From Script to Digital: A Deep Learning Approach to Arabic Handwriting Recognition

Hamza Ahmed Abushahla*
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
b00090279@aus.edu

Ariel Justine Navarro Panopio*
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
b00088568@aus.edu

Layth Al-Khairulla*
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
b00087225@aus.edu

Alex Aklson†
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
aaklson@aus.edu

***Abstract*—Handwritten Text Recognition (HTR) for Arabic script is crucial for enabling digital accessibility and automating the conversion of handwritten documents into searchable digital formats. The cursive nature of Arabic script, with its positional letter shapes and diacritical marks, presents significant challenges that require specialized recognition systems. These challenges are compounded by the variability in handwriting styles and the limitations of techniques developed for other languages.**

In this study, we leverage the KHATT dataset to develop an end-to-end deep learning-based HTR system. Our approach effectively addresses the complexities of Arabic cursive handwriting using a segmentation-based model. To further enhance recognition accuracy, we incorporated KenLM for post-processing. Our baseline system, without KenLM, achieved a mean Character Error Rate (CER) of 1.894 and a Word Error Rate (WER) of 2.543. By integrating KenLM with n-gram models ranging from 2-gram to 4-gram, we observed significant improvements, attaining a CER of 0.903 and a WER of 0.998. These results demonstrate the effectiveness of KenLM in refining HTR outputs. This work underscores the potential of deep learning in advancing Arabic HTR, enabling the digitization of both contemporary and historical texts, and supporting broader applications in cultural preservation and digital workflows.

***Keywords*—Arabic Handwriting Recognition, KHATT Dataset, Arabic Handwriting, Optical Character Recognition, Deep Learning, Cursive Text Recognition**

I. Introduction

The Arabic language is the official language of 24 sovereign countries and is spoken by over 400 million people worldwide [1]. It is also one of the six official working languages of the United Nations (UN), reflecting its international importance. Beyond its widespread use, Arabic holds profound cultural, literary, and religious significance, serving as a cornerstone of the heritage of Arab and Muslim communities globally [2]. This extensive reach emphasizes the importance of developing accurate Arabic handwritten text recognition (HTR) systems, which have diverse applications in educational, governmental, and cultural preservation contexts [3]. For instance, the ability to accurately convert both historical and contemporary handwritten Arabic documents into digital text is vital for the digitization of archives, streamlining administrative processes, and supporting large-scale data analysis efforts.

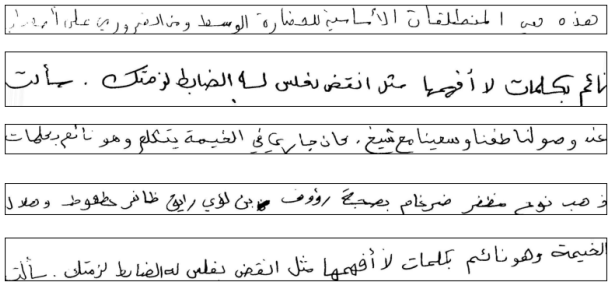

Figure 1. Sample line images from the KHATT dataset [4].

In particular, the automated recognition and digitization of Arabic cursive handwriting present unique challenges due to the inherent complexities of the script. For one, Arabic is written from right to left, with letters assuming different shapes based on their positional context within words. In addition, diacritical marks, known as "حركات" (harakat), and other special symbols such as the "همزة" (hamzah) further complicate the digitization process, as many letters share identical base shapes but are distinguished by one or more dots placed above or below the character [5]. These characteristics, combined with the variability in individual handwriting styles, make the task of accurately recognizing and digitizing Arabic handwriting particularly demanding. Furthermore, because of these unique challenges, techniques developed for recognizing handwriting in other languages, such as Latin-based scripts, cannot be directly applied to Arabic, necessitating tailored approaches and specialized models [6].

*These authors contributed equally to this work.
†Author to whom correspondence should be addressed.

Traditional Arabic HTR systems have predominantly relied on shallow learning techniques [3]. These methods typically involve handcrafted feature extraction processes that are sensitive to noise and degradation, limiting their effectiveness and scalability [7]. Furthermore, most existing systems adopt a segmentation-based approach, requiring the segmentation of words into individual characters before recognition. This segmentation process is particularly challenging for Arabic due to the cursive connections between letters and the presence of similar character shapes, making it difficult to accurately isolate each character [8], [9]. In contrast, segmentation-free models (holistic approaches) recognize words as whole-word images without any segmentation processes, which can be more effective for specific applications with limited vocabularies [10], [11].

Over the past decade, significant advancements in deep learning have revolutionized the field of HTR. Convolutional Neural Networks (CNNs) have emerged as the foundation of modern HTR systems due to their ability to extract complex spatial features, making them particularly effective for cursive scripts like Arabic [6], [12], [13]. Combined with Bidirectional Long Short-Term Memory (BiLSTM) networks and Connectionist Temporal Classification (CTC), deep learning models can effectively handle sequence modeling for continuous text recognition [3], [12], [14], [15]. More recently, Transformers and attention-based models have introduced new paradigms [16]–[18], enabling HTR systems to focus on specific regions of an image and capture long-range dependencies. These advances have significantly improved accuracy and generalizability, particularly for complex and diverse handwriting styles. Additionally, the integration of language models during post-processing has further improved prediction accuracy by leveraging contextual language information [3].

To advance the development of robust HTR systems for Arabic, leveraging comprehensive and diverse datasets is paramount. While historical datasets like Muharaf [1] have been valuable for analyzing historical manuscripts, there is a growing need to focus on contemporary datasets to achieve broader applicability. One such benchmark dataset is the KHATT [4], [19] dataset, which was developed by King Fahd University of Petroleum and Minerals (KFUPM) in Saudi Arabia. KHATT, an acronym for **K**FUPM **H**andwritten **A**rabic Tex**T**, also derives its name from the Arabic word (خط), meaning 'handwriting.' This dataset consists of 4,000 handwritten paragraphs contributed by 1,000 writers, with each individual providing six paragraphs. The contributors reflect diverse demographics, including participants from countries such as Saudi Arabia, Morocco, the USA, Palestine, Kuwait, Egypt, Tunisia, and Yemen, representing a wide range of age groups, educational backgrounds, and both genders.

KHATT includes scanned images of handwritten text along with corresponding ground truth annotations, making it a vital resource for training and evaluating handwriting recognition systems. Its diversity in writing styles and demographics makes it especially suitable for developing generalized HTR systems that can handle the complexities of Arabic handwriting across various contexts.

This study presents a segmentation-based HTR model utilizing advanced deep learning techniques to achieve high accuracy in Arabic handwriting recognition. The proposed system employs ResNet as a backbone for feature extraction, followed by a BiLSTM-CTC architecture for sequence modeling. To further enhance performance, a language model is incorporated during the post-processing phase, improving prediction accuracy by leveraging contextual information. In general, our contributions can be summarized as follows:

- We develop a CNN-based deep learning approach combined with BiLSTM-CTC and a language model to accurately recognize Arabic handwritten text.
- We incorporate KenLM for post-processing, significantly improving the CER and WER of our recognition system.
- We evaluate our system using the KHATT dataset, demonstrating its effectiveness across diverse handwriting styles.

The remainder of this paper is organized as follows: Section 2 provides the background, including an overview of HTR, the unique characteristics of Arabic handwriting, and essential terminology. Section 3 reviews related work, covering traditional methods, advancements in deep learning, and the importance of datasets in Arabic handwriting recognition. Section 4 outlines our methodology, detailing the proposed solution, algorithms, and techniques used to address the problem. Section 5 presents the results and evaluation, discussing the findings and their implications. Finally, Section 6 concludes the study by summarizing key contributions, addressing limitations, and proposing directions for future research.

II. Background

A. Handwritten Text Recognition

1) Introduction to OCR and the Transition to HTR

Optical Character Recognition (OCR) revolutionized text recognition by enabling the automated extraction of text from printed documents, laying the groundwork for modern systems. Early OCR systems relied heavily on accurate text localization and segmentation to isolate individual characters or lines, ensuring precise feature extraction [20], [21]. These systems employed both micro-level features, such as character-specific edges and textures, and macro-level features, capturing broader document patterns like text alignment and spacing. Techniques such as edge-based detection (e.g., Sobel or Canny filters) [22] and statistical texture analysis quantified local properties, while keypoint descriptors like Scale-Invariant Feature Transform (SIFT) [23] enhanced robustness by identifying invariant features under varying scales, rotations, or distortions.

Despite their success in recognizing machine-printed text, OCR systems faced significant challenges when applied to handwritten text. Unlike printed text, handwriting exhibits high variability in style, uneven spacing, and cursive connections, which often overlap and lack clear boundaries [20], [24]. These complexities demanded more sophisticated approaches that

could capture the nuanced and dynamic nature of handwriting, moving beyond the rigid assumptions of traditional OCR.

The 1990s marked a pivotal era in text recognition, driven by the rise of neural network models [7]. Innovations like LeNet [25] pioneered the use of convolutional layers, allowing systems to automatically learn hierarchical features directly from raw input images. This approach eliminated the reliance on handcrafted features, enabling more robust handling of variations in font styles, sizes, and distortions. With advancements in computational power, neural networks laid the groundwork for extending OCR capabilities to the more complex domain of HTR.

2) Levels of HTR

HTR systems tackle text recognition at different levels of granularity, beginning with character-level recognition. This approach involves recognizing isolated characters, often requiring segmentation of the text. While effective for non-cursive scripts or logographic languages like Japanese [26] and Chinese [27], character-level recognition struggles with cursive languages like Arabic, where the shapes of letters depend on their position and surrounding context.

To overcome these limitations, word-level recognition transcribes individual handwritten words as single units, reducing the dependency on character segmentation and improving accuracy for cursive scripts [28]. More advanced techniques have focused on line-level recognition, where entire lines of text, including spaces, are transcribed. Line-level HTR can either rely on pre-segmented input or integrate segmentation and recognition into a unified framework. Line-level recognition introduces challenges related to spatial and contextual coherence, as the relationships between words and characters must be preserved. However, it offers a balance between complexity and efficiency, making it particularly relevant for cursive scripts like Arabic.

In recent developments, systems also tackle paragraph- and page-level recognition, incorporating layout analysis for handling complex document structures. These approaches combine transcription with spatial segmentation tasks, allowing models to process structured documents and extract meaningful text from diverse layouts [28], [29]. As the levels progress from character to page, the complexity increases, requiring more robust handling of spatial and contextual relationships.

3) Sequential Nature of Handwriting

Handwriting is inherently sequential, with characters and words influenced by their position and surrounding context within a sequence. Recognizing handwriting requires models capable of capturing these dependencies to ensure accurate transcription. Rcurrent neural networks (RNNs), introduced in [30], are particularly well-suited for this task. Their feedback loops enable information to persist across time steps, allowing them to model sequential dependencies effectively. This capability makes RNNs a natural choice for handwriting recognition, where the relationship between characters is critical.

Despite their advantages, standard RNNs face significant challenges in handling long sequences due to the vanishing gradient problem [31]. During training, the backpropagation algorithm calculates gradients to adjust network weights. In RNNs, gradients are propagated through time steps by repeated multiplication. Over long sequences, these gradients tend to shrink exponentially, often becoming too small to update the weights effectively. This issue is exacerbated by activation functions such as sigmoid or tanh, which compress their outputs into a narrow range, further diminishing the gradients.

For that, long short-term memory (LSTM) networks, introduced in [32], were specifically designed to overcome the vanishing gradient problem. LSTMs introduce memory cells and gating mechanisms (input, forget, and output gates) that regulate the flow of information through the network. These mechanisms enable LSTMs to selectively retain or discard information, allowing them to maintain long-term dependencies. For example, in handwriting recognition, an LSTM can retain information about a diacritical mark, like "ضمّة" (damma), at the beginning of a word while processing subsequent characters, ensuring accurate transcription within context.

Bidirectional LSTMs (BiLSTMs), introduced in [33], extend the LSTM architecture by processing sequences in both forward and backward directions. This bidirectional processing allows the model to consider both preceding and succeeding context, making it particularly effective for cursive handwriting, where the interpretation of a character often depends on its neighbors.

For tasks involving 2D input data, such as handwritten line images, Multidimensional LSTM (MDLSTM) networks, introduced in [34], further enhance sequential modeling by introducing recurrence along two spatial dimensions. This architecture captures dependencies in both horizontal and vertical directions, making it ideal for processing handwritten text. However, the computational demands of MDLSTMs have led to a shift toward hybrid models that combine Convolutional Neural Networks (CNNs) for spatial feature extraction with BiLSTMs for sequence modeling. These hybrid architectures balance accuracy and efficiency, forming the backbone of modern handwriting recognition systems.

4) CTC Loss and Decoding

A key component in modern HTR systems is the use of Connectionist Temporal Classification (CTC) as a loss function and decoding mechanism. Introduced by Graves et al. [35], CTC enables sequence-to-sequence learning when the input (e.g., image features) and output (e.g., character sequences) lengths differ. It allows models to align variable-length inputs and outputs without requiring pre-segmented data, a significant challenge in HTR.

CTC introduces a special "blank" token ("−") to handle gaps in alignment, letting the model represent transitions between characters or accommodate uneven spacing and cursive connections. This design is ideal for HTR, where the input sequence often exceeds the target transcription in length and characters may not align neatly to time steps.

CTC computes the conditional probability of a target transcription l given the model's output Y, which consists of T time steps. Each step y_t is a probability distribution over all possible characters, including the blank token. The conditional probability $p(l|Y)$ is defined in Equation 1 as:

$$p(l|Y) = \sum_{\pi:M(\pi)=l} p(\pi|Y), \tag{1}$$

where π represents all possible alignments of Y that map to l, and $M(\pi)$ removes repeated characters and blanks from π. For example, "−−لل−−اااا−−ث−مم" gets aligned to the target "مثال". The sequence output is the probability of π at time step t, which is defined in Equation 2 as:

$$p(\pi|Y) = \prod_{t=1}^{T} y_{\pi_t}^t, \tag{2}$$

where $y_{\pi_t}^t$ is the output probability of the character π_t at time step t, as described in [36]. This formula accounts for the cumulative probability of the alignment π across all time steps.

The CTC loss function, shown in Equation 3, sums over all possible alignments π that produce the target sequence l. It is efficiently computed using a dynamic programming approach, which calculates the forward and backward paths:

$$\mathcal{L}_{CTC} = -\log \sum_{\pi:M(\pi)=l} \prod_{t=1}^{T} y_{\pi_t}^t. \tag{3}$$

Furthermore, CTC decoding converts the probabilistic output Y into a transcription through two main strategies: (1) greedy decoding and (2) beam search decoding. On the one hand, greedy decoding selects the character with the highest probability at each time step. While fast and computationally efficient [35], it often ignores global dependencies, leading to suboptimal results in complex handwriting or ambiguous cases. On the other hand, beam search decoding improves accuracy by maintaining multiple candidate transcriptions (the "beam") and evaluating their overall probabilities. This method captures context more effectively and is particularly advantageous for cursive or highly variable handwriting, as highlighted in [36], [37].

CTC transforms HTR by allowing models to learn temporal alignments directly from data, avoiding the need for explicit segmentation.

5) Deep Learning Advancements in HTR

Recent advancements in deep learning, particularly attention-based models and transformers, have revolutionized HTR. Unlike RNNs, which process sequences step by step, transformers analyze an entire input sequence simultaneously, capturing long-range dependencies while dynamically focusing on specific regions of the input image. This architecture, introduced in [38], has fundamentally changed how sequential data is processed in deep learning.

A helpful analogy is how humans read text. Instead of scanning every letter in a linear fashion, we focus more on unclear or important areas while using surrounding context to fill in gaps. Similarly, transformers use attention mechanisms to emphasize ambiguous or complex sections of handwriting while retaining global context. For example, if a character in the word "مثال" (example) is unclear, the transformer can reference the rest of the word to infer its correct identity.

This dynamic attention mechanism enables modern HTR systems to excel with diverse handwriting styles, from cursive to decorative scripts, by adapting to stylistic variations and spatial irregularities. Transformers' ability to process sequences holistically has set new benchmarks in accuracy and generalizability, making them integral to state-of-the-art HTR systems.

B. Characteristics of Arabic Handwriting

Arabic handwriting possesses several distinctive features that present unique challenges for Handwritten Text Recognition (HTR) systems. A comprehensive understanding of these characteristics is essential for developing effective recognition models tailored to the intricacies of the Arabic script. This section delineates the primary attributes of Arabic handwriting that influence the design and performance of HTR systems.

1) Character Level Characteristics

The Arabic alphabet comprises 28 primary characters, each of which does not have distinct uppercase or lowercase forms. Instead, each character can adopt two to four different shapes depending on its position within a word: isolated, initial, medial, or final [6]. Consequently, the 28 Arabic letters are represented by 84 distinct forms. Table I illustrates the Arabic letters in their four positional forms, detailing the letter order in the Arabic alphabet, pronunciation, and the various shapes each letter assumes based on its placement within a word. Notably, separating characters, such as "ا" (Alif) and "ر" (Raa), have only two forms: isolated and final.

2) Diacritical Marks (Dots) Dots are integral components of Arabic letters, playing a crucial role in distinguishing characters that share the same primary strokes. Arabic utilizes 16 main strokes, and the presence, absence, or placement of dots can differentiate between letters with identical base shapes. For instance, the letters "ب" (Ba), "ت" (Ta), and "ث" (Tha) are differentiated solely by the number and position of dots below them. Accurate recognition of these diacritical marks is essential to prevent misclassification and ensure the correct interpretation of words. Table II shows the 16 main strokes of the Arabic script, and able III displays the letters categorized by one, two, or three dots.

3) Hamzah: A Special Character

The hamzah "ء" is a unique character in the Arabic script that can function as a primary character or as a subcomponent integrated within other primary characters [39]. For example, the hamzah appears as a standalone character in words like

Table I
ARABIC CHARACTERS IN THE 4 FORMS: ISOLATED, BEGINNING, MIDDLE, AND END

No.	Pronunciation	Isolated	Beginning	Middle	End	No.	Pronunciation	Isolated	Beginning	Middle	End
1	Alef	ا	ا	ا	ـا	15	Dhad	ض	ضـ	ـضـ	ـض
2	Baa	ب	بـ	ـبـ	ـب	16	Taa	ط	طـ	ـطـ	ـط
3	Taa	ت	تـ	ـتـ	ـت	17	Thaa	ظ	ظـ	ـظـ	ـظ
4	Thaa	ث	ثـ	ـثـ	ـث	18	Ain	ع	عـ	ـعـ	ـع
5	Jeem	ج	جـ	ـجـ	ـج	19	Ghain	غ	غـ	ـغـ	ـغ
6	Haa	ح	حـ	ـحـ	ـح	20	Faa	ف	فـ	ـفـ	ـف
7	Khaa	خ	خـ	ـخـ	ـخ	21	Qaf	ق	قـ	ـقـ	ـق
8	Dal	د	د	د	ـد	22	Kaf	ك	كـ	ـكـ	ـك
9	Thal	ذ	ذ	ذ	ـذ	23	Lam	ل	لـ	ـلـ	ـل
10	Raa	ر	ر	ر	ـر	24	Meem	م	مـ	ـمـ	ـم
11	Zai	ز	ز	ز	ـز	25	Noon	ن	نـ	ـنـ	ـن
12	Seen	س	سـ	ـسـ	ـس	26	Ha	ه	هـ	ـهـ	ـه
13	Sheen	ش	شـ	ـشـ	ـش	27	Waw	و	و	و	ـو
14	Sad	ص	صـ	ـصـ	ـص	28	Yaa	ي	يـ	ـيـ	ـي

Table II
MAIN ARABIC CHARACTERS STROKES

Arabic Main Stroke
ا ب ج د ر س ص ط ع ف ك ل م ن و ي

Table III
ARABIC CHARACTERS THAT HAVE ONE, TWO, OR THREE DOTS

No. of Dots	Letters
One dot	ب ج خ ذ ز ض ظ غ ف ن
Two dots	ت ق ي
Three dots	ث ش

"سماء" (samā', meaning sky) and "هواء" (hawā', meaning air). Additionally, the hamzah can be incorporated within other letters, such as "ك", "أ", "ئ", "ؤ", and "إ". Moreover, the placement of the hamzah, whether above, below, or within a letter, determines its pronunciation and role within different word contexts.

4) Diacritics: Tashkeel and Tanween

"التشكيل" (tashkeel), also known as "حركات" (harakat), and "التنوين" (tanween) are diacritical marks used in Arabic to indicate pronunciation and grammatical nuances [40]. Tashkeel consists of five symbols that modify the meaning and pronunciation of words. For example, the words "عِلْم" (ilm, meaning knowledge) and "عَلَم" (alam, meaning flag) share the same base characters but differ in their Tashkeel, altering their meanings significantly. Although Tashkeel is not extensively used in everyday Arabic writing, since context often clarifies meaning, it is essential for novice readers and in educational materials. Tashkeel can change the semantic interpretation of words, making its accurate recognition (if present) vital for preserving textual integrity.

5) Cursive Nature and Baseline Structure Arabic is inherently a cursive script in both its printed and handwritten forms, meaning that letters within words are connected by a horizontal line known as the baseline [6]. Portions of characters that extend below the baseline are referred to as "descenders," while those that extend above are termed "ascenders." Figure 2 depicts the baseline with ascenders and descenders, highlighting how these elements contribute to the overall structure and flow of Arabic handwriting. This cursive connectivity facilitates fluid writing but complicates the recognition process, as it introduces variability in letter shapes and spacing.

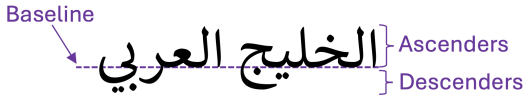

Figure 2. Illustration of the baseline, ascenders, and descenders in the Arabic phrase "الخليج العربي" (The Arabian Gulf).

6) Separating Characters and Word Segmentation

Arabic includes six separating characters namely "ا", "ر", "ز", "د", "ذ" and "و", which are not connected to the characters that follow them [6]. These separating characters divide words into smaller units known as Parts of Arabic Words (PAWs). Each word may consist of one or more PAWs depending on the presence of separating characters. Accurate segmentation of PAWs is critical for effective HTR, as it influences the ability to isolate and recognize individual characters within connected scripts. Examples of words composed of one, two, three, four, or five PAWs are presented in Table IV.

Table IV
EXAMPLES OF WORDS WITH ONE, TWO, THREE, FOUR, AND FIVE PAWS

No. of PAWs	Example	Division (Right-to-Left)
One PAW	ليث	ليث
Two PAWs	حمزة	حمز + ة
Three PAWs	إبراهيم	إ + برا + هيم
Four PAWs	نورة	نو + رة
Five PAWs	أوراد	أ + و + را + د

7) Overlapping in Handwritten PAWs In handwritten Arabic, PAWs may exhibit vertical overlapping, where characters within a word or between adjacent words overlap vertically. Figure 3 illustrates examples of such overlapping, demonstrating how characters can intertwine, making segmentation and recognition more challenging. This overlapping is a common occurrence in cursive handwriting and requires robust HTR systems capable of accurately distinguishing and recognizing characters despite such distortions.

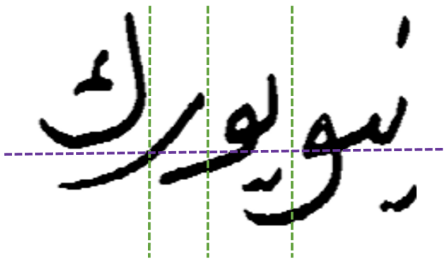

Figure 3. Vertical overlapping between two PAWs occurring twice within the same handwritten word.

C. Challenges in Arabic HTR

The recognition of Arabic handwritten text poses unique challenges due to the intricacies of the script, the variability of its handwritten forms, and the limitations of current resources and methodologies. This section explores these challenges in depth, highlighting the inherent difficulties that hinder the development of robust Arabic HTR systems.

1) High Variability in Handwriting Styles

Arabic handwriting exhibits a significant degree of variability, making it one of the most challenging scripts to process [41]. Unlike many other scripts, Arabic handwriting allows for a broad range of stylistic diversity that is still considered acceptable. Individual differences in stroke thickness, letter spacing, slant, and elongation further contribute to the variation in handwritten forms. These stylistic differences are not arbitrary; they are often influenced by cultural and regional preferences. For example, traditional calligraphic styles such as "النسخ" (Naskh), "الرقعة" (Ruq'ah), and "الكوفي" (Kufi) have distinct proportions and connections, which can manifest even in casual handwriting. Figure 4 illustrates examples of handwritten Arabic, showcasing the variability across different writers and styles.

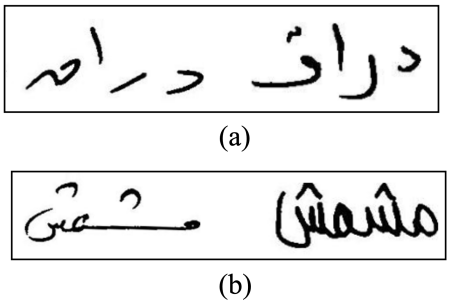

Figure 4. Handwritten Arabic words from the KHATT dataset: (a) دراق (Duraq, 'plum') and (b) مشمش (Mishmish, 'apricot'), showcasing variability in letters like ش, م, د, ر, and ق across writers.

Additionally, Arabic handwriting is not restricted to a single standardized form. Variants of Modern Standard Arabic (MSA) often appear in handwritten texts, including informal adaptations and regional orthographic practices. Although HTR systems are not tasked with interpreting dialects or regional meanings, the visual form of these handwritten variants, such as alternative spellings or elongated structures, introduces challenges in character recognition. This vast variability complicates segmentation and encoding, where traditional OCR systems fail due to their reliance on rigid, rule-based approaches. Unlike scripts with default spaces between characters, Arabic's cursive nature creates continuous connections between letters, making segmentation into discrete units a formidable task.

2) Right-to-Left Directionality and Structural Bias

The right-to-left (RTL) directionality of Arabic script presents significant challenges for HTR systems, which are predominantly designed for left-to-right (LTR) languages [42]. This fundamental difference impacts multiple aspects of HTR, including preprocessing, feature extraction, and sequence modeling, necessitating specific adaptations to ensure effective processing of Arabic text.

Arabic text flows from the top-right to the left, a structure that conflicts with the design assumptions underlying most LTR-oriented algorithms. This discrepancy is especially evident in preprocessing tasks, such as baseline detection and text alignment. In Arabic handwriting, the baseline is often influenced by interactions between ascenders (e.g., "ط") and descenders (e.g., "ي"), which are more prominent in cursive styles. Traditional baseline detection methods, typically designed for LTR languages, often fail to account for these interactions, leading to errors in segmentation and recognition [43]. Such misalignments not only distort spatial positioning but also propagate errors through subsequent processing stages.

Convolutional Neural Networks (CNNs), widely used in modern HTR systems, exacerbate these challenges due to their

sliding window mechanism. CNNs extract features by moving a kernel across an image from left to right [25]. While this design is effective for LTR scripts, it introduces a structural bias when applied to Arabic handwriting. In this case, the CNN begins processing at the visual end of a word and progresses toward its beginning, disrupting the natural contextual flow. This misalignment particularly affects connected letters and diacritics, both of which are critical for accurate recognition of Arabic text.

One common workaround is to horizontally flip the input image, effectively reversing the text flow to make it appear left-to-right [1], [44]. While this approach enables LTR-designed systems to process Arabic text more effectively, it introduces its own set of problems. Flipping the text can distort spatial relationships, particularly for overlapping characters or diacritics, and fails to address the inherent architectural bias in the CNN. Although flipped input aligns the image for feature extraction, it does not resolve sequence alignment issues during decoding, where the reversed order can still affect performance.

Sequence modeling architectures, such as RNNs and transformers, encounter additional challenges with RTL directionality. RNNs process sequences step-by-step, making them sensitive to the order of input data. When trained on LTR sequences, an RNN processes Arabic text in reverse, beginning at the end of a word or sentence and losing the natural semantic flow required for contextual understanding [44]. Although transformers are more flexible due to their self-attention mechanisms, they face similar problems. Positional encodings in pretrained transformer models are typically designed for LTR scripts. Applying these encodings to Arabic text misaligns attention distributions, reducing the model's ability to accurately focus on the initial characters in a sequence.

III. Related Work

In this section, we review previous research on Arabic HTR, focusing on the development and utilization of Arabic handwriting datasets, preprocessing techniques, and advancements in deep learning methodologies that have significantly advanced the field.

A. Datasets of Arabic Handwriting

Several datasets have been developed to facilitate Arabic handwriting research, each varying in focus, size, and availability. Below, we summarize the most relevant datasets, including KHATT, IFN/ENIT, MADCAT, AHDB, and Muharaf.

- **KHATT Dataset [4], [19]:**
  The KHATT dataset, used in this study, is a modern Arabic handwriting dataset comprising 4,000 paragraphs written by 1,000 scribes, with each scribe contributing four paragraphs. Developed jointly by researchers from KFUPM in Saudi Arabia, Technische Universität Dortmund (TU Dortmund), and Technische Universität Braunschweig (TU Braunschweig) in Germany, KHATT provides a comprehensive resource for training and evaluating HTR systems. The dataset includes 1,000 forms, each containing four pages: the first page holds writer information, the second and third pages feature fixed and free text paragraphs respectively, and the fourth page contains open and free text written on a ruled baseline to minimize skew. This structure ensures consistency and facilitates the extraction of 2,000 randomly selected and fixed paragraphs from 46 different sources.
- **IFN/ENIT Dataset [45]:**
  The IFN/ENIT dataset is one of the most widely used and cited Arabic handwritten databases. Developed by the Institute for Communications Technology (IFN) and Ecole Nationale d'Ingénieur de Tunis (ENIT) in Tunisia, this dataset consists of handwritten Tunisian town and village names. It includes 2,265 forms filled out by 411 different writers, covering 946 Tunisian town and village names along with their postcodes. Each form is scanned, and the handwritten words are automatically extracted and annotated with ground truth data, which is then manually verified. The dataset encompasses 26,459 handwritten words and 115,585 PAWs, making it a valuable resource for evaluating HTR systems' performance in recognizing geographically specific Arabic text.
- **MADCAT Dataset [46]:**
  The MADCAT (Multilingual Automatic Document Classification, Analysis, and Translation) dataset is part of a five-year technology evaluation program sponsored by the US Defense Advanced Research Projects Agency (DARPA). Created by the Linguistic Data Consortium (LDC), MADCAT focuses on machine translation and includes a training set of Arabic handwritten text. The dataset comprises 9,693 handwritten pages, each accompanied by scanned images, tokenized and sentence-segmented original Arabic source documents, ground truth annotations, and English translations. The data sources include newsgroups, newswire, and weblogs. However, MADCAT is not freely available to the public, restricting its use to members and participants of the DARPA program.
- **AHDB Dataset [47]:**
  The Arabic Handwritten Database (AHDB) consists of Arabic handwritten words, free handwritten sentences, and pages. The dataset includes the top 20 most common Arabic words, collected through a six-page form containing 96 instances of these words and their literal amounts used in cheques. Each image is annotated with the corresponding word and writer identification numbers. AHDB provides a diverse set of handwriting samples, aiding in the training of deep learning-based HTR models capable of handling both isolated words and connected text.
- **Muharaf Dataset [1]:**
  The Muharaf dataset is the largest publicly available Arabic dataset with fully annotated and transcribed historical manuscripts at the text-line level. It encompasses 1,644 pages (1,216 public and 428 restricted), spanning from the early 19th to the early 21st century, and includes

36,311 text lines (24,495 public). Line-level images are stored in PNG format and were generated using line warping software to create consistent horizontal grids. Extensive metadata, including ground truth annotations, is provided in JSON format. This dataset is instrumental for training and evaluating HTR systems aimed at digitizing and preserving historical Arabic literature, addressing challenges related to noise, degradation, and archaic handwriting styles.

B. Preprocessing Techniques

Preprocessing techniques are a crucial pre-requisite to any model's input. It ensures that the model is able to focus on the important features of the input to enable it to better detect patterns. In our case, our input is images that are handwritten Arabic documents and so it is crucial to ensure that the models are receiving input that can allow it to identify handwriting, letter styles, and strokes.

A means of doing this is binarization. Binarization is a crucial preprocessing step that transfroms input images, whether they are RGB or grayscale, into a simplified binary format. The purpose of this is to highlight character strokes and remove unneccessary image color information. There are different ways to apply thresholds for binarization, the most prominent being global thresholding where beyond a certain threshold, the pixel is set to 1 and below that, set to 0. A more robust way is Otsu's method [48] that finds an optimal global threshold by minimizing intra-class variance. It has been also proven to exhibit lower running times at a mean of 2 seconds [49]. Another approach is differentiable binarization that integrates the binarization process into network training, allowing it to be part of optimization and loss calculations [50]. Differentiable binarization uses an adaptive threshold map learned from the network to apply different thresholds on different parts of the image which is useful in areas of low confidence in boundary regions compared to areas of high confidence centrally within the image. Furthermore, this layer can be removed once the model is trained to ensure that inference time is not impacted.

Another measure of preprocessing is noise removal. Noise removal is a crucial step as images often do not come in ideal conditions leading to poor model performance if fed directly. One of the techniques of noise removal is median filtering. The median of a group of pixels (3x3, 5x5, 7x7, etc.) that surround a specific pixel (its neighborhood) replaces the pixel to smooth out and reduce noise within the image [49], [51]. Another common category of techniques is morphological operations. These operations manipulate the shape of the input images. These operations include dilation, erosion, opening, and closing [51]. Dilation adds pixels to the boundaries of objects in an image making them thicker within the image. Conversely, erosion removes pixels from object boundaries, eliminating pixels that are isolated. Opening and closing are then combinations of erosion and dilation that help find the right balance between refining object shapes and eliminating noise.

C. Deep Learning Approaches for HTR

Many HTR systems have been developed to recognize Arabic handwritten characters and words. CNNs have been widely employed in handwritten recognition due to their capacity to autonomously acquire hierarchical features from raw pixel inputs [6].

There have been multiple variations of CNNs developed to tackle a number of datasets pertaining to Arabic handwritten text. An example of this is a simple CNN trained on the Arabic Handwritten Characters Dataset (AHCD) that used two convolutional, two pooling, and two fully connected layers [52]. This architecture was able to gain an accuracy of 94.9% which is overall good but might be attributed to the dataset and its characteristics. Similar models were run on different datasets such as Hijja [13] that got a lower accuracy of 88% but improved results on AHCD (97%). The addition of batch normalization and dropout layers only slightly improved performance on AHCD getting an accuracy of 97.6% [53]. Adding more convolutional layers (four) only increases the initial accuracy to 97.2% [54].

Another common approach involves combining CNNs with other classifiers, such as Support Vector Machines (SVMs), to enhance recognition accuracy. For instance, Elleuch et al. [55] utilized a combination of CNN and SVM with dropout regularization to recognize offline Arabic handwritten characters, achieving error classification rates of 5.83% on the Handwritten Arabic Characters Database (HACDB) and 7.05% on the IFN/ENIT dataset.

RNNs, particularly BiLSTM networks, have also been integrated with CNNs to capture sequential dependencies in handwriting. For example, Jemni et al [56] proposed an Arabic HTR based on multiple BiLSTM-CTC combinations. They employed two feature extraction techniques: (1) segment-based features and (2) Distribution-Concavity (DC)-based features. These features were trained on different levels of the BiLSTM-CTC combinations, including low-level fusion, mid-level combination methods, and high-level fusion. Experiments on the KHATT dataset demonstrated that high-level fusion yielded a WER of 29.13% and a CER of 16.27%.

Sliding window techniques have also been explored for feature extraction in Arabic HTR. Stahlberg and Vogel [57] proposed a recognition system using a sliding window approach with a window width of 3 pixels and an overlap of 2 pixels. They extracted pixel-based features from raw grayscale values and segment-based features such as centroid and height. Utilizing the Kaldi toolkit for classification, their method achieved a WER of 30.5% on the HACDB dataset.

Moreover, BenZeghiba et al. [58] developed a hybrid Hidden Markov Model (HMM) and Artificial Neural Network (ANN) framework to recognize Arabic handwritten text. Their system utilized MDLSTMs to extract pixel values by scanning text line images in four directions. During training, CTC was used, and the Viterbi algorithm served as the decoding strategy to generate the best hypothesis of a character sequence. Incorporating a hybrid language model that included words

and PAWs, the system achieved a WER of 33% on the KHATT dataset.

Data augmentation and normalization techniques have also been explored to enhance the robustness of HTR systems. Wigington et al. [59] introduced a novel profile normalization strategy for both word and line images, alongside augmenting existing text images using random perturbations on a regular grid. These techniques, combined with a CNN-LSTM architecture, improved handwriting text recognition, particularly for datasets with diverse handwriting styles. Recognizing the distinct characteristics of children's handwriting, Altwaijry et al. [13] trained their models on children's handwriting samples to address variations unique to younger writers.

Recent advancements have focused on leveraging various CNN architectures to tackle the specific challenges of Arabic script, such as cursive writing and diacritical markings. Khayati et al. [60] emphasized the development and efficiency of several CNN architectures in addressing these challenges. Additionally, Berriche et al. [61] developed a CNN-graph theory method for segmenting linked and overlapping cursive Arabic letters, significantly improving character identification accuracy. In another study, AlShehri et al. [62] introduced DeepAHR, a deep neural network designed for Arabic handwritten recognition, which demonstrated enhanced feature extraction and recognition capabilities, outperforming previous models in both accuracy and processing speed.

Furthermore, pretrained CNN models such as VGG [63], ResNet [64], and Inception [65] have been adapted for Arabic HTR. For example, Alghyaline et al. [66] evaluated these models on datasets like Hijja, the Arabic Handwritten Character Dataset (AHCD), and the AlexU Isolated Alphabet (AIA9K), achieving accuracies of 93.05%, 98.30%, and 96.88% respectively using the VGG model. Transformer-based architectures have also been explored, with Momeni et al. [67] employing transformer transducers and cross-attention mechanisms on the KHATT dataset, resulting in a CER of 18.45%.

More recently, a new dataset, Muharaf, was released by Saeed et al. [1], and they conducted page-level HTR on it using a combination of three CNN-based models: the Start of Line (SOL) Network, the Line Follower (LF) Network, and the Handwriting Recognition (HW) Network. The SOL Network is a CNN designed to detect the starting coordinates of each text line in a page image. The LF Network, another CNN, tracks the trajectory of text lines by iteratively predicting the next coordinates and orientation. The HW Network is a CNN–BiLSTM model trained with CTC loss for end-to-end text recognition, which was also employed for line-level HTR. Using the Muharaf dataset, their system achieved a CER of 0.157 and a WER of 0.398. Additionally, they tested their architecture on other datasets, obtaining a CER of 14.1% on KHATT and 5.5% on MADCAT.

Furthermore, Chan et al. [44] developed a transformer-based encoder-decoder model called HATFormer for line-level HTR. HATFormer utilizes Vision Transformers (ViT) [68]to extract visual features and employs RoBERTa (Robustly Optimized BERT Approach), a pretrained language model, for accurate text decoding. They also introduced a BlockProcessor to preserve image aspect ratios and reduce horizontal information loss. Evaluated primarily on the Muharaf dataset, HATFormer achieved a CER of 11.7%, outperforming traditional CNN-RNN models in recognizing Arabic handwriting. However, this advancement comes with a trade-off of higher computational costs due to the complexity of transformer architectures. HATFormer was also tested on the KHATT and MADCAT datasets, achieving CERs of 14.1% and 4.2%, respectively.

In addition to the standard CNN-BiLSTM-CTC models, Mutawa et al. [3] proposed an enhanced HTR system that incorporates a post-processing step using KenLM, a statistical language model trained on Arabic text. Their architecture follows the traditional CNN-BiLSTM-CTC framework, utilizing a 32-layer ResNet for feature extraction followed by three BiLSTM layers with CTC loss for sequence prediction. The integration of KenLM refines the model's predictions by incorporating contextual word probabilities, thereby improving recognition accuracy. This approach achieved state-of-the-art performance, benchmarking on the KHATT and AHTID/MW datasets with a CER of 13.2% and a WER of 27.31% on KHATT, and a CER of 6.6% and a WER of 17.42% on AHTID/MW, surpassing the performance of transformer-based models like HATFormer. Table V below lists the recent works discussed in this section.

IV. Methodology

In this section, we detail the steps we took to reproduce the results of [3]. We discuss the data preparation, model architecture, training strategies, and evaluation metrics to assess our model for robustness and reproducibility.

A. Data Preparation and Preprocessing

To prepare the data for model training, we utilized the filtered line-by-line images of the KHATT dataset from Kaggle. The data has already been preprocessed by binarizing the images, cropping out the unnecessary whitespaces around the handwritten texts, and straightening out the lines as much as possible. Thus, the only preprocessing we had to do was adhering to the paper's [3] resizing of the images to 64 pixels for the height and 1048 for the width, normalizing the images, and horizontally flipping the images and the labels because a CNN traverses the image from left-to-right. Furthermore, we used an 80-10-10 train-validation-test split stratified on the length of the sentences because longer sentences may impact model's discriminative ability. Before this step, we had to aggregate sentences with less than eight characters to ensure an adequate number of samples of varying sentence lengths for each split. Lastly, we employed data augmentation through Keras' ImageDataGenerator to increase our dataset's size and improve the model's generalizability.

1) Model Architecture

As we aim to reproduce the results of [3], we reimplemented the architecture detailed in the paper to the best of our ability. This led to us creating the CNN-BiLSTM architecture, which

Table V
DEEP LEARNING METHODS FOR ARABIC HANDWRITTEN TEXT RECOGNITION

Reference	Year	Dataset	Architecture	Performance
Stahlberg and Vogel [57]	2015	HACDB IFN/ENIT	Sliding Window + Kaldi	WER: 30.5% WER: 11.5%
BenZeghiba et al. [58]	2015	KHATT Maurdor	HMM + MDLSTM	WER: 31.3% WER: 33.2%
Elleuch et al. [55]	2016	HACDB IFN/ENIT	CNN + SVM	ECR: 5.83% ECR: 7.05%
Jemni et al. [56]	2018	KHATT	CNN + BiLSTM + CTC	CER: 16.27% WER: 29.13%
Berriche et al. [61]	2019	IESK-ArDB	Segmentation Hypothesis Graph + CNN	Accuracy: 96.97%
Altwaijry et al. [13]	2021	Hijja AHCD	Segmentation + CNN	Accuracy: 88% Accuracy: 97%
Momeni et al. [67]	2023	KHATT	Transformer with Cross-attention	CER: 18.45%
AlShehri [62]	2024	Hijja AHCD	Segmentation + CNN	Accuracy: 88.24% Accuracy: 98.66%
Alghyaline [66]	2024	Hijja AHCD AIA9K	Pretrained VGG	Accuracy: 96.88% Accuracy: 98.30% Accuracy: 93.05%
Saeed et al. [1]	2024	Muharaf KHATT MADCAT	SOL + LF + HW Networks	CER: 14.9% CER: 14.1% CER: 5.5%
Mutawa et al. [3]	2024	KHATT AHTID/MW	CNN-BiLSTM-CTC + KenLM	CER: 13.2% WER: 27.31% CER: 6.6% WER: 17.42%
Chan et al. [44]	2024	Muharaf KHATT MADCAT	HATFormer (ViT + RoBERTa + BlockProcessor)	CER: 11.7% CER: 14.1% CER: 4.2%

is the most popular framework for the OCR task [1], [3], [12], [14], [15] for a number of reasons. Moving away from using handcrafted features like SIFT descriptors, the CNN extracts the hierarchical spatial features inherent in an image, especially in handwritten text images. As the CNN traverses the handwritten text from left-to-right, it learns, through its various filters, how to capture the shapes of diacritics, letters, and words in Arabic handwritten text. For example, learning the distinction between the Arabic letters "ب" (Ba) and "ت" (Ta) relies on the model knowing where to place the dots and how many.

Moreover, adding more and more layers can result in a better representation of the handwritten text. However, training deeper networks is not necessarily easy, primarily due to the vanishing gradient problem—where multiplying many small gradients together hinders the neural network's learning process. Thus, the introduction of skip connections or residual connections.

Skip connections allow a layer to bypass one or more subsequent layers by adding or concatenating its output as another input to a non-consecutive layer. That is, they allow the outputs of a layer to be fed to a much later layer either through element-wise addition or concatenation. This is the foundational idea of the ResNet architecture. With these skip connections, training deeper networks is easier, and the impact of vanishing gradients is reduced. We recreated the ResNet32 architecture, serving as the backbone and the feature extractor of our handwritten text images. The residual blocks within ResNet32 contain skip connections that perform element-wise addition and also batch normalization layers. The presence of batch normalization layers helps stabilize training and accelerate convergence by normalizing the feature maps. The only change we made to this ResNet32 architecture was adding a 1×1 convolutional layer in the residual block to handle the varying filter sizes for the skip connections. This ensures compatibility between the layers.

After adding a dropout layer after the ResNet block to lessen the chances of overfitting to the training set, the feature maps are fed into the three BiLSTMs. The three BiLSTMs address the sequential nature of handwriting and the necessity of considering the connections between each letter to form a word, which is especially important in cursive-styled text like Arabic. Furthermore, the bidirectional behavior of this particular LSTM is essential to the network learning how the preceding and subsequent characters relate to each other. For instance, the form of the letter "ع" (Ain) depends on its position within a word. This is how the BiLSTM layers handle these contextual variations, improving character recognition

accuracy.

Succeeding the dropout layer directly after all three BiLSTM layers is a fully connected dense layer, followed by the CTC as the loss function to help optimize how the model learns to align its predictions with the ground truth labels. For Arabic OCR, this means the prediction of a character sequence in a handwritten word can be made without explicitly segmenting each individual character—a challenging task due to the cursive nature of Arabic and the distinct writing styles of different writers. Both of the aforementioned dropout layers randomly drop neurons at a rate of 0.2 during training to force the model to better generalize to unseen data.

To boost the accuracy of the predicted text, we employed Word Beam Search decoding in conjunction with the KenLM language model for post-processing. Word Beam Search decoding generates possible character sequences constrained by a dictionary of valid words, ensuring that only logical word combinations are considered. To further improve the quality of the output, a probability-based language model, KenLM, is used.

KenLM is an n-gram-based statistical language model that predicts the likelihood of a word sequence by analyzing patterns in the training data. An n-gram refers to a sequence of n words. For example, "كتاب عربي" (*Arabic book*) is a bigram or 2-gram because it is a pair of consecutive words. A 3-gram or trigram considers word triplets, whereas a 4-gram evaluates a sequence of four words. The idea behind increasing n is for the language model to capture more context within a word sequence, thereby improving its predictive capabilities. While increasing n generally improves the quality and coherence of the generated word sequence, it also increases the computational cost of running KenLM.

2) Model Training Details

OCR is a sequence-to-sequence problem because a sequence of characters is recognized from the images, and a sequence of characters is predicted to match the characters seen from the handwritten text. For this specific task, we used the CTC as our loss function during our training process, lasting 300 epochs. The Adadelta optimizer was used with an initially high learning rate of 1.0. The hyperparameters for model training are mentioned below in Table VI:

Table VI
GENERAL TRAINING HYPERPARAMETERS

Parameter	***Value***
Optimizer	Adadelta
Loss Function	CTC
Initial Learning Rate	1.0
Batch Size	128
Number of Epochs	300
Dropout Rate	0.2

3) Overfitting Reduction

To reduce the onset of overfitting to the training set, we implemented dropout layers directly after the ResNet32 block and after the three consecutive BiLSTM layers. The rate at which random neurons are dropped during the training phase is 0.2 for both dropout layers. Additionally, we used various callbacks to help the model not get stuck in a local minimum and to boost generalization capabilities. The two main callbacks for these goals are ReduceLROnPlateau and EarlyStopping. ReduceLROnPlateau is a scheduler that adaptively reduces the learning rate whenever the validation loss plateaus for every 5 epochs. The reduction in the learning rate is by a factor of 0.5 to ensure smoother network convergence. The learning rate scheduler's parameters are detailed in Table VII. To prevent wasting time and resources on overtraining and to avoid overfitting, EarlyStopping with a patience of 50 epochs is used. If the model does not improve on the evaluation metric monitored by EarlyStopping – validation loss in our case – then, model training will be halted. Its parameters are listed in Table VIII.

Table VII
LEARNING RATE SCHEDULER PARAMETERS

Parameter	***Value***
Learning Rate Scheduler	ReduceLROnPlateau
Monitor	Validation Loss
Scheduler Reduction Factor	0.5
Scheduler Patience	3 epochs
Mode	Min
Minimum Learning Rate	1×10^{-8}

Table VIII
EARLY STOPPING PARAMETERS

Parameter	***Value***
Early Stopping Metric	Validation Loss
Early Stopping Patience	50 epochs
Mode	Min

4) Supplementary Callbacks

To supplement the training process, we utilized additional callback functions. We periodically saved the model every 25 epochs, outputted training and validation metrics for each epoch, and cleared the displayed outputs in the Jupyter Notebook every 10 epochs to improve the notebook's reaedability during model training.

B. Post-processing

After training the models, we improved the accuracy of the predicted text by integrating KenLM with Word Beam Search decoding. We trained KenLM on our dataset's corpus of text with increasing n-gram orders: 2-gram (bigrams), 3-gram (trigram), and 4-gram. With these n-grams, we could conduct ablation studies to see the impact of higher-order n-grams in

refining the decoded outputs of our OCR system based on our evaluation metrics.

C. Performance Evaluation

Due to the nature of this task, especially with complex scripts such as Arabic, we had to resort to robust metrics to evaluate the performance of the model. These are character error rate (CER) and word error rate (WER). CER measures the character-level differences between the predicted text and the groundtruth, while WER concerns word-level discrepancies. These particular metrics are derived from calculating the Levenshtein distance, a measure of how different a sequence of characters is from another. The Levenshtein Distance is the minimum number of transformations necessary to convert a sequence into another sequence. These transformations to a given sequence could be insertions, deletions, or substitutions. Below are the similar mathematical formulations for CER and WER:

$$\text{CER} = \frac{S + D + I}{N}, \tag{4}$$

- S = Number of character substitutions
- D = Number of character deletions
- I = Number of character insertions
- N = Total number of characters in the reference (ground truth)

$$\text{WER} = \frac{S + D + I}{N}. \tag{5}$$

- S = Number of word substitutions
- D = Number of word deletions
- I = Number of word insertions
- N = Total number of words in the reference (ground truth)

V. Results and Discussion

A. Experimental Conditions

Model training was conducted on two NVIDIA A100 (SXM4) Tensor Core GPUs, each with 80 GB of memory. One was provided by the American University of Sharjah's (AUS) Computer Science and Engineering (CSE) Department from their Artificial Intelligence (AI) lab, while another was accessed remotely through RunPod's platform. This setup facilitated the parallel execution of two seeds to obtain two distinct models. Upon generating two distinct models, we computed the average and standard deviation of the evaluation metrics across both runs. This approach helped ensure the model's stability and reliability when applied to real-world data.

The training environment was configured with Python 3 and TensorFlow libraries and ran on Ubuntu 24.04.1. GPU computations were optimized using CUDA version 12.4 and cuDNN, ensuring efficient resource utilization. The training time varied based because of the shared CPU usage on the AUS CSE AI Lab's A100 GPU. The average training time is 2 hours, 48 minutes, and 34 seconds.

A key challenge we encountered during experimentation involved the capability of loading the full models to either resume training or re-calculate the evaluation metrics. This difficulty arose due to the addition of a custom layer for calculating the CTC Loss through the CTC layer, which entailed ensuring its compatibility with TensorFlow's serialization and deserialization framework.

To address this concern, each custom layer was implemented to fully support model serialization and deserialization. The `get_config()` method was used to store initialization parameters, ensuring the proper reconstruction of the CTC layer's configurations. Furthermore, the `@tf.keras.utils.register_keras_serializable()` decorator was employed to make the layer compatible with TensorFlow's saving and loading mechanisms. The `call()` method defined the forward pass logic, ensuring consistency during both training and inference. By following these practices, the custom layer was seamlessly integrated into the TensorFlow workflow, enabling efficient experimentation and potential deployment across various setups.

B. Main Results

We assessed our models using the evaluation metrics averaged across both seeds, 42 and 503, as shown in Table IX. For this analysis, we only consider the evaluation on the unseen test data.

1) Baseline Performance without KenLM

Without integrating a language model, the model achieved a mean CER of approximately 1.894 (standard deviation of 0.121) and a mean WER of about 2.543 (0.196). These baseline results are relatively high, highlighting the inherent challenges of Arabic OCR—especially with handwritten text. Not only did the model have to handle significant variability in how a writer forms their characters and words, it also had to deal with the cursive nature of Arabic and the presence of diacritics. These characteristics further complicate the model's ability to learn how to correctly predict a character sequence. The extremely high CTC test loss, averaging around 182.699 (3.016), underscores the complexity of this task in the absence of linguistic constraints during decoding.

2) Impact of KenLM Integration

Once we integrated KenLM during the decoding stage, our CER and WER metrics improved considerably. Specifically, introducing a 2-gram KenLM model decreased the CER from 1.894 to approximately 0.903 (standard deviation of 0.0421) and the WER from 2.543 to around 0.998 (0.0017). This significant improvement highlights the essential role that contextual, language-level information plays in narrowing down valid word sequences, thereby mitigating many ambiguities that may arise solely from the model's raw character predictions. Essentially, using a language model effectively guides the Word Beam Search decoder towards more linguistically coherent and proper output sequences.

Table IX
TEST RESULTS: LOSS, CER, AND WER

Metric	Mean No KenLM - Baseline (Std)	Mean 2-gram KenLM (Std)	Mean 3-gram KenLM (Std)	Mean 4-gram KenLM (Std)
Test Loss	182.699 (3.016)	-	-	-
CER	1.894 (0.121)	0.903 (0.0421)	0.903 (0.0418)	0.903 (0.0417)
WER	2.543 (0.196)	0.998 (0.00170)	0.998 (0.00245)	0.998 (0.00255)

The performance improvements obtained by using KenLM are not only substantial but also consistent. The low standard deviation values for both CER and WER after integrating KenLM indicate that the improvements are stable across both runs. This consistency is important for practical applications, suggesting that the approach does not rely on favorable splits, for example. Instead, combining the model with KenLM can result in better generalizations and reliably reduce error rates.

3) Comparison of Different n-gram Orders in KenLM

To investigate the effect of capturing more extensive contextual dependencies, we experimented with 2-gram, 3-gram, and 4-gram KenLM models. Surprisingly, the results for CER and WER remained statistically similar across these different n-gram orders. For instance, the CER consistently hovered around 0.903 irrespective of whether a 2-gram, 3-gram, or 4-gram model was employed. Similarly, the WER remained at approximately 0.998 across all three configurations.

This observation suggests that increasing the context window beyond bigrams yields minimal measurable improvements within our specific setup. However, this may instead be explained by the subpar models produced during training, even if these are our best-validated models. The bottleneck in performance might lie in the feature extractor or the sequential handling of the feature maps. Therefore, it is important to acknowledge that the effectiveness of the KenLM n-gram models is inherently tied to the performance of the underlying neural network. That is, the neural network's performance severely constrains the benefits of integrating KenLM as a post-processor in an OCR task.

To conclude, while the baseline CNN-BiLSTM model with a ResNet32 backbone and CTC loss demonstrate forgettable results, post-processing with a language model, such as KenLM, significantly alleviates this issue. The language model integration reduces both CER and WER by more than half, illustrating how combining a neural network with a language model can yield a much more accurate handwriting OCR system for Arabic. Notably, these improvements are consistent across both random seeds and do not appreciably increase with higher-order n-grams, probably due to the fact that the underlying network could perform better.

Overall, these findings further highlight that post-processing is the key to unlocking better performance in Arabic OCR systems. Moreover, this does make us wonder how much better large language models would be as the post-processing component in an Arabic OCR pipeline.

VI. CONCLUSIONS AND FUTURE WORK

Overall, we explored deep learning techniques to improve the recognition of handwritten Arabic manuscripts. Arabic holds a key cultural significance across hundreds of millions of people, and ensuring that there is a framework to preserve and digitize Arabic handwriting is essential in the sustainability of the preservation of the language and extending its online presence. Our research also addressed the complexity of Arabic handwriting, including its cursive nature, various styles, and the use of diacritics. We explored the different levels of HTR and the use of CTC loss as our loss function. In addition, we reviewed the distinct characteristics of the Arabic language and the various challenges, such as high variability in styles, right-to-left directionality, and structural bias.

As part of our literature review, we examined different Arabic datasets, including the one we used, KHATT, and other preprocessing techniques. Throughout the papers surveyed, various models were used to perform Arabic handwriting text recognition, such as simple CNNs, VGG, CNN-BiLSTM, Vision Transformers, and some involved CTC and a language model for refined results. After resizing the dataset's images, our CNN-BiLSTM architecture gained a CER of 1.894 and a WER of 2.543. However, the introduction of KenLM proved valuable as the CER dropped to around 0.9 and 0.99 for CER and WER, respectively.

There are many potential avenues we will be exploring in future work, such as adding a BlockProcessor that can help standardize the input blocks of handwritten text into the model. Additionally, gaining access to the original KHATT dataset could prove helpful in exploring other ways to perform preprocessing on the dataset. Incorporating an attention-based approach with positional embeddings could also further improve the model's ability to focus on a text's relevant parts, especially for longer documents. Moreover, LLMs may improve the refinement of the output of our model over KenLM and are a promising means of improving the model's accuracy. Testing on different feature extractors could provide different insights compared to the CNNs employed and could better recognize specific text features. To avoid overfitting, the addition of L2 regularization may help the need to train for more epochs to capture more intricate details of handwriting. Finally, improving our CTC loss implementation may ensure that we receive more stable and effective training outcomes for a more reliable model.

These findings demonstrate that while deep learning architectures provide a strong foundation for HTR tasks, incorporating language models is essential for achieving optimal

accuracy in complex scripts like Arabic. Moving forward, exploring more sophisticated language modeling techniques and refining neural network architectures will further enhance the precision and reliability of digitizing Arabic manuscripts. Imagine a future where every handwritten Arabic document, from timeless classics to modern literary works, is effortlessly preserved and accessible, seamlessly bridging cultural heritage with the digital age.

REFERENCES

[1] M. Saeed, A. Chan, A. Mijar, J. Moukarzel, G. Habchi, C. Younes, A. Elias, C.-W. Wong, and A. Khater, "Muharaf: Manuscripts of handwritten arabic dataset for cursive text recognition," *arXiv preprint arXiv:2406.09630*, 2024.

[2] M. A. Ayuba, "Information and communication technologies in preserving arabic and islamic manuscripts," *Global Journal Al-Thaqafah*, vol. 3, no. 2, pp. 7–14, 2013.

[3] A. Mutawa, M. Y. Allaho, and M. Al-Hajeri, "Machine learning approach for arabic handwritten recognition," *Applied Sciences*, vol. 14, no. 19, p. 9020, 2024.

[4] S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, M. Alshayeb, M. T. Parvez, V. Märgner, and G. A. Fink, "Khatt: An open arabic offline handwritten text database," *Pattern Recognition*, vol. 47, no. 3, pp. 1096–1112, 2014.

[5] S. S. El-Dabi, R. Ramsis, and A. Kamel, "Arabic character recognition system: A statistical approach for recognizing cursive typewritten text," *Pattern recognition*, vol. 23, no. 5, pp. 485–495, 1990.

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

[7] M. T. Parvez and S. A. Mahmoud, "Offline arabic handwritten text recognition: a survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, pp. 1–35, 2013.

[8] S. Faizullah, M. S. Ayub, S. Hussain, and M. A. Khan, "A survey of ocr in arabic language: applications, techniques, and challenges," *Applied Sciences*, vol. 13, no. 7, p. 4584, 2023.

[9] A. Mezghani, R. Maalej, M. Elleuch, and M. Kherallah, "Recent advances of ml and dl approaches for arabic handwriting recognition: A review," *International Journal of Hybrid Intelligent Systems*, vol. 19, no. 1, 2, pp. 61–78, 2023.

[10] J. H. AlKhateeb, J. Jiang, J. Ren, and S. Ipson, "Component-based segmentation of words from handwritten arabic text," *International Journal of Computer Systems Science and Engineering*, vol. 5, no. 1, 2009.

[11] F. M. Nashwan, M. A. Rashwan, H. M. Al-Barhamtoshy, S. M. Abdou, and A. M. Moussa, "A holistic technique for an arabic ocr system," *Journal of Imaging*, vol. 4, no. 1, p. 6, 2017.

[12] L. Mosbah, I. Moalla, T. M. Hamdani, B. Neji, T. Beyrouthy, and A. M. Alimi, "Adocrnet: A deep learning ocr for arabic documents recognition," *IEEE Access*, 2024.

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

[14] R. Ahmad, S. Naz, M. Z. Afzal, S. F. Rashid, M. Liwicki, and A. Dengel, "A deep learning based arabic script recognition system: benchmark on khat." *Int. Arab J. Inf. Technol.*, vol. 17, no. 3, pp. 299–305, 2020.

[15] S. Aabed and A. Khairaldin, "An end-to-end, segmentation-free, arabic handwritten recognition model on khatt," *arXiv preprint arXiv:2406.15329*, 2024.

[16] T. Wang, Y. Zhu, L. Jin, C. Luo, X. Chen, Y. Wu, Q. Wang, and M. Cai, "Decoupled attention network for text recognition," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 216–12 224.

[17] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "Trocr: Transformer-based optical character recognition with pre-trained models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 13 094–13 102.

[18] A. K. Bhunia, S. Ghose, A. Kumar, P. N. Chowdhury, A. Sain, and Y.-Z. Song, "Metahtr: Towards writer-adaptive handwritten text recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 830–15 839.

[19] S. A. Mahmoud, I. Ahmad, M. Alshayeb, W. G. Al-Khatib, M. T. Parvez, G. A. Fink, V. Märgner, and H. El Abed, "Khatt: Arabic offline handwritten text database," in *2012 International conference on frontiers in handwriting recognition*. IEEE, 2012, pp. 449–454.

[20] T. Steinherz, N. Intrator, and E. Rivlin, "A comprehensive survey of ocr techniques," *Computer Vision and Image Understanding*, vol. 83, no. 2, pp. 237–270, 1999.

[21] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 690–706, 1996.

[22] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[24] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] T. Clanuwat, A. Lamb, and A. Kitamoto, "Kuronet: Pre-modern japanese kuzushiji character recognition with deep learning," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 607–614.

[27] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[28] A. K. Bhunia, A. Das, A. K. Bhunia, P. S. R. Kishore, and P. P. Roy, "Handwriting recognition in low-resource scripts using adversarial learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4767–4776.

[29] F. P. Such, D. Peri, F. Brockler, H. Paul, and R. Ptucha, "Fully convolutional networks for handwriting recognition," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2018, pp. 86–91.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[31] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

[33] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[34] A. Graves, S. Fernandez, and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in *International Conference on Artificial Neural Networks*. Springer, 2007, pp. 549–558.

[35] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 369–376.

[36] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[37] A. Praneetha, S. Harisa, K. Satish, and B. Vivekananda, "Handwritten text recognition using word beam search," in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2023, pp. 1878–1882.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[39] R. Ahmed, K. Dashtipour, M. Gogate, A. Raza, R. Zhang, K. Huang, A. Hawalah, A. Adeel, and A. Hussain, "Offline arabic handwriting recognition using deep machine learning: A review of recent advances," in *Advances in Brain Inspired Cognitive Systems: 10th International Conference, BICS 2019, Guangzhou, China, July 13–14, 2019, Proceedings 10*. Springer, 2020, pp. 457–468.

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

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

[42] A. Abdulmuttalib and M. Khalid, "Rtl text recognition: Challenges and future directions," *International Journal of Computer Vision*, vol. 129, no. 2, pp. 322–336, 2021.

[43] A. Khattab, N. Salim, and S. Badran, "Arabic handwritten text recognition: Baseline detection and structural challenges," *Pattern Recognition Letters*, vol. 125, pp. 70–78, 2019.

[44] A. Chan, A. Mijar, M. Saeed, C.-W. Wong, and A. Khater, "Hatformer: Historic handwritten arabic text recognition with transformers," *arXiv preprint arXiv:2410.02179*, 2024.

[45] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, H. Amiri *et al.*, "Ifn/enit-database of handwritten arabic words," in *Proc. of CIFED*, vol. 2. Citeseer, 2002, pp. 127–136.

[46] D. Lee, S. Ismael, S. Grimes, D. Doermann, S. Strassel, and Z. Song, "Madcat phase 1 training set," *LDC2012T15. DVD. Philadelphia: Linguistic Data Consortium*, 2012.

[47] S. Al-Ma'adeed, D. Elliman, and C. A. Higgins, "A data base for arabic handwritten text recognition research," in *Proceedings eighth international workshop on frontiers in handwriting recognition*. IEEE, 2002, pp. 485–489.

[48] N. Otsu, "Aa threshold selection method from grey scale histogram," *IEEE Transactions on Systems Man and Cybernetics*, 1979.

[49] A. T. Sahlol, C. Y. Suen, M. R. Elbasyouni, and A. A. Sallam, "A proposed ocr algorithm for the recognition of handwritten arabic characters," *J. Pattern Recognit. Intell. Syst*, vol. 2, pp. 8–22, 2014.

[50] M. Liao, Z. Zou, Z. Wan, C. Yao, and X. Bai, "Real-time scene text detection with differentiable binarization and adaptive scale fusion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 919–931, 2022.

[51] S. Alghyaline, "Arabic optical character recognition: A review." *CMES-Computer Modeling in Engineering & Sciences*, vol. 135, no. 3, 2023.

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

[53] K. S. Younis, "Arabic hand-written character recognition based on deep convolutional neural networks," *Jordanian Journal of Computers and Information Technology*, vol. 3, no. 3, 2017.

[54] H. M. Najadat, A. A. Alshboul, and A. F. Alabed, "Arabic handwritten characters recognition using convolutional neural network," in *2019 10th International Conference on Information and Communication Systems (ICICS)*, 2019, pp. 147–151.

[55] M. Elleuch, R. Maalej, and M. Kherallah, "A new design based-svm of the cnn classifier architecture with dropout for offline arabic handwritten recognition," *Procedia Computer Science*, vol. 80, pp. 1712–1723, 2016.

[56] S. K. Jemni, Y. Kessentini, S. Kanoun, and J.-M. Ogier, "Offline arabic handwriting recognition using blstms combination," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 31–36.

[57] F. Stahlberg and S. Vogel, "The qcri recognition system for handwritten arabic," in *Image Analysis and Processing—ICIAP 2015: 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part II 18*. Springer, 2015, pp. 276–286.

[58] M. F. BenZeghiba, J. Louradour, and C. Kermorvant, "Hybrid word/part-of-arabic-word language models for arabic text document recognition," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 671–675.

[59] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a cnn-lstm network," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 639–645.

[60] M. El Khayati, I. Kich, and Y. Taouil, "Cnn-based methods for offline arabic handwriting recognition: A review," *Neural Processing Letters*, vol. 56, no. 2, p. 115, 2024.

[61] L. Berriche, A. Alqahtani, and S. RekikR, "Hybrid arabic handwritten character segmentation using cnn and graph theory algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 36, no. 1, p. 101872, 2024.

[62] H. AlShehri, "Deepahr: a deep neural network approach for recognizing arabic handwritten recognition," *Neural Computing and Applications*, pp. 1–13, 2024.

[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[66] S. Alghyaline, "Optimised cnn architectures for handwritten arabic character recognition," *Computers, Materials & Continua*, vol. 79, no. 3, pp. 4905–4924, 2024. [Online]. Available: http://www.techscience.com/cmc/v79n3/57159

[67] S. Momeni and B. BabaAli, "A transformer-based approach for arabic offline handwritten text recognition," *Signal, Image and Video Processing*, vol. 18, no. 4, pp. 3053–3062, 2024.

[68] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.