N. (2023). CALText: Contextual attention localization for offline handwritten text. Neural Processing Letters, 55(6), 7227-7257.
99. Lamtougui, H., El Moubtahij, H., Fouadi, H., & Satori, K. (2023). An efficient hybrid model for Arabic text recognition. Computers, Materials & Continua, 74(2), 2871-2888.
100. Jemni, S. K., Ammar, S., & Kessentini, Y. (2022). Domain and writer adaptation of offline Arabic handwriting recognition using deep neural networks. Neural Computing and Applications, 34(3), 2055-2071.
101. Noubigh, Z., Mezghani, A., & Kherallah, M. (2021). Contribution on Arabic handwriting recognition using deep neural network. In Hybrid Intelligent Systems: 19th International Conference on Hybrid Intelligent Systems (HIS 2019) held in Bhopal, India, December 10-12, 2019 19 (pp. 123-133). Springer International Publishing.
102. Noubigh, Z., Mezghani, A., & Kherallah, M. (2020, November). Transfer Learning to improve Arabic handwriting text Recognition. In 2020 21st International Arab Conference on Information Technology (ACIT) (pp. 1-6). IEEE.
103. Elsayed, M., Alnaggar, A., Abdeen, M., Wahdan, A., & Gomaa, W. Arabic Handwritten Text Recognition using Advanced CNN-RNN Architecture.
104. Graves, A., & Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. Advances in neural information processing systems, 21.
105. Märgner, V., & El Abed, H. (2009, July). ICDAR 2009 Arabic handwriting recognition competition. In 2009 10th International

Conference on Document Analysis and Recognition (pp. 1383-1387). IEEE.

106. Maalej, R., & Kherallah, M. (2022). New MDLSTM-based designs with data augmentation for offline Arabic handwriting recognition. Multimedia Tools and Applications, 81(7), 10243-10260.

107. Jayech, K., Mahjoub, M. A., & Amara, N. E. B. (2016). Synchronous multi-stream hidden markov model for offline Arabic handwriting recognition without explicit segmentation. Neurocomputing, 214, 958-971.

108. Metwally, A. H., Khalil, M. I., & Abbas, H. M. (2017, December). Offline Arabic handwriting recognition using hidden Markov models and post-recognition lexicon matching. In 2017 12th International Conference on Computer Engineering and Systems (ICCES) (pp. 238-243). IEEE.

109. Amrouch, M., Rabi, M., & Es-Saady, Y. (2018). 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 8 (pp. 265-274). Springer International Publishing.

110. Akram, H., & Khalid, S. (2017). Using features of local densities, statistics and HMM toolkit (HTK) for offline Arabic handwriting text recognition. *Journal of Electrical Systems and Information Technology*, *4*(3), 387-396.

111. Parvez, M. T., & Mahmoud, S. A. (2013). Arabic handwriting recognition using structural and syntactic pattern attributes. Pattern Recognition, 46(1), 141-154.

112. AlKhateeb, J. H., Pauplin, O., Ren, J., & Jiang, J. (2011). Performance of hidden Markov model and dynamic Bayesian network classifiers on handwritten Arabic word recognition. knowledge-based systems, 24(5), 680-688.

113. Hamdani, M., El Abed, H., Kherallah, M., & Alimi, A. M. (2009, July). Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition. In 2009 10th International

Conference on Document Analysis and Recognition (pp. 201-205). IEEE.

114. Kessentini, Y., Paquet, T., & Hamadou, A. B. (2010). Off-line handwritten word recognition using multi-stream hidden Markov models. Pattern Recognition Letters, 31(1), 60-70.

115. Jayech, K., Mahjoub, M. A., & Amara, N. E. B. (2016). Arabic handwritten word recognition based on dynamic bayesian network. *Int. Arab J. Inf. Technol.*, *13*(6B), 1024-1031.

116. Chadli, M. A., Bachir Bouiadjra, R., & Fekir, A. (2022, November). Offline arabic handwritten text recognition for unsegmented words using convolutional recurrent neural network. In International Conference on Artificial Intelligence: Theories and Applications (pp. 276-287). Cham: Springer Nature Switzerland.

117. Schaefer, S., McPhail, T., & Warren, J. (2006). Image deformation using moving least squares. In ACM SIGGRAPH 2006 Papers (pp. 533-540).

118. Balaha, H. M., Ali, H. A., Saraya, M., & Badawy, M. (2021). A new Arabic handwritten character recognition deep learning system (AHCR-DLS). Neural Computing and Applications, 33, 6325-6367.

119. El-Sawy, A., Loey, M., & El-Bakry, H. (2017). Arabic handwritten characters recognition using convolutional neural network. WSEAS Transactions on Computer Research, 5(1), 11-19.

120. Altwaijry, N., & Al-Turaiki, I. (2021). Arabic handwriting recognition system using convolutional neural network. Neural Computing and Applications, 33(7), 2249-2261.

121. Al-Muhtaseb, H. A., Mahmoud, S. A., Qahwaji, R. S., Demiralp, M., Baykara, N., & Mastorakis, N. (2009, June). A novel minimal Arabic script for preparing databases and benchmarks for Arabic text recognition research. In WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering (No. 8). World Scientific and Engineering Academy and Society.