Suez Canal University
Faculty of Engineering
Department of Computers & Control

Arabic Handwriting Recognition

Graduation project report submitted to the
Faculty of Engineering
Suez Canal University
In partial fulfillment to the requirements for the
Degree of Bachelor of Engineering in Computers & Control

Supervised By

Prof. Khalid Abd El Salam

Prepared By

Ayman Saber Gad
Ahmed Taha Ahmed Abd El Rahim
Ahmed Nagah Ahmed Mohamed
Mohammed Fathi Mohammed Mohammed Ali
Abanoub Ayed Khalaf
Reem Mohammed Fouad
Rawan Gamal Mohammed
Nada Mahmoud Mohammed
Nada Hussein Asran
Kerollos Samir Fathi
Mohammed Abd El Fattah Fathi

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of this project. Their unwavering support, guidance, and encouragement have been invaluable throughout this journey.

First and foremost, we extend our deepest gratitude to the Almighty God for granting us the strength, wisdom, and perseverance to undertake and complete this project. Without the divine blessings and grace, none of this would have been possible.

We are immensely grateful to Dr. Khaled Abdelsalam, whose expertise and guidance have been instrumental in shaping the direction of this project. Dr. Khaled's extensive knowledge, insightful feedback, and constant motivation have been crucial in refining our ideas and methodologies. Their mentorship has truly been a privilege, and we are indebted to their invaluable contributions.

We would like to extend our appreciation to our fellow teammates who have collaborated on this project. Their commitment, hard work, and dedication have been instrumental in overcoming challenges and achieving our goals. We are grateful for the sense of unity we shared throughout this endeavor.

We would also like to express our gratitude to our families and friends who have provided unwavering encouragement and understanding during this project. Their love, patience, and belief in our abilities have been a constant source of motivation.

Lastly, we would like to acknowledge the contributions of all the individuals who have provided assistance, feedback, or resources at various stages of this project. Your invaluable insights and support have enriched this work and made it possible to achieve our objectives.
Thank you.

PUBLICATIONS

The Publications section of our graduation project book highlights the significant research contributions made by our team during our academic journey. This section encompasses the various papers, articles, and conference presentations that showcase our dedication to advancing knowledge in our field. Each publication reflects our commitment to rigorous research methodologies, innovative problem-solving, and the dissemination of findings to the broader academic and professional communities. We take pride in the recognition and validation our work has received through these scholarly outputs, which underscore our team's efforts and achievements.

First Publication: We are proud to announce that our research, titled "A Comprehensive Approach to Arabic Handwriting Recognition: Deep Convolutional Networks and Bidirectional Recurrent Models for Arabic Scripts" was presented at the 6th Annual Conference “Artificial Intelligence: Future Prospects for Achieving the Sustainable Development Goals” for Student Research and Creativity held on May 7, 2024, at Suez Canal University. This study, conducted by Ayman Saber Gad Mohammed, Ahmed Taha Ahmed Abd El Rahim, and Dr. Khaled Abd El Salam from the Faculty of Engineering's Electrical Department, explores advanced deep learning techniques for Arabic handwriting recognition. These outcomes demonstrate a significant advancement over traditional methods, highlighting our contribution to the field of Arabic handwriting recognition.

The 6th Annual Conference for Student Research and Creativity (7 MAY 2024)

(8): ES-O-ENG

A Comprehensive Approach to Arabic Handwriting Recognition: Deep Convolutional Networks and Bidirectional Recurrent Models for Arabic Scripts

Ayman Saber Gad Mohammed, Ahmed Taha Ahmed Abd El Rahim, and Khaled Abd El Salam

Electrical Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt

ABSTRACT: Arabic handwriting recognition is a significant area of study because it has a lot of applications in many different fields like education, data analysis, and information retrieval, but it has peerless challenges due to the complexities of Arabic language nature, the diversity of forms of the same letter, and the immeasurable variations in writing styles. Our approach is to apply deep learning techniques, precisely focusing on Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (Bi-LSTM) networks adapted for recognizing handwritten Arabic text. We utilized the KHATT dataset for training and evaluation, employing proper preprocessing steps. Our method, which focused on the ResNet152 architecture for feature extraction, achieved impressive results with a character error rate of roughly 2.96% and an accuracy rate of 97.04% on the test dataset. These results outperform earlier techniques, marking a significant improvement in accuracy.

Keywords arabic handwritten text recognition, deep learning, document digitization, optical character recognition (OCR).

Second Publication: We are also pleased to announce that our research, titled "A Comprehensive Approach to Arabic Handwriting Recognition: Deep Convolutional Networks and Bidirectional Recurrent Models for Arabic Scripts," has been published in the **International Journal of Telecommunications (IJT)**, Volume 15, on June 15, 2024. This study, authored by Ayman Saber Gad Mohammed, Ahmed Taha Ahmed Abd El Rahim, and Khaled Abd El Salam from the Faculty of Engineering's Electrical Department, delves into cutting-edge deep learning techniques for recognizing Arabic handwriting. By leveraging Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (Bi-LSTM) networks, our method achieved outstanding results, including a character error rate of roughly 2.96% and an accuracy rate of 97.04% on the KHATT dataset. These results significantly surpass traditional approaches, underscoring our substantial contributions to the domain of Arabic handwriting recognition.

INTERNATIONAL JOURNAL OF
Telecommunications
AIR DEFENSE COLLEGE

A Comprehensive Approach to Arabic Handwriting Recognition: Deep Convolutional Networks and Bidirectional Recurrent Models for Arabic Scripts

Citation: Saber, A.; Taha, A.; Abd El Salam, K. A Comprehensive Approach to Arabic Handwriting Recognition: Deep Convolutional Networks and Bidirectional Recurrent Models for Arabic Scripts. *Inter. Jour. of Telecommunications*, ***IJT*'20xx**, *Vol. xx, Issue xx.* https://ijt-adc.org/articles/xxxxx

Editor-in-Chief: Firstname Lastname
Received: date
Accepted: date
Published: date

Ayman Saber[1,*], Ahmed Taha[1] and Khaled Abd El Salam[1,2]
[1] Electrical Engineering Department, Suez Canal University, Ismailia, Egypt.
[2] Department of Information System, College of Information Technology, Misr University for Science and Technology (MUST), 6th of October City 12566, Egypt.
* Correspondence: ayman.saber@eng.suez.edu.eg.
Contributing authors: UGS.161750@eng.suez.edu.eg khaled.abdelsalam@eng.suez.edu.eg, Khaled.abdelsalam@must.edu.eg

Abstract: Arabic handwriting recognition presents unique challenges due to the complexities of Arabic calligraphy and variations in writing styles. Proposing a novel approach to address these challenges by leveraging advanced deep learning techniques. This focus is on Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (Bi-LSTM) networks, which are tailored specifically for recognizing handwritten Arabic text. Utilizing the KHATT dataset for comprehensive training and evaluation, implementing rigorous preprocessing steps to enhance data quality. Central to this methodology is the Res-Net152 architecture for feature extraction, which has proven highly effective. This approach achieved remarkable results, with a character error rate of approximately 2.96% and an accuracy of 97.04% on the testing dataset. These results significantly outperform the previous method, representing a substantial advancement in the field of Arabic handwriting recognition. The study demonstrates the potential of deep learning models in overcoming the unique challenges posed by Arabic script, paving the way for further improvements and applications.

Keywords: Optical Character Recognition (OCR), Artificial Neural Networks, Text segmentation, Document Digitization, KHATT Dataset.

ABSTRACT

This graduation project, titled "Arabic Handwriting Recognition," was undertaken by a team of 11 students at the Faculty of Engineering, Suez Canal University, under the supervision of Prof. Khalid Abd El Salam. The project's primary objective was to develop a system capable of recognizing and digitizing handwritten Arabic scripts. The system was designed to convert various types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data.

The project involved several stages, including data collection and preparation, system planning and design, system implementation, AI training, and results analysis. The team utilized various techniques and technologies, such as image preprocessing, character recognition models, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and deep learning algorithms.

Despite facing challenges such as the complexity of Arabic calligraphy, variations in writing styles, and the lack of publicly available standardized datasets for handwritten Arabic text, the team successfully developed a system that achieved high levels of accuracy and robustness in recognizing handwritten Arabic text. The project not only contributes to the field of Optical Character Recognition (OCR) but also opens up new avenues for future research and development in the specialized subset of Arabic Handwriting Recognition (AHR).

LIST OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHAWP	Arabic Handwritten Alphabets, Words and Paragraphs
AHR	Arabic Handwriting Recognition
ANN	Artificial Neural Network
Bi-LSTM	Bidirectional Long Short-Term Memory
CER	Character Error Rate
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRAFT	Character Region Awareness for Text Detection
CRUD	Create, Read, Update, Delete
CTC	Connectionist Temporal Classification
DocTr	Document Image Transformer
DOM	Document Object Model
FC	Fully connected
GPU	Graphics Processing Unit
ITC	Ink-to-Text Conversion
KHATT	KFUPM Handwritten Arabic TexT
LLM	Large Language Models
LSTM	Long Short-Term Memory
MCR	Magnetic Character Recognition
NLP	Natural Language Processing
OCR	Optical Character Recognition
OMR	Optical Mark Recognition
OOP	Object-Oriented Programming
PDO	PHP Data Objects
PHP	Hypertext Preprocessor
RAM	Random Access Memory
ResNet	Residual Network
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
VGG	Visual Geometry Group

LIST OF TABLES

CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

1.1 Overview

Optical Character Recognition (OCR) is a technology that enables the conversion of different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable, searchable and machine-readable data [1][2][3][4][5]. OCR systems work by analyzing the text in these documents and translating it into machine-encoded text.
Arabic Handwriting Recognition (AHR) is a specialized subset of OCR focused on recognizing and digitizing handwritten Arabic scripts [6][7].

1.1.1 OCR Overview

OCR systems use various techniques to recognize and extract characters from images or scanned documents, allowing them to be edited, searched, and processed digitally [8].

OCR has greatly simplified the digitization process of printed documents, making it easier to convert large volumes of physical documents into electronic formats [3][9].

It has also made it possible to extract text from images, such as photographs or screenshots, enabling the recognition of text in various contexts [10].

OCR is widely used as a form of data entry from printed paper data records. whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printed data, or any suitable documentation [3]. It is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed online, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining.

1.1.2 Types Of OCR

Optical character recognition (OCR) is usually referred to as an off-line character recognition process [11], meaning that the system scans and recognizes static images of the characters. It refers to the mechanical or electronic translation of images of Handwritten characters or printed text into machine code without any variation.

Figure 1.1 [12] shows types of OCR [13].

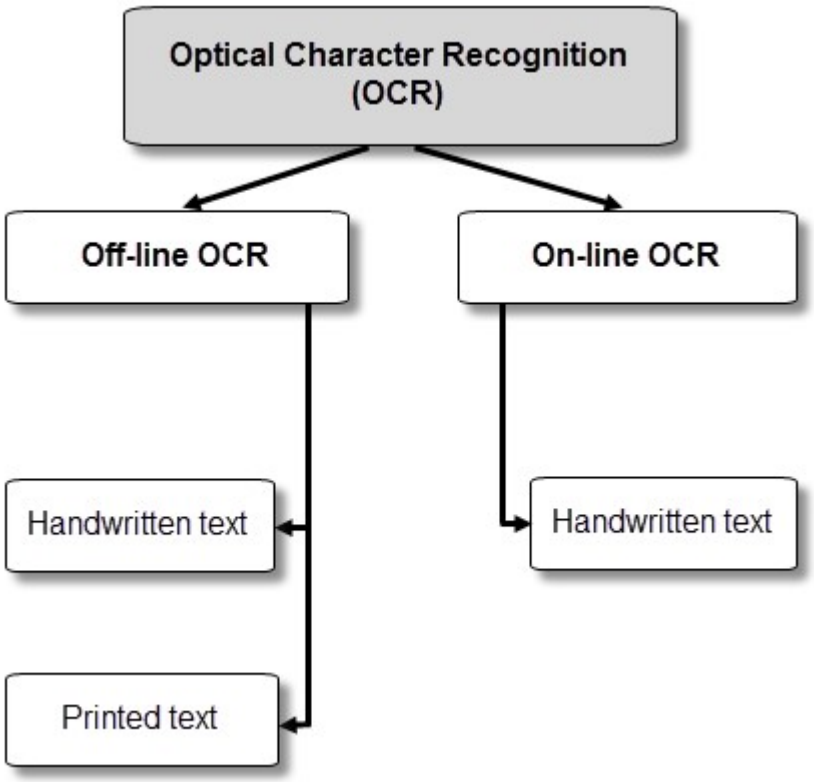


Figure 1.1: Types of character recognition

1.1.2.1 Arabic Printed and Handwritten OCR

The characters are written on paper or directly on a touch screen using a pen or fingers. Printed OCR takes an image containing printed text as input and converts it into editable text.

Recognizing Arabic handwritten text is more challenging than recognizing printed text [14]. People have diverse ways of writing, which makes it difficult even for a human to recognize it.

Figure 1.2 [15] shows a sample of Arabic scripts. The handwritten script is usually written in cursive, even in Latin scripts. Segmenting cursive scripts is more complicated than segmenting printed scripts.

The handwritten script has many sizes, orientations, and resolutions compared with printed text, and there are no standard font sizes and orientations.

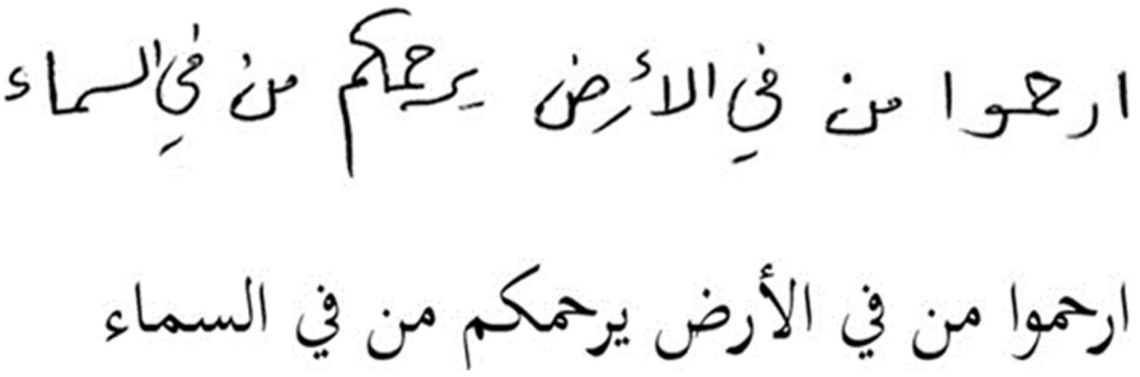


Figure 1.2: Sample handwritten and machine-printed Arabic texts.

1.1.2.2 Online Character Recognition System

Online character recognition is recognizing handwriting recorded with a digitizer as a time sequence of pen coordinates. The handwriting is captured and stored in digital form via different means in real time. Usually, a special pen is used in conjunction with an electronic surface. As the pen moves across the surface, the two-dimensional coordinates of successive points are represented as a function of time and are stored in order. It is accepted that the online method of recognizing handwritten text has achieved better results than its offline counterpart. This may be attributed to the fact that more information may be captured in the online case such as the direction, speed, and the order of strokes of the handwriting.

Online handwriting recognition has several distinguishing features that must be exploited to get more accurate results than online recognition [16][17][18].

- It is adaptive: immediate feedback is given by the writer whose corrections can be used to further train the recognizer.

- It is a real time process: It captures the temporal or dynamic information of the writing. This information consists of the number of pen strokes, the order of pen-strokes. The direction of the writing for each pen stroke and the speed of the writing within each pen stroke.

- Very little preprocessing is required. The operations such as smoothing, and feature extraction operations such as the detection of line orientations corner loops are easier and faster with the pen trajectory data than on pixel images.

- Segmentation is easy: Segmentation operations are facilitated by using the pen lift information particularly for hand printed characters.

On the other hand, the disadvantages of the online character recognition are as follows:

- The writer requires special equipment which is not as comfortable and natural to use as pen and paper.
- It cannot be applied to documents printed or written on papers, punching is much faster and easier than handwriting for small size alphabet such as English or Arabic.

1.1.2.3 Offline Character Recognition System

Offline handwriting recognition refers to the process of recognizing words that have been scanned from a surface (such as a sheet of paper) and are stored digitally in grayscale format [19]. After being stored, it is conventional to perform further processing to allow superior recognition. offline character recognition can be further grouped into two types:

- Magnetic Character Recognition (MCR)
- Optical Character Recognition (OCR)

In MCR, the characters are printed with magnetic ink. The reading device can recognize the characters according to the unique magnetic field of each character. MCR is mostly used in banks for check authentication. OCR deals with the recognition of characters acquired by optical means, typically a scanner or a camera. The characters are in the form of pixelated images, and can be either printed or handwritten, of any size, shape, or orientation. The OCR can be subdivided into handwritten character recognition and printed character recognition. Handwritten Character Recognition is more difficult to implement than printed character recognition due to diverse human handwriting styles and customs. In printed character recognition, the images to be processed are in the forms of standard fonts like Times New Roman.

1.1.3 Arabic handwriting Recognition Overview

Arabic handwriting recognition focuses specifically on the conversion of handwritten Arabic text into machine-readable text. It involves the analysis of handwritten Arabic characters to accurately recognize and interpret the handwritten content [20].

The process of Arabic handwriting recognition typically involves several steps:
First, the handwritten document or text is scanned or captured to create a digital image. Then, the recognition software analyzes the image and applies various techniques to segment the text into individual characters or words. These techniques may include stroke analysis, contour detection, and shape recognition.

Once the text is segmented, the recognition software compares the shapes and patterns of the characters against a database of known Arabic characters or employs machine learning algorithms to improve recognition accuracy. These algorithms are trained on large datasets of labeled Arabic handwriting samples to learn the variations and patterns in Arabic handwriting.

Arabic handwriting recognition has several applications across different domains [7]. It can be used to digitize handwritten Arabic documents, allowing for easier storage, search ability, and editing of the content. It also enables the integration of Arabic handwriting input in devices such as tablets or smartphones, facilitating natural and convenient input methods for Arabic users. Additionally, Arabic handwriting recognition plays a crucial role in text mining, information retrieval, and language processing tasks that involve handwritten Arabic text.

1.2 Problem Definition

The field of Arabic handwriting recognition has gained significant interest in recent years due to its practical applications in various domains, including document analysis, automatic text recognition, and postal services. The ability to accurately recognize handwritten Arabic text poses a significant challenge due to the complexities of Arabic calligraphy and the inherent variations in writing styles. Consequently, the development of powerful and effective techniques for Arabic handwriting recognition has become a pressing matter.

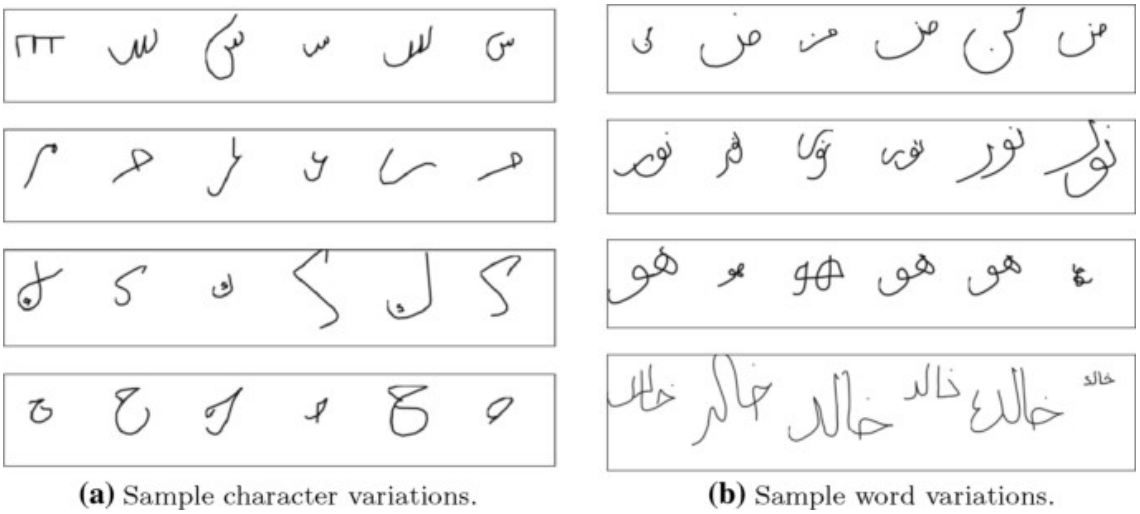
1.2.1 Importance of OCR for Arabic language

The Arabic language is considered one of the most important languages, because:

- Arabic is the fifth most spoken language in the world [21].
- Arabic is the Language of the Qur'an, the Holy Book of Islam.
- Rich Cultural History of Arabic in the World.
- Arabic is one of the oldest in the world with a wealth of knowledge that Archeologists to this day are still trying to uncover.
- Arabs have also made significant contributions in: Literature, Mathematics, Navigation, and Architecture.

1.2.2 Arabic Handwriting Recognition Challenges

Arabic handwriting recognition presents unique challenges shown in Figure 1.3 [22] compared to other scripts, including:
Arabic is a connected language, where letters are linked together, and the same letter can have different forms depending on its position within a word.



(a) Sample character variations. **(b)** Sample word variations.

Figure 1.3: The Variations in Words & Characters in Arabic

Handwritten Arabic exhibits significant variation among individuals due to factors such as writing style, spacing, and letter arrangement. These factors contribute to increased difficulty in accurately recognizing handwritten Arabic text.

Ambiguities and irregularities in handwritten Arabic script, such as ligatures, diacritics, and contextual variations. Despite the growing interest in Arabic handwriting recognition, significant gaps and challenges remain. Current recognition systems struggle to achieve acceptable accuracy and robustness, especially with unconstrained and noisy handwritten Arabic text. Additionally, the lack of publicly available standardized datasets for handwritten Arabic text hinders the development and evaluation of new algorithms and methodologies [23].

1.2.3 Problem Statement

The problem addressed in this project is the automation of grading handwritten Arabic exams. Traditional methods of exam grading are labor-intensive, time-consuming, and prone to errors. By leveraging advanced handwriting recognition techniques, we aim to create a system that can accurately assess and score handwritten responses in Arabic script, thereby eliminating the need for manual grading by instructors.

In this project, we have overcome these obstacles, filled the gaps, and we raised the level of efficiency in recognizing Arabic handwriting and benefiting from its applications in several fields.

1.3 Related Work

Arabic handwriting recognition (AHR) is a challenging task due to the cursive nature of Arabic script and the wide variability in handwriting styles. In this section, we review previous research efforts and methodologies related to AHR.

1.3.1 Traditional Approaches

Early AHR systems relied on classical pattern recognition and feature extraction techniques [24][25][26]. These approaches involved segmenting individual characters or words from handwritten text and then extracting handcrafted features. While these methods achieved moderate success, they struggled with handling the inherent variability and complexity of Arabic script [25][27][28].

1.3.2 Deep Learning Advances

Recent advancements in deep learning have led to significant improvements in AHR performance [29]. Convolutional Neural Networks (CNNs)[30] and Recurrent Neural Networks (RNNs)[31] have shown promising results in recognizing Arabic handwritten text without the need for explicit feature engineering. Models such as long short-term memory (LSTM)[31][32] networks and attention mechanisms have been applied to capture long-range dependencies and improve recognition accuracy, even for cursive Arabic script [33].

1.3.3 Challenges and Limitations

Despite the progress made in AHR, several challenges remain. Variations in handwriting styles, font types, and the lack of large-scale annotated datasets pose significant challenges to existing recognition systems [34].

1.3.4 Recent Trends and Future Directions

Emerging trends in AHR involve integrating deep learning with transfer and multi-task learning to enhance accuracy and generalization. Real-time online AHR systems are gaining traction, and creating public benchmarks and datasets is crucial for advancing research.

1.3.5 Comparison with Existing Work

Existing AHR systems have shown promising results [35][36], but challenges remain in unconstrained Arabic handwriting recognition. Our approach uses state-of-the-art deep learning techniques to address variability in handwriting styles through robust feature representations and model architectures. Despite the significant variation in Arabic handwriting, making accurate OCR recognition difficult, several programs partially address this issue, which we will review.

1.3.5.1 Google Lens

Google Lens [37] is a popular OCR application designed to offer text recognition capabilities. It uses image processing and deep learning algorithms to accurately extract text from images captured by the device's camera. Google Lens can recognize various languages and fonts, making it versatile for diverse use cases.

Google Lens leverages Google's vast dataset and machine learning models to continuously improve its accuracy and performance over time. It is available as a standalone app and is also integrated into various Google services such as Google Photos and Google Assistant.

Limitations:

- Not an open-source application.
- Limited Transparency: This lack of transparency can lead to uncertainty about the security and functionality of the software.
- Connectivity Dependency: Some features of Google Lens, such as real-time translation, require an internet connection to function.
- Privacy Concerns.

1.3.5.2 Tesseract OCR

Tesseract OCR [38] is currently an open-source OCR engine originally developed by Hewlett-Packard (HP)[39] and later maintained by Google [40]. It is widely used for text recognition in various applications and has been integrated into numerous software projects and applications.

Key Features:

- It's Open Source; meaning anyone can use it or develop it further.
- Tesseract OCR is known for its high accuracy in recognizing printed text in images, including complex fonts and languages.
- Language Support

Limitations:

- Very limited support for handwriting recognition, especially with Arabic.
- It has a somewhat complex setup.

1.3.5.3 Sakhr

Sakhr [41] is a commercial software made by Sakhr Software, it supports the Arabic language or the languages that use Arabic characters such as Farsi, Urdu Pashto. Sakhr claimed at the time to be the best available OCR for Arabic. According to US evaluators, Sakhr had 99.8% accuracy for the documents with high-quality images.

Key Features:

- High accuracy for Arabic script recognition.
- Supports multiple languages using Arabic script.
- Recognizes both printed and handwritten text.
- Provides both online and offline recognition.

Limitations:

- High cost due to being commercial software.

- Limited support for non-Arabic scripts.
- Runs exclusively on Windows operating systems.

Given the limitations of each of these solutions, there is a clear need for a different one; one that can address the unique challenges of unconstrained Arabic handwriting recognition. Our proposed approach aims to fill this gap by leveraging state-of-the-art deep learning techniques, robust feature representations, and innovative model architectures. By focusing on the variability in Arabic handwriting styles and employing comprehensive pre- and post-processing methods, our solution seeks to achieve higher accuracy and adaptability, making it a more effective tool for diverse applications.

CHAPTER 2

SYSTEM PLANNING

CHAPTER 2 SYSTEM PLANNING

System planning is a critical phase in the development of any software project. It involves defining the objectives, scope, and requirements of the system to ensure that it meets the needs of its users. In the context of our Arabic Handwritten OCR project, careful planning is essential to handle the complexities of recognizing handwritten Arabic characters and ensuring the system is robust, accurate, and user-friendly.

2.1 Requirement Gathering

Requirement gathering is the process of identifying the needs and constraints of the stakeholders involved in the project [42]. This phase involves collecting detailed and specific information to ensure that the system we develop aligns with the expectations and requirements of its users.

The primary requirements for our Arabic Handwritten OCR system are as follows:

- **Operate on mobile devices**: The program should be optimized to run efficiently on various mobile devices, including smartphones and tablets.
- **Recognize handwritten Arabic characters from any angle**: The system must accurately recognize handwritten Arabic characters regardless of the orientation in which they are written or scanned.
- **Preserve errors in the written words without correcting them**: The OCR system should capture the handwritten text as it is, including any mistakes, and not attempt to correct them. This is crucial for applications where the original written form, including errors, must be retained, like the automatic exam grading application for example.

2.2 Problems with Performing OCR

Developing Optical Character Recognition (OCR) for Arabic presents unique challenges due to the linguistic and script characteristics of the Arabic language [43]. Addressing these challenges requires a combination of advanced image processing techniques, robust language models, and extensive training data tailored specifically for the Arabic language and its various degrees. Continued research and development in these areas are crucial for improving the accuracy and reliability of Arabic OCR systems.

2.2.1 Geometric Warping

Geometric unwarping is a critical pre-processing step in the development of Optical Character Recognition (OCR) systems, particularly for languages such as Arabic. The process involves correcting distortions in the scanned images of documents to ensure that the text is accurately recognized.

2.2.1.1 Angle of Taking the Picture

- Distortion and Perspective: The angle at which a photo is taken can introduce distortion and perspective issues. Text may appear skewed or warped, making it harder for optical character recognition (OCR) software to accurately interpret characters.

- Reflections and Glare: Depending on the lighting conditions and angle, reflections and glare can obscure parts of the text, reducing OCR accuracy.

2.2.1.2 Cropping of the image

- Incomplete Text Capture: Improper cropping can lead to parts of the text being cut off or obscured. OCR systems rely on a clear view of the entire text to accurately recognize and extract characters.

- Background Noise: Unimportant details or backgrounds in the image can interfere with OCR algorithms, causing them to misinterpret text or ignore it altogether.

2.2.1.3 Unwarping of the Image

Image Distortion Correction: Images taken from an angle may need correction to straighten lines of text as shown in Figure 2.1 [44], which is crucial for accurate OCR. Distorted text lines can confuse OCR algorithms, leading to errors in text extraction.

Quality of Image Processing: The process of unwarping an image involves computational techniques to correct perspective and distortion, which can vary in effectiveness depending on the software used and the complexity of the image.

Figure 2.1: Example of Document Warping

2.2.2 Segmentation Of Paragraphs

Creating a model to extract text from entire paragraphs in Arabic OCR involves handling an enormous range of variations in paragraph structure, making it impracticable to create a model that can classify and extract text from every possible paragraph directly. The infinite variations necessitate a highly generalized model, making any small amount of data completely impractical to work with. To recognize text in paragraphs, the process should involve preprocessing; starting with line segmentation to isolate individual lines of text, word segmentation to break down lines into “islands”, meaning words or sub-words, using space detection algorithms.

CHAPTER 3

SYSTEM DESIGN

CHAPTER 3 SYSTEM DESIGN

Our system comprises two primary components: an image preprocessing module and a character recognition model. The preprocessing module is crucial because the recognition model is optimized to handle only the smallest possible segments of the image. In our Arabic Handwritten OCR Project, the preprocessing step prepares and segments the input images. This allows the recognition model to be trained more practically on a smaller data while achieving high accuracy.

3.1 Geometric Unwrapping

Geometric unwrapping is a crucial preprocessing step in Arabic Handwritten OCR to ensure that the text is properly aligned and readable for the OCR system. The main challenge addressed here is the distortion caused when images of handwritten text are taken at an angle, which can lead to skewed and curved text lines. To correct this, we need to straighten the text lines and flatten the paper surface.

3.1.1 The Naive Approach

Initially, we employed a straightforward method using OpenCV to detect and correct the skew in our images. This approach involved detecting straight lines within the image and using these lines to crop and align the image. Specifically, we used OpenCV's Hough Line Transform to detect the edges of the paper [45]. This method relies on the assumption that the paper's edges are straight and can be used as a reference to realign the text. The steps were as follows:

1. Convert the image to grayscale to reduce complexity.
2. Apply edge detection using the Canny edge detector.
3. Use the Hough Line Transform to identify the lines in the image.
4. Identify the lines that correspond to the paper's borders.
5. Crop and warp the image based on these lines to obtain a straightened text image.

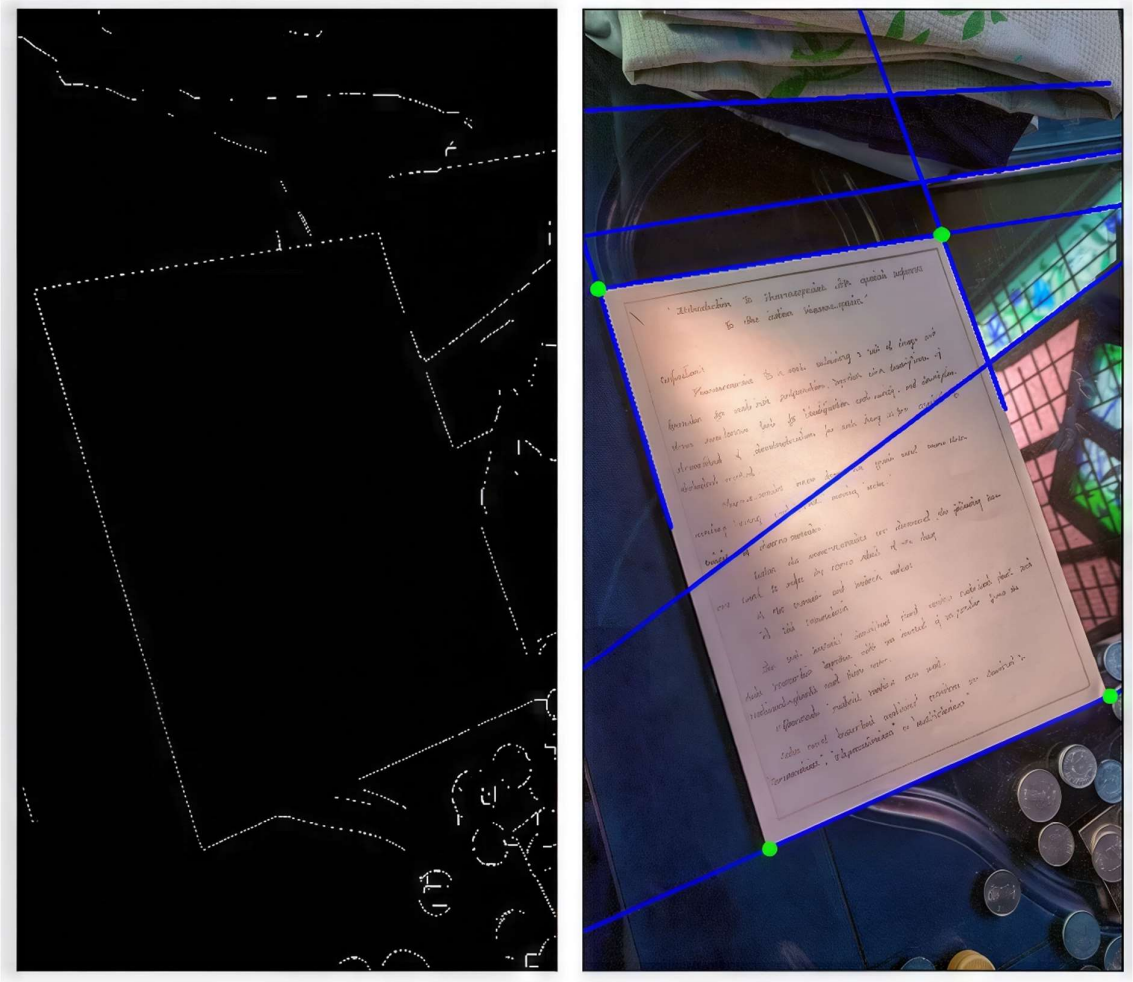


Figure 3.1: Document Unwarping Using Hough Line Transform

However, this naive approach has several limitations. First, it requires the entire paper to be visible within the image. If any part of the paper is cut off or obscured, the line detection fails, resulting in an error. Second, the background must be clearly distinguishable from the paper. If the paper's background is cluttered or similar in color and texture, the edge detection can be unreliable, leading to poor alignment and cropping. Additionally, this method assumes the paper is perfectly flat, which is rarely the case in real-world scenarios where paper might be crumpled or curved.

3.1.2 DocTr: Document Image Transformer

To address the limitations of the naive approach, we adopted a state-of-the-art method called DocTr: Document Image Transformer [46][47][48][49]. DocTr leverages deep learning techniques to perform both geometric unwarping and illumination correction, ensuring high-quality preprocessing of handwritten document images.

DocTr utilizes a transformer-based model to understand and correct distortions in document images. The model is trained on a large dataset of document images, learning to recognize and rectify common issues such as geometric distortions and uneven lighting. By analyzing the spatial relationships within the image, the transformer can accurately predict the necessary transformations to straighten the text and normalize the lighting, resulting in a clear and well-aligned image.

The process begins with the model taking an image of the handwritten document and generating a set of transformation parameters. These parameters include adjustments for geometric distortions, such as skew and curvature. The image is then processed using these parameters, producing a flattened and uniformly lit output. This advanced preprocessing technique significantly enhances the quality of the input image for subsequent OCR tasks, leading to more accurate recognition of handwritten text, an illustrative example of this process can be found in Figure 3.2 [46].

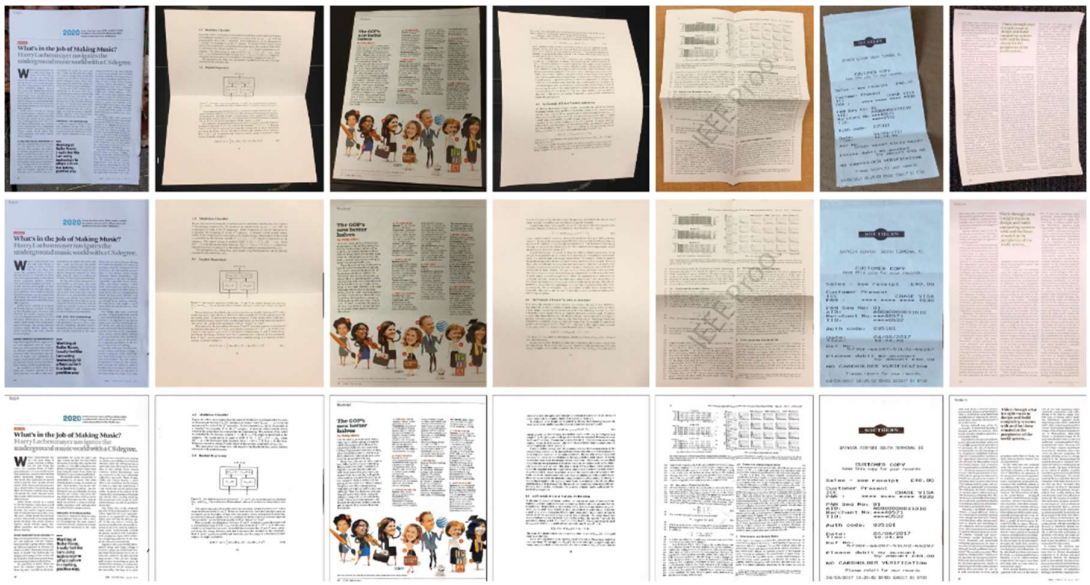


Figure 3.2: Examples from DocTr

By employing DocTr, we overcome the challenges posed by the naive approach and achieve a more robust preprocessing pipeline. This method does not require the entire paper to be visible or the background to be clearly different from the paper, making it suitable for a wide range of real-world scenarios. Additionally, the use of a transformer model allows for precise and efficient corrections, ensuring that our OCR system receives the best possible input for accurate text recognition.

3.2 Paragraph Segmentation

Effective OCR begins with a structured approach to text extraction and analysis. First, the text is split into paragraphs using line breaks and indentation. Each paragraph is then divided into lines, and each line into words, separated by spaces or punctuation. Finally, a detailed analysis at the sub-word level examines characters and features. This systematic breakdown allows OCR algorithms to accurately extract and interpret the document's content, supporting various applications.

3.2.1 Segmentation Based on Histogram Projections

Our system relies on the utilization of histogram projections to accurately extract textual information from document images. The first step in this technique is to obtain the vertical histogram of the white pixels within the image. Vertical histogram represents the distribution of pixel intensities or densities along the vertical axis of the document, providing valuable insights into the locations of text lines. This helps to identify the locations of individual characters or lines of text within a document image [50][51].

With the vertical histogram in hand, the next step is to calculate a suitable threshold. Our threshold was based on the weighted average of the histogram values. This is done by computing the vertical histogram of the image and then calculating the weighted average of the histogram by multiplying each histogram value with its corresponding row index (y-coordinate) and then dividing the sum of these products by the total sum of the histogram values.

This threshold helps identify the gaps or spaces between text lines, which are considered as the "valleys" in the histogram. By isolating these valley regions, the system can effectively segment the image into individual text lines for further processing as shown in Figure 3.3.

Once the text lines have been extracted, the algorithm then employs a pathfinding technique, to traverse the identified line regions from right to left. During this process, the system can discard any extraneous elements that fall outside the designated line areas, further refining the segmentation. We used the A* (A-star) algorithm as a pathfinding technique [52][53]. It is a type of informed search algorithm that uses heuristics to estimate the cost of reaching the goal from a given node. It is used to find the shortest or optimal path between two points in a graph or a grid-based environment by maintaining a set of nodes to be explored, known as the "open set," and a set of nodes that have already been explored, known as the "closed set." The algorithm starts by adding

the starting node to the open set and then repeatedly selects the node with the lowest estimated total cost (the sum of the actual cost to reach that node and the estimated cost to reach the goal) from the open set, moves it to the closed set, and then explores its neighbors.

Figure 3.3: Example of line segmentation

With the text lines now accurately segmented, we turned our attention to the horizontal dimension to extract words, utilizing a similar approach except using the horizontal histogram instead of vertical. This histogram projects the pixel distributions along the horizontal axis, enabling the identification of individual words within each line of text. By applying thresholding and pathfinding techniques, the system can effectively separate the words, preparing them for character-level recognition.

Finally, the segmentation process extends to the sub-word or island level, where each word is further broken down into its constituent characters or subcomponents. This granular analysis is crucial for accurate optical character recognition, as it allows the system to identify and interpret the individual features that make up the textual content. By systematically decomposing the text into these hierarchical components, the OCR algorithm can achieve a high degree of accuracy in extracting and interpreting the written information and Figure 3.4 shows how word segmentation works.

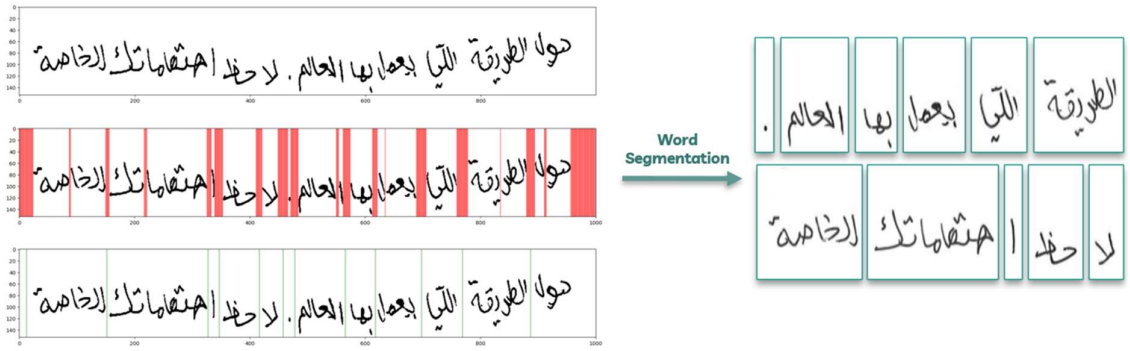


Figure 3.4: Example of word segmentation

Figure 3.5 Summarizes the system procedure in a diagram.

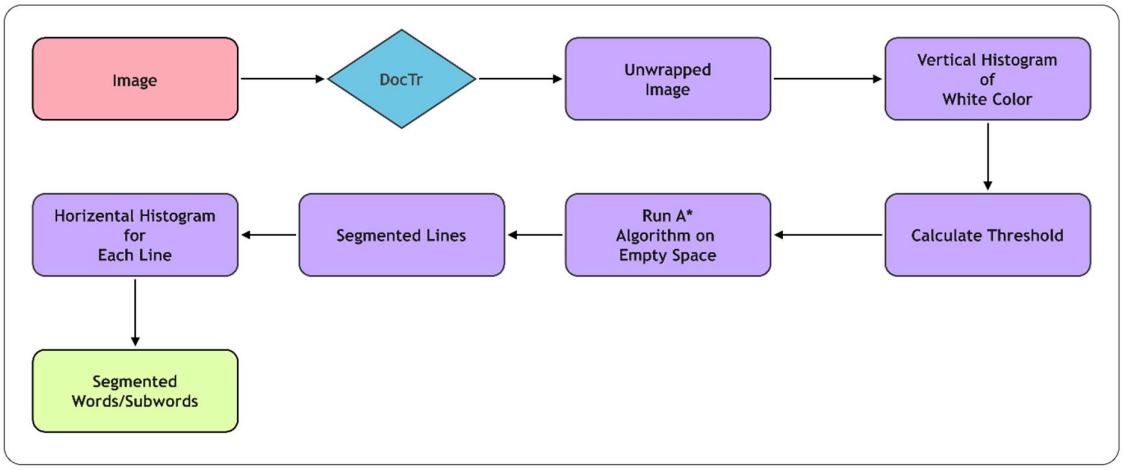


Figure 3.5: Diagram of histogram-based segmentation system

3.2.2 Issues with Histogram Projection Segmentation

During the preprocessing stage, we encountered two main challenges with the histogram projection-based segmentation approach. These problems are outlined as follows:

One major limitation is the inability to effectively handle skewed or slanted text lines. The vertical histogram, which is the foundation of this technique, relies on the assumption that text lines are oriented horizontally. However, in the presence of skewed text, the histogram peaks may not accurately represent the true line positions, leading to

inaccurate segmentation. This issue becomes particularly problematic when dealing with handwritten or scanned documents, where text lines can exhibit significant skewness due to various factors, such as uneven page orientation or the user's natural handwriting inclination.

The second challenge is the difficulty in determining the appropriate threshold for separating text lines and words. The method typically relies on identifying peaks in the vertical histogram to locate the text lines, but the selection of the threshold value can be a complex task, especially in the presence of varying text densities or irregular spacing between words. If the threshold is set too high, it may fail to detect some text lines, while a low threshold can result in over-segmentation, where individual words are incorrectly identified as separate lines. This problem is further exacerbated in documents with complex layouts, such as those with text interleaved with graphics or tables, where the histogram may not provide a clear distinction between text and non-text regions.

3.3 CRAFT: Character Region Awareness for Text Detection

CRAFT, or Character Region Awareness for Text Detection, is a model designed to detect text in images [54]. It operates by generating a heat map that highlights areas in the image where text is likely to be present (Figure 3.6). This heat map assists in identifying and localizing text within the image, making it a valuable tool for text detection in various contexts, including OCR for handwritten documents.

Figure 3.6: Sample Results of CRAFT

Once the text regions are identified using CRAFT, thresholding is applied to these regions to enhance the visibility of the text. Thresholding is a process that converts grayscale images into binary images by turning all pixels below a certain threshold value to black and all pixels above that threshold to white. This binarization simplifies further processing steps by reducing the complexity of the image and emphasizing the text areas.

After thresholding, histogram segmentation is employed to segment the image into individual words or sub-words. This involves analyzing the vertical and horizontal histograms of pixel intensities to detect gaps and boundaries between text regions. Specifically, repeated vertical columns of pixels are analyzed from right to left. Columns with a sum of zero indicate empty spaces, which are used as cut points to segment the text. This method effectively breaks down the text into manageable parts for further processing.

This method addresses both the issue of image angle and paragraph size effectively. The initial challenge was correcting the image angle to ensure all text lines were horizontal and properly aligned. Using CRAFT,

the model can detect text regions even if the image is taken at an angle, allowing for more flexible input conditions. Additionally, the segmentation approach helps manage varying paragraph sizes by precisely identifying text boundaries without relying on predefined templates or extensive preprocessing, Figure 3.7 shows the diagram of the CRAFT based system.

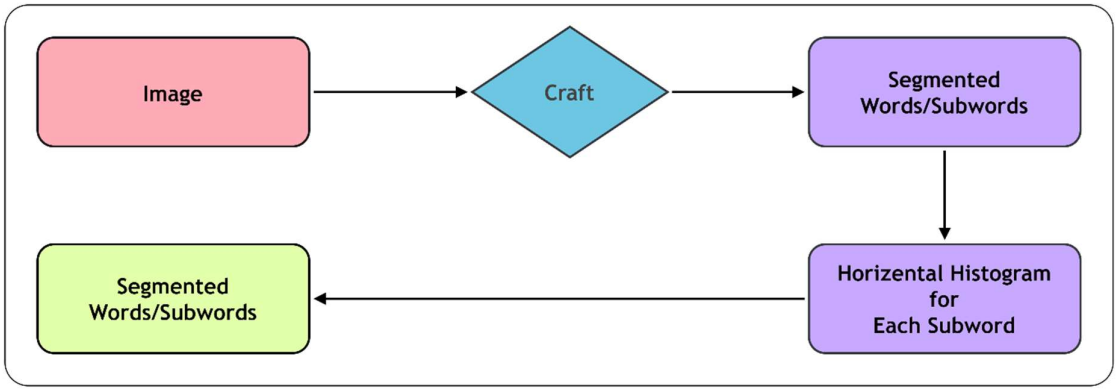
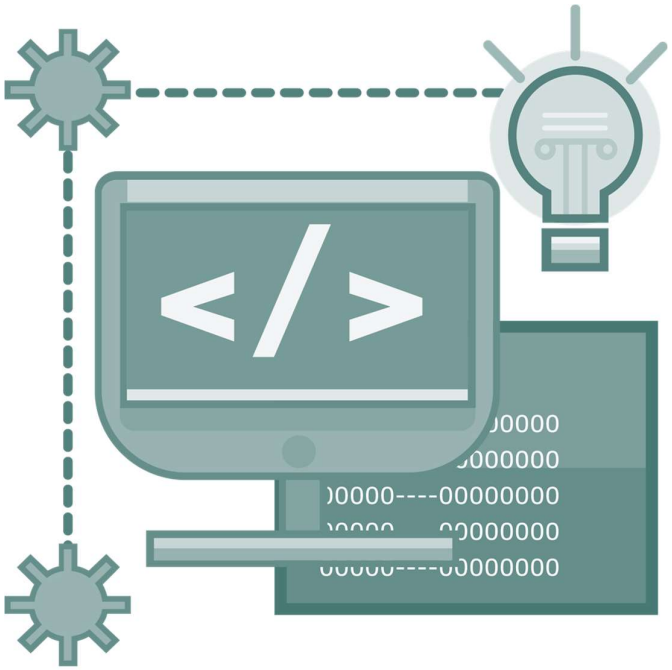


Figure 3.7: Diagram of the final segmentation system

Although applying CRAFT introduces an additional computational step, the overall accuracy and robustness of the OCR system are significantly improved, making it a worthwhile compromise.

In summary, the integration of the CRAFT model significantly enhances the preprocessing for Arabic handwritten OCR. By accurately detecting text regions and handling image angles, CRAFT ensures proper text alignment. The histogram segmentation technique further refines the input by precisely cutting the smallest unit of text possible. This preprocessing pipeline produces clean, well-segmented text images, perfectly primed for the recognition system. Overall, these steps ensure higher accuracy and reliability in the OCR outputs.

CHAPTER 4



SYSTEM DEVELOPMENT

CHAPTER 4 SYSTEM DEVELOPMENT

With the system design completed, the next crucial step is the implementation phase. This stage involves integrating various technologies and algorithms tailored to handle the complexities of Arabic script. It requires meticulous attention to detail, from configuring image processing libraries like OpenCV for segmentation to fine-tuning deep learning models for accurate character recognition. Collaborative efforts between experts are essential to refine the system iteratively and ensure optimal performance across diverse handwritten texts and input conditions. Rigorous testing and validation will pave the way for a robust OCR solution tailored to Arabic handwritten documents.

4.1 Segmentation System

We utilized OpenCV for the initial segmentation system of our Arabic Handwritten OCR project. The system was meticulously programmed to segment text lines and words within an image using the aforementioned image processing techniques.

And then, to manage the complex requirements of text detection, we implemented CRAFT (Character Region Awareness for Text detection) via the KerasOCR library [55][56], which includes a TensorFlow-compatible implementation of CRAFT able to segment words or sub-words efficiently as shown in Figure 4.1.

Figure 4.1: Segmentation Results of CRAFT

4.2 Recognition System

The recognition system in our Arabic Handwritten OCR project is designed to accurately transcribe Arabic handwritten text into machine-readable format. Leveraging machine learning models and deep neural networks, the system analyzes input images of handwritten Arabic text to identify and convert characters into digital text. This process involves training models on labeled datasets to recognize the intricate variations and styles inherent in Arabic handwriting, enabling the OCR system to achieve high accuracy in text recognition tasks.

4.2.1 Dataset

In our search for a pre-labeled dataset of Arabic handwritten text, we found that no suitable dataset was publicly available.

Most of the publicly available datasets seem to only contain characters and their different shapes. We needed to include the largest amount of words possible.

Consequently, we had to create our own dataset derived from the publicly available KHATT dataset [57][58].

The KHATT dataset consists of several paragraphs like the one demonstrated in Figure 4.2, the whole paragraph is labeled but due to our segmentation system, each paragraph gets segmented in a different way, so we had to label the new data manually.

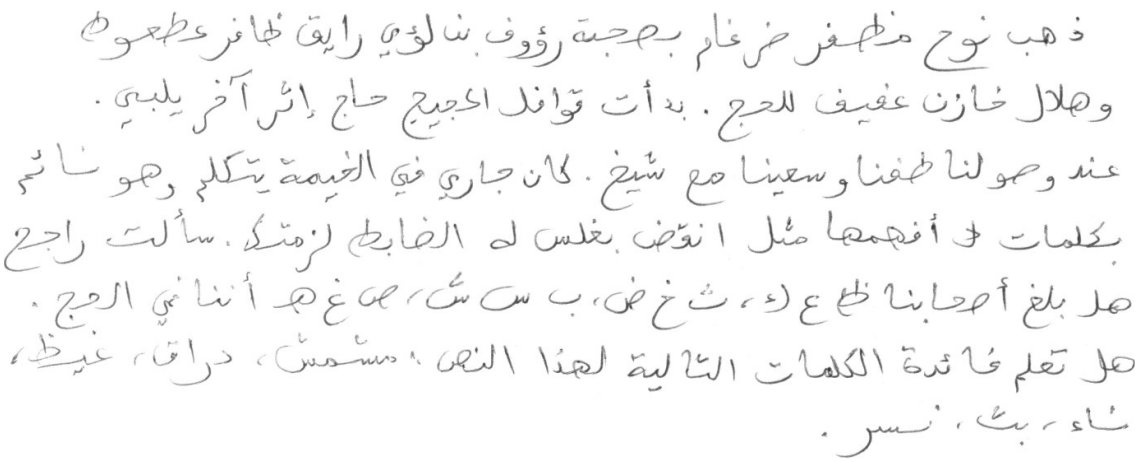


Figure 4.2: A Sample of a Full Paragraph from KHATT Dataset

4.2.1.1 Figuring Out an Approach to Data Labeling

We were faced with the challenge of labeling about 200,000 samples, which was a daunting task to complete manually within a month. As a result, we needed to find an application that would enable us to collaborate on the data labeling process from any location.

After exploring several options, we found that the available applications were either too complex, designed for broader data labeling purposes such as image recognition and segmentation, or were commercially oriented with expensive licenses and long-term commitments. Consequently, we made the decision to create our own labeling application that would support collaboration, streamline data entry, and facilitate quick data review.

We quickly drafted a prototype UI (see Figure 4.3) that resembles a spreadsheet, featuring images on the left and a text input field on the right.

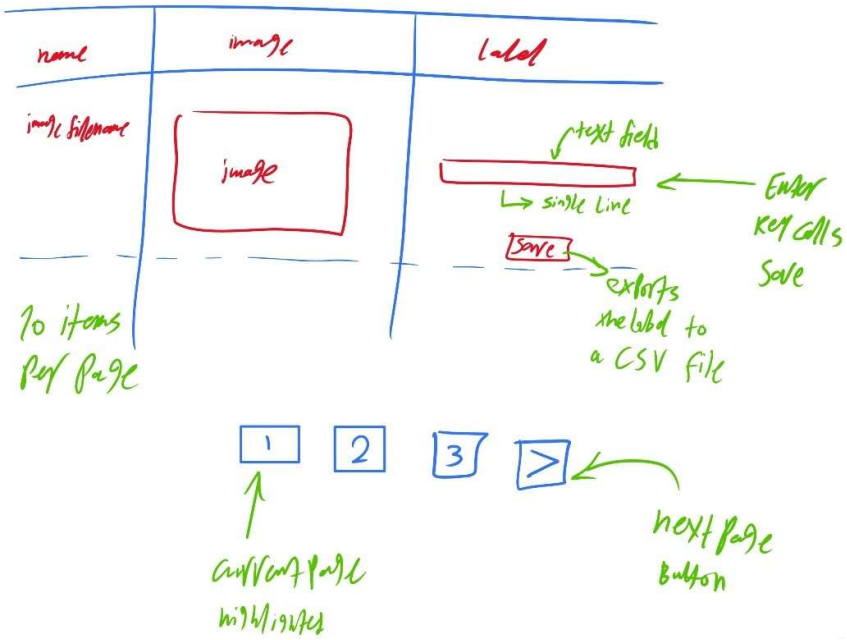


Figure 4.3: A draft of labeling app UI

4.2.1.2 Implementing The Labeling App:

The goal of this application is to help us in the process of data labeling which is used for training the model. The idea of this application is to display a set of images from the KHATT dataset, then we add the appropriate label for each image, and by pressing enter that label is saved in our database.

We used HTML, CSS and JavaScript for the frontend implementation and PHP for backend implementation for our web application.

4.2.1.2.1 PHP

PHP (Hypertext Preprocessor) is a widely used, open-source, server-side scripting language specifically designed for web development. PHP code is executed on the server, generating HTML that

is then sent to the client. The client receives the results of the script but remains unaware of the underlying code [59].

PHP is supported by a large community of developers who continuously contribute to its development and maintenance, PHP benefits from the advantages of being open source. Developers have access to the source code and can modify and customize it to suit their needs. Additionally, PHP supports object-oriented programming (OOP), allowing developers to create reusable code and streamline development processes.

One of PHP's strengths is its low learning curve. Its simple syntax, similar to C and other popular programming languages, allows developers to quickly learn and start building web applications.

A key feature of PHP is its rich set of built-in functions and libraries for various tasks. Whether working with databases, handling files, or processing forms, PHP offers functions that can significantly speed up development.

PHP provides robust features and functions for efficient data storage and retrieval. PHP supports a wide range of databases, including MySQL, PostgreSQL, SQLite, and Oracle. It offers built-in functions like mysqli and PDO (PHP Data Objects) that simplify connecting to databases, executing queries, and ensuring secure and reliable data access. PHP's seamless integration with databases through its OOP capabilities allows developers to create classes and objects representing database tables, making it easier to interact with data and perform CRUD (Create, Read, Update, Delete) operations.

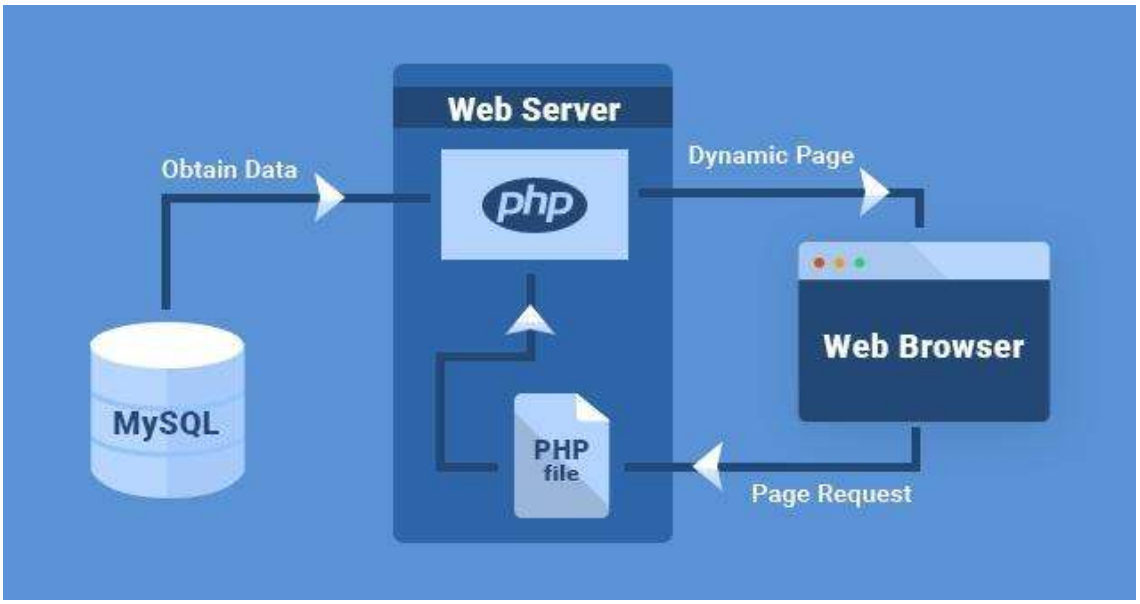


Figure 4.4: PHP Database Connection Diagram

4.2.1.2.2 Image-Tag App Details

Our app efficiently loads and displays images sourced from a specified directory, presenting them in a user-friendly interface tailored for seamless labeling. This interface allows users to input accurate labels directly, with the entered data promptly saved into the database for streamlined data management.

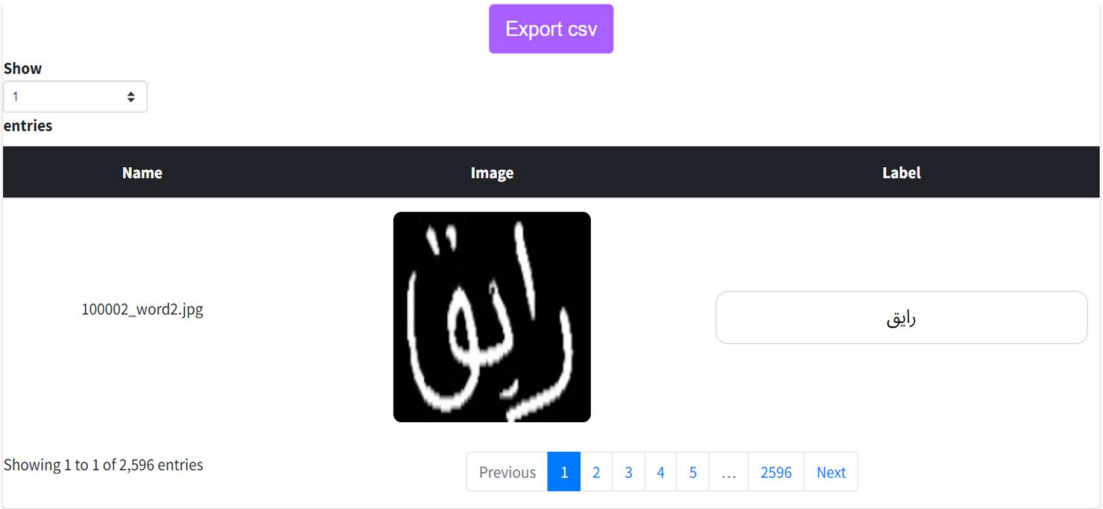


Figure 4.5: Main Page of Labeling APP

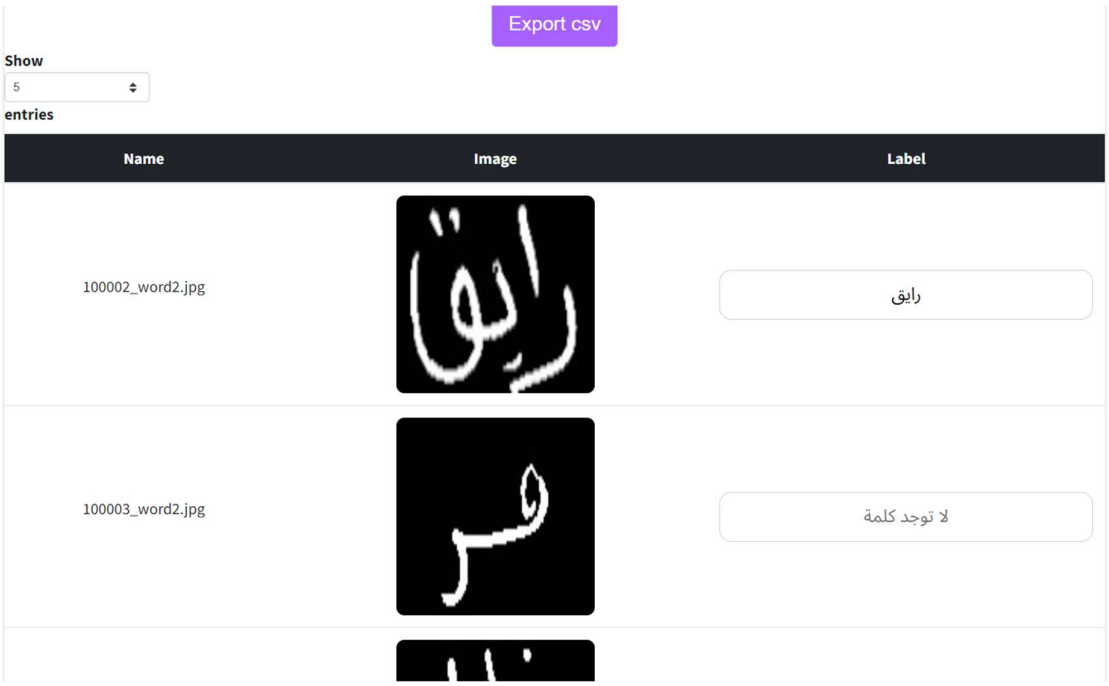


Figure 4.6: Main Page of Labeling APP

The app helps in manual data Labeling as Users are empowered to provide precise labels for each image via the intuitive interface. This manual labeling process ensures data accuracy and reliability, which in turn significantly enhances the model's precision and overall effectiveness.

To enhance user experience, the app supports a pagination feature that organizes image viewing into manageable subsets per page. This feature simplifies the labeling process by allowing users to smoothly navigate between pages, facilitating efficient viewing and labeling of a larger image set.

An essential feature of our app is its capability to export labeled data into a CSV file. This export functionality is instrumental for model training, providing a structured and accessible dataset that contributes to the optimization of the training process.

4.2.1.2.3 Benefits Of Image-Tag App in Our Project:

Improved Data Quality: Manual labeling ensures accurate and reliable data, leading to enhanced model precision and effectiveness.

Streamlined Training Process: The program creates a ready-to-use labeled dataset, reducing time and effort in data preparation and optimizing model training.

Organized Data Management: Storage of labels in a database and export functionality to CSV maintains an organized system for managing labeled data, enhancing data management efficiency.
In essence, this program is a pivotal component in developing an effective and robust model. The meticulous labeling process and structured data management it offers are foundational elements in achieving success in machine learning endeavors.

4.2.1.3 Tagger وسّام

To improve our web application, we later rewrote it from scratch and added user login functionality to enable secure access to our app. This login functionality includes creating user accounts using authentication rules such as passwords.

Application concurrent work support was also a crucial thing, allowing multiple users to collaborate on the same project simultaneously with features such as real-time updates and version control. We also improved the user interface for the aim of simplifying the user interface and improving ease of use and adoption. We also used the dark blue theme for the text readability.

These improvements make the application more powerful, secure, and easy to use, meeting the needs of individual and group workflows.
This is what the “Tagger وسّام” application aims to achieve, and through it we were able to create a dataset that helped in developing our model.

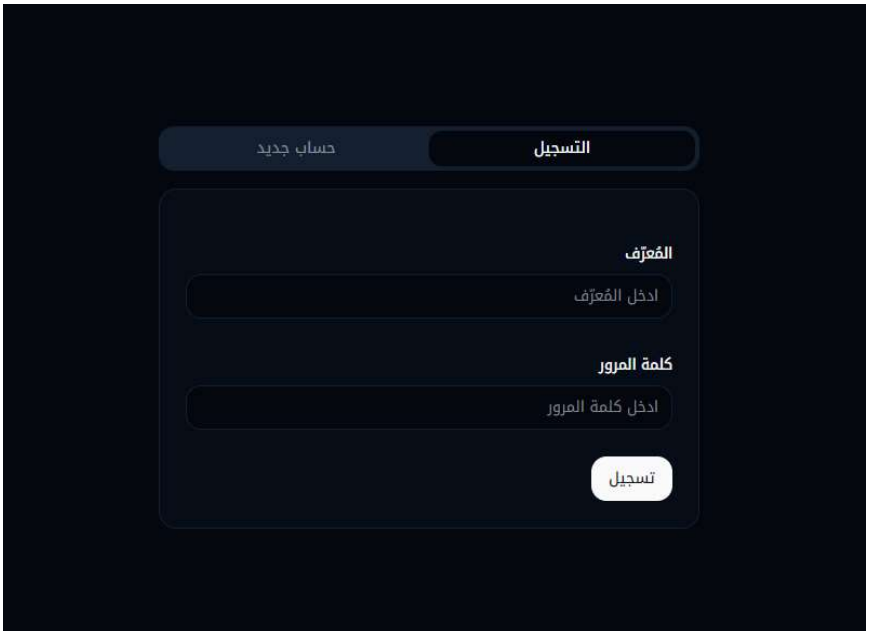


Figure 4.7: Login page of Tagger

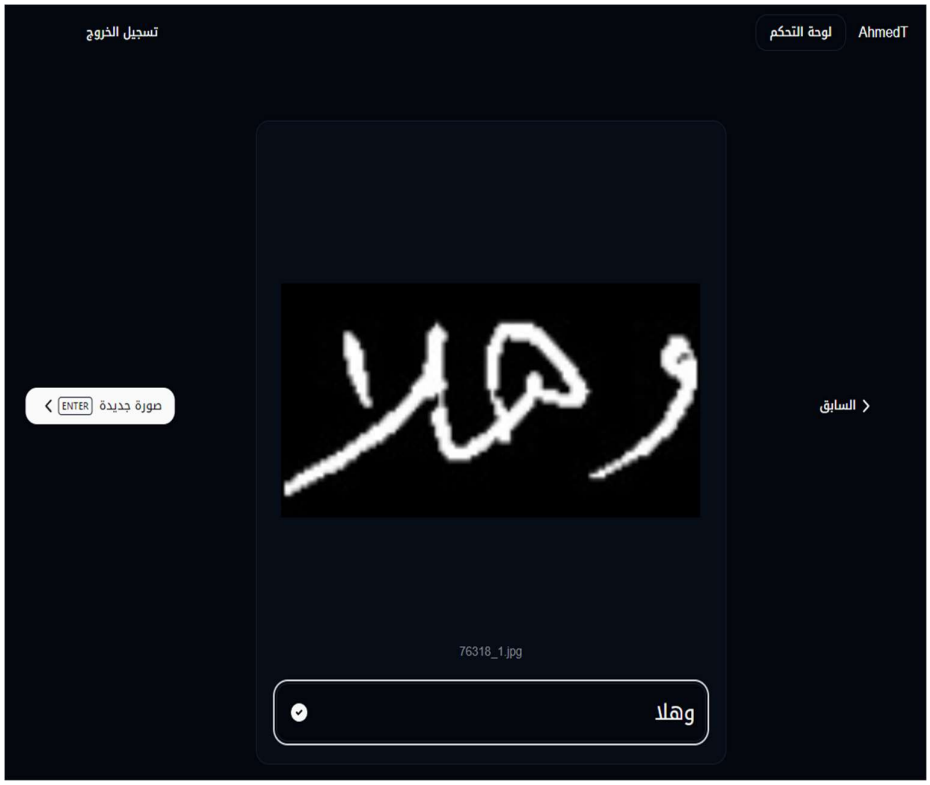


Figure 4.8: Main page of Tagger

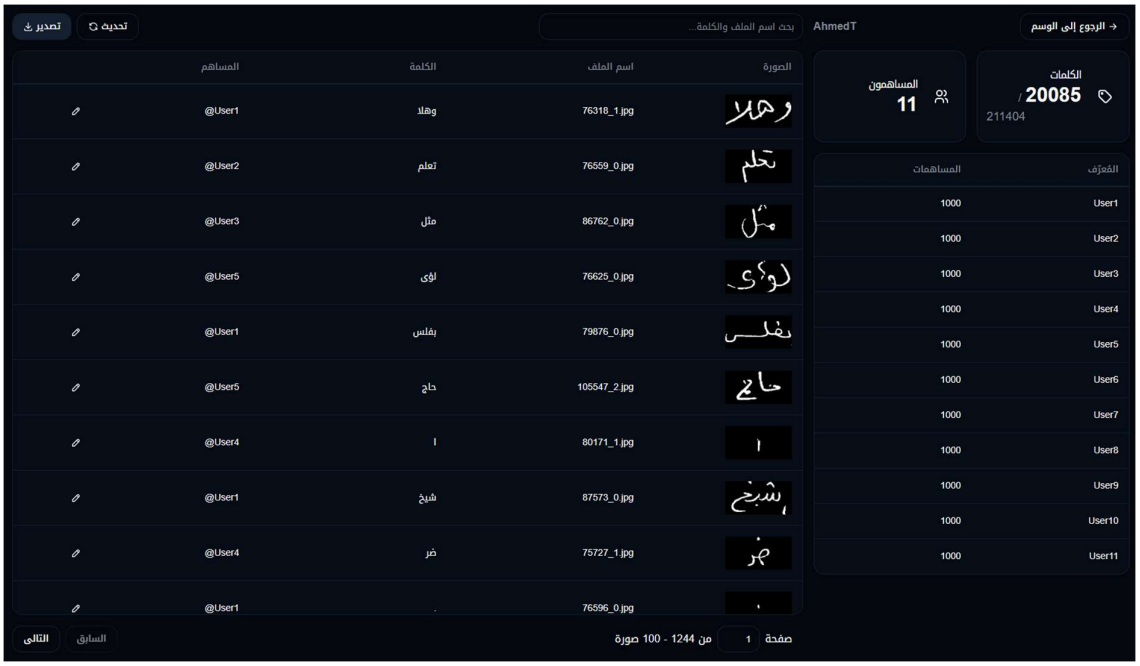


Figure 4.9: Dashboard of Tagger

4.2.2 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are a fascinating and powerful type of machine-learning model inspired by the structure and function of the human brain [60]. ANNs mainly consist of 3 main layers which are the Input Layer, Hidden Layers, and Output Layer as shown in Figure 4.10. Let's delve deeper into the intricacies of these networks

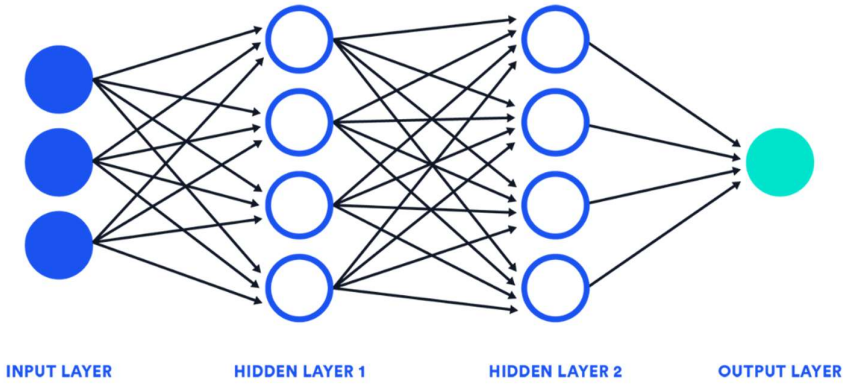


Figure 4.10: The General Architecture of ANNs

4.2.2.1 Convolutional Neural Network (CNN)

A convolutional neural network (CNN) shown in Figure 4.11 [61] is a category of machine learning model [62], namely a type of deep learning algorithm well suited to analyzing visual data [63][30]. They are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs classifying and dealing with them. The general layers of a CNN are:

- Convolutional layers & Pooling layers which are used for feature extraction of the images
- Fully connected (FC) layer which provides the output or the classification

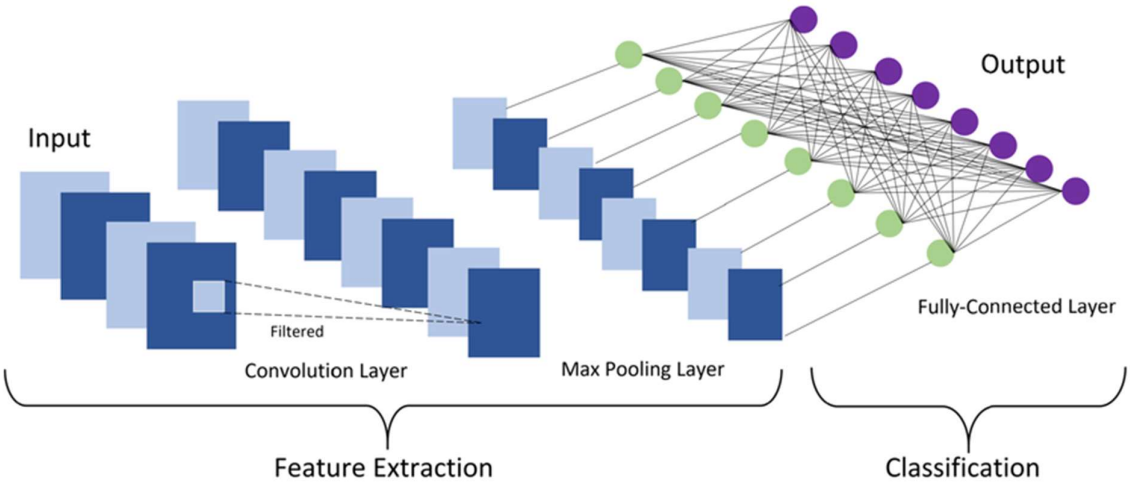


Figure 4.11: Demonstration of CNN Layers and Architecture

4.2.2.2 Recurrent Neural Network (RNN)

A recurrent neural network (RNN)[31] is a type of artificial neural network which uses sequential data or time series data [64][65][66]. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), and speech recognition [67][68]. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output.

4.2.2.3 Usage of CNN and RNN in OCR

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two fundamental architectures in deep learning, designed to tackle different types of data and tasks [69][70].

4.2.2.3.1 CNNs for OCR

Due to their powerful feature extraction capabilities, CNNs are highly effective at recognizing individual characters within handwritten text. They can discern the varied shapes and strokes that constitute handwritten characters, making them suitable for segmenting and recognizing characters in isolation.

4.2.2.3.2 RNNs for OCR

RNNs, with their ability to process sequences, excel in understanding the flow and context of handwriting. This is particularly beneficial for cursive writing or scripts where characters are connected, and the context is necessary for accurate recognition. RNNs, especially variants like LSTMs or GRUs, can better handle the temporal dependencies inherent in handwritten text as shown in Figure 4.12 [71].

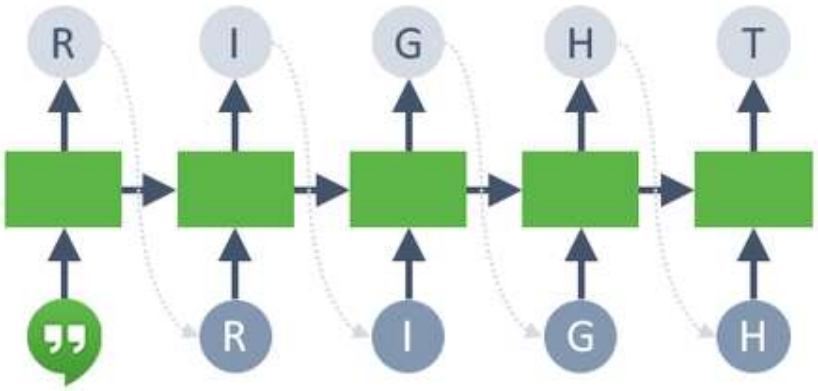


Figure 4.12: An Example of how RNNs Handle Text Sequences

4.2.2.4 CTC Loss Layer

CTC is an algorithm employed for training deep neural networks in tasks like speech recognition and handwriting recognition [72]. CTC facilitates end-to-end training of neural networks for sequence-to-sequence tasks without the need for explicit alignment annotations. It demonstrates resilience to labeling errors or inconsistencies within the training data by implicitly learning sequence alignments [73]. The algorithm is applicable across a diverse array of use cases.

It Calculates loss between a continuous (unsegmented) time series and a target sequence. It does this by summing over the probability of possible alignments of input to target, producing a loss value which is differentiable with respect to each input node. The alignment of input to target is assumed to be “many-to-one”, which limits the length of the target sequence such that it must be the input length [74].

4.2.2.5 Requirements For Implementing the System

Implementing the recognition system required meticulous planning and execution, focusing on the hardware and software components necessary for efficient operation. Our choice of hardware was dictated by the need for substantial computational power, primarily due to the large datasets and complex neural network models involved in Arabic handwriting recognition.

Initially, we considered using Google Colab for training our models. However, we encountered several limitations that made this option impractical:

1. Internet Dependence: Google Colab requires a constant internet connection, which isn't always available. This limitation posed significant challenges for our development process, especially in environments with unstable connectivity.

2. Data Upload Constraints: The datasets we used were substantial in size, making it time-consuming and inefficient to upload them to Google Colab. Handling such large volumes of data locally was more feasible.
3. Free Tier Limitations: The free tier of Google Colab offers limited usage time, which often interrupts our training sessions. This restriction necessitated a more reliable solution to ensure continuous and uninterrupted model training.

Given these challenges, we opted to work locally, which required a powerful GPU. Our initial setup included a GPU with 6GB of VRAM, but this proved insufficient for our needs. We needed a GPU with at least the capabilities of the Nvidia Tesla T4, which is provided in the free tier of Google Colab. After evaluating various options, we decided on the Nvidia RTX 3060, which provided the necessary computational power and memory capacity to handle our training requirements efficiently.

To implement the recognition system, we used Python and TensorFlow [75]. These tools were selected because of their efficiency and the robust libraries available for deep learning. TensorFlow offered extensive support for neural network implementation and training, making it an ideal choice for our project.

4.2.2.6 Nvidia’s RTX 3060

The Nvidia RTX 3060 played a crucial role in the development and training of our Arabic handwriting recognition system. With 12GB of VRAM [76], it offered the necessary memory capacity to handle large datasets and complex neural network models without running into memory limitations. The GPU's architecture allowed for efficient parallel processing, significantly reducing the training time compared to our previous hardware setup.

This upgrade was essential for achieving the performance and accuracy required for our system. The additional VRAM enabled us to experiment

with larger batch sizes and more complex models, which improved the recognition accuracy of our system. The RTX 3060's compatibility with TensorFlow and other machine learning frameworks ensured a smooth integration into our development pipeline, allowing us to leverage its full potential for our project.

In summary, the integration of the CRAFT model significantly enhances the preprocessing for Arabic handwritten OCR. By accurately detecting text regions and handling image angles, CRAFT ensures proper text alignment. The histogram segmentation technique further refines the input by precisely cutting the smallest unit of text possible. This preprocessing pipeline produces clean, well-segmented text images, perfectly primed for the recognition system. Overall, these steps ensure higher accuracy and reliability in the OCR output

CHAPTER 5

AI SYSTEM

CHAPTER 5 AI SYSTEM

Developing machine learning models for handwriting recognition involves several key stages. Initially, a suitable dataset is collected and prepared, including data cleaning, labeling, and augmentation to enhance generalization. Next, various model architectures are tested to identify the best fit for the task, optimizing configurations and hyperparameters. Training the model is an iterative process requiring careful tuning and validation. Techniques like transfer learning can further improve performance by leveraging pre-trained models. Throughout development, performance metrics are continually evaluated to ensure the model meets accuracy and reliability standards and Figure 5.1 [77] shows the relation between the training data and the model.

Figure 5.1: Model Development Process

5.1 Data Collection & Preparation

Developing a robust handwriting recognition model begins with the meticulous collection and preparation of a comprehensive dataset. This process involves curating a diverse array of samples that

encompass various handwriting styles and nuances essential for training our models effectively. For our endeavor, we focused on integrating datasets such as KHATT for words and sub-words, and AHAWP alongside Arabic Handwritten Characters as Alphabetical Arabic Characters datasets. Each dataset was meticulously cleaned, accurately labeled, and split into training and validation sets to optimize training and evaluate model performance. To enhance the dataset's variability and prepare our models for real-world scenarios, we employed advanced techniques like data augmentation. These efforts were instrumental in ensuring our models could generalize well and achieve high accuracy in recognizing Arabic handwriting, as evidenced by the results depicted in Figure 5.2.

5.1.1 Dataset Collection and Preparation

We gathered datasets including KHATT [57], AHAWP [78], and Arabic Handwritten Characters to cover diverse handwriting styles.
Data was split into training (80%) and validation (20%) sets to optimize model training and evaluate performance [79].

Each dataset underwent rigorous cleaning and accurate labeling to ensure high-quality input for training.

5.1.2 Data Augmentation Techniques

We Implemented image rotation by +10 and -10 degrees to enhance dataset diversity. This step was crucial in preventing overfitting by exposing the model to Increased variability of training samples [80].

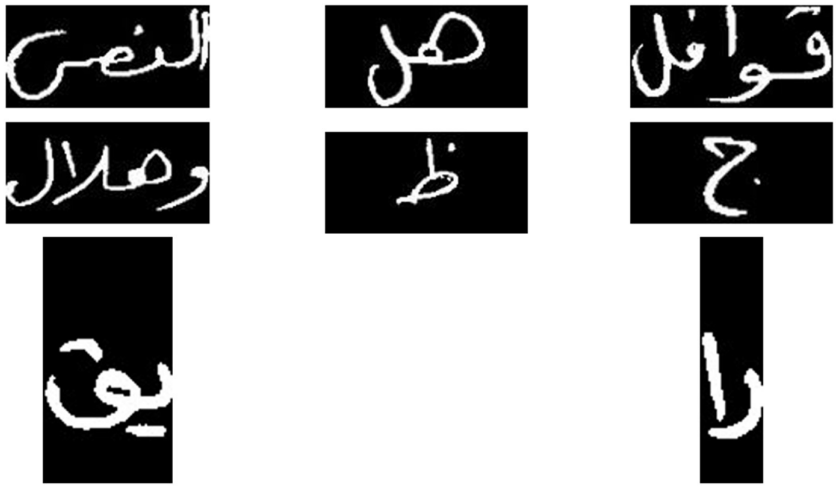


Figure 5.2: Dataset Samples

5.2 Defining The Training Process

The training process utilizes the prepared data to teach the model to make accurate predictions by iteratively adjusting its parameters to minimize error [81]. During this phase, the model undergoes multiple cycles of learning, where it continuously refines its internal parameters to better capture the patterns and relationships within the data. This process involves feeding the data into the model, calculating the loss or error between the predicted and actual values, and then updating the model's parameters to reduce this loss. Through repeated iterations, the model gradually improves its performance, learning to generalize from the training data to make accurate predictions on unseen data. Several crucial key hyperparameters are adjusted during this phase mentioned below.

5.2.1 Batch Size

The batch size in machine learning determines the number of training samples used in one iteration. Smaller batch sizes provide more frequent updates, which can introduce noise into the learning process but can help the model to learn more detailed patterns [82]. On the other

hand, larger batch sizes offer more stable updates and require more memory, often leading to more efficient training.

5.2.2 Epochs

Epochs indicate how many times the entire dataset is passed through the model during training. Increasing the number of epochs allows the model to learn more thoroughly, but too many epochs can lead to overfitting, where the model performs well on the training data but poorly on new, unseen data [83].

5.2.3 Optimizer

The optimizer is the algorithm that governs how the model's parameters are updated during training. Common optimizers include Adam and Stochastic Gradient Descent (SGD). The choice of optimizer significantly affects the speed and quality of convergence to the minimum of the loss function, impacting overall model performance [84].

5.2.4 Learning Rate

The learning rate influences the step size at each iteration while moving toward the minimum of the loss function. A higher learning rate can lead to faster convergence but might overshoot the minimum, causing instability. Conversely, a lower learning rate ensures more precise convergence but may take longer and could get stuck in local minima [83]. Proper tuning of these hyperparameters is essential for achieving a well-performing and generalized model.

5.3 Defining Model Evaluation

Model evaluation metrics are essential for assessing the performance and effectiveness of machine learning models, particularly

in tasks like character recognition. Key metrics used include *accuracy*, *precision*, *recall*, and the *character error rate (CER)* [85].

The CER is calculated using equation 1

$$CER = (i + s + d) / n \quad (1)$$

Here, *i* denotes the number of insertion errors where extra characters are predicted, *s* represents substitution errors where incorrect characters are predicted, and *d* indicates deletion errors where characters in the ground truth are missing from the prediction. The total number of characters evaluated is denoted by *n*. Additionally, CER is closely related to the Levenshtein distance, which measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another as shown in Figure 5.3. By considering these factors, the CER provides a comprehensive measure of how well a model performs in generating correct character sequences. A lower CER indicates higher accuracy, as it signifies fewer errors in character prediction relative to the ground truth, crucial for evaluating models in handwriting recognition, OCR, and other applications requiring precise character transcription.

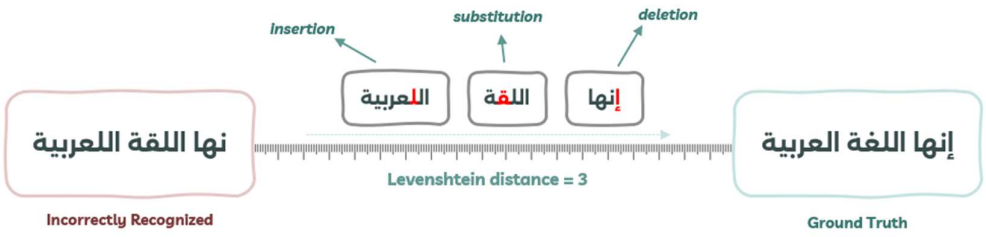


Figure 5.3: Demonstration of CER Metric

5.4 Dealing with Characters' Sequence: Bi-LSTM

Long Short-Term Memory networks (LSTMs) Shown in Figure 5.4 [86] are a specialized type of recurrent neural network (RNN) designed

to address the vanishing gradient problem and effectively capture long-term dependencies in sequential data.

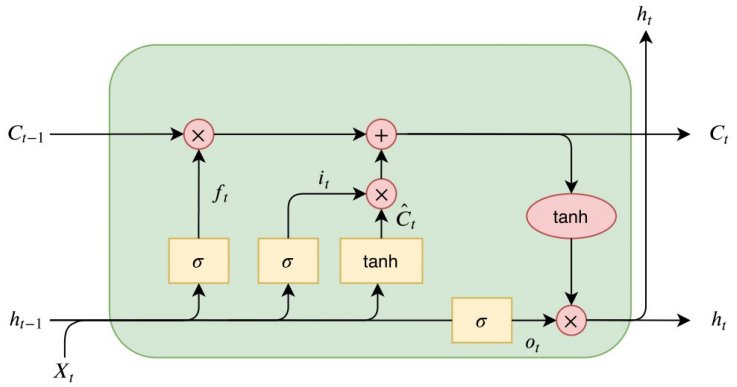


Figure 5.4: Diagram of A Single LSTM Unit

LSTMs utilize a memory cell and gate mechanisms to regulate the flow of information through the network over time. LSTMs include **Input (*Xt*):** The input at time step t, **Memory Cell (*Ct*):** Maintains information over long sequences, preventing the vanishing gradient problem in traditional RNNs, **Hidden State (*ht*):** Carries information from the current time step to the next. It is used along with the next input to compute the gates and cell state in the following time step, **Forget Gate (*ft*):** Controls what information should be discarded from the cell state, **Input Gate (*it*):** Modulates the input information to update the cell state, and **Output Gate(*ot*):** Produces the output based on the current cell state, ensuring relevant information is passed to the next layer or output.

BiLSTMs are a type of recurrent neural network (RNN) designed to capture dependencies in sequences by processing information in both forward and backward directions simultaneously as shown in Figure 5.5 [87]. Unlike traditional LSTMs that only consider past context, BiLSTMs incorporate future context as well, enhancing their understanding of sequence data.

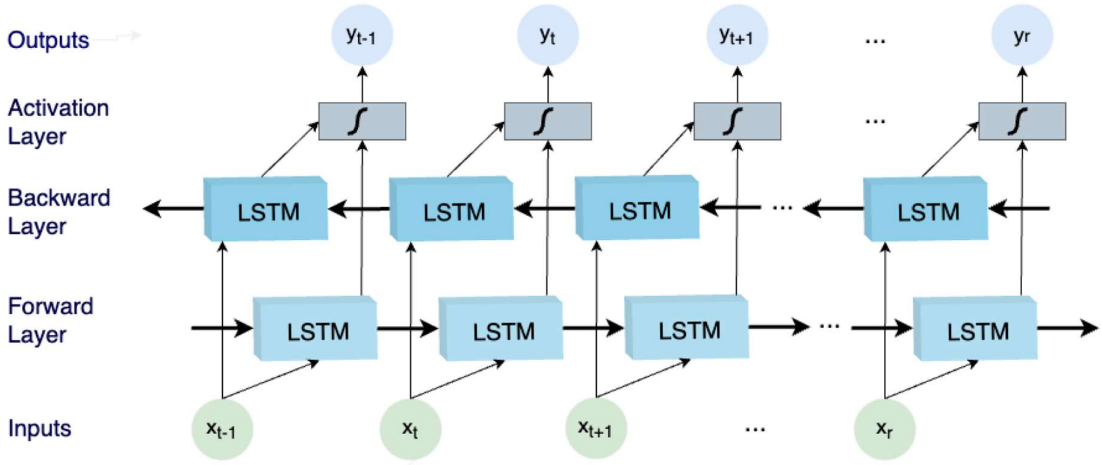


Figure 5.5: Bi-LSTM Architecture

The BiLSTM (Bidirectional Long Short-Term Memory) architecture consists of two LSTM layers: one processes sequences in a forward direction, and the other processes them in a backward direction. Each LSTM unit computes its hidden state based on the input, previous hidden state, and previous cell state. This dual-layer approach allows the network to capture dependencies from both past and future contexts within the sequence, enhancing its understanding of the data.

The mathematical formulation of BiLSTMs utilizes sigmoid and tanh activation functions to control the flow of information through the network. These activation functions facilitate selective forgetting and updating of information over time, enabling the network to maintain long-term dependencies within sequences. This capability is crucial for tasks that require a nuanced understanding of sequential data, as it allows the network to effectively handle and learn from complex patterns.

In handwriting recognition, the BiLSTM architecture is ideal for modeling character sequences in words, where the context from both preceding and succeeding characters influences interpretation. This makes it particularly effective for recognizing Arabic handwriting, as it can capture the intricate relationships between characters within words. By leveraging the bidirectional processing capability, BiLSTMs enhance the accuracy and robustness of handwriting recognition systems.

Figure 5.6 shows the initial design of the recognition system

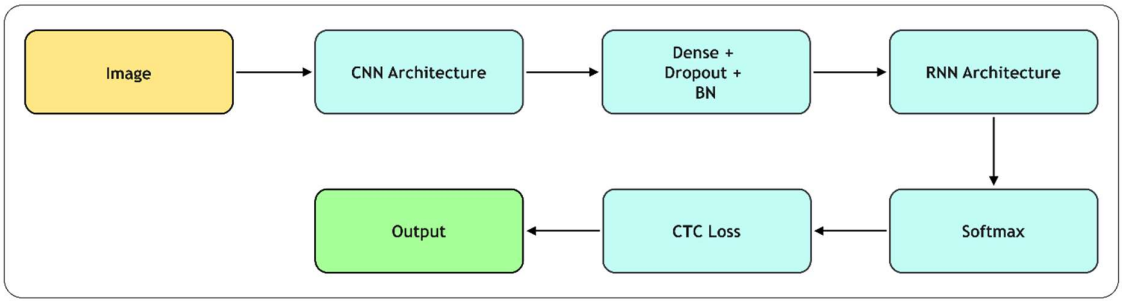


Figure 5.6: Diagram of the recognition system

With this, the final system becomes as Figure 5.7.

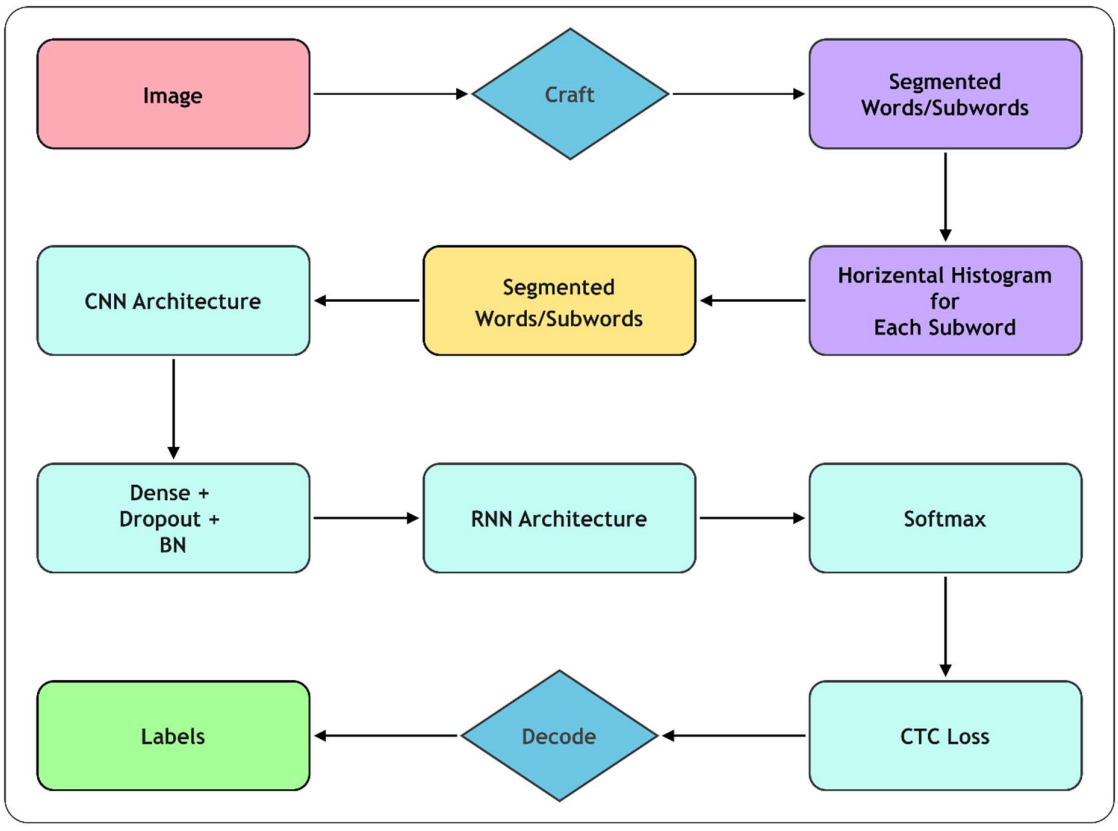


Figure 5.7: Diagram of entire system

CHAPTER 6

APPLICATION

CHAPTER 6 APPLICATION

6.1 Overview

In the development of our application for Arabic Handwritten OCR, we adopted a robust architecture to ensure efficiency and scalability. The core of our system is built on a front-end and back-end model, which allows us to separate concerns between the user interface and the computational heavy lifting. This separation is crucial for handling the intensive tasks involved in OCR, such as image processing and text recognition, which require significant computational resources. By offloading these tasks to a powerful server, we ensure that the application remains responsive and performs optimally even on devices with limited processing power. This chapter delves into the specifics of our application's architecture, the technologies used, and the rationale behind our design choices.

6.2 Frontend and Backend Model

In software engineering, the terms frontend and backend (sometimes written as back end or back-end) refer to the separation of concerns between the presentation layer (frontend), and the data access layer (backend) of a piece of software, or the physical infrastructure or hardware. In the client–server model, the client is usually considered the frontend and the server is usually considered the backend, even when some presentation work is done on the server itself.

We employed a front-end and back-end model in developing the application to offload the heavy computational tasks from the user's device to a powerful server. This approach ensures that the application remains responsive and efficient, even on devices with limited processing power. By handling intensive tasks like image processing and text recognition on the server, we can leverage more robust hardware and advanced algorithms, resulting in faster and more accurate OCR

performance. This division of labor allows the front-end to focus on providing a smooth user experience while the back end manages the complex computations, ensuring that our application is both user-friendly and highly effective.

6.2.1 Frontend

In the Frontend, we designed a web application which can be used by the user. It works by uploading an image that has the text which will be converted to a digital text by our system.

The website was created using HTML5, CSS3, JavaScript, and React JS frontend framework.

6.2.1.1 Native Application

A native application is a software program designed specifically for use on a particular platform or device. Unlike web applications that run in a browser, native applications are developed using platform-specific programming languages and tools. This allows for better performance, integration with device hardware, and a more seamless user experience.

We decided against using native applications because they are typically limited to a single platform, requiring separate versions for iOS, Android, and other operating systems, which significantly increases development time and costs. Moreover, native applications often demand powerful resources and frequent updates to maintain compatibility with new operating system versions and devices. This results in higher maintenance efforts and resource allocation, making native applications less efficient for our project needs.

6.2.1.2 Web Technology

Web technology refers to the tools, software, and programming languages used to create and maintain websites and web applications.

It encompasses a wide range of technologies and frameworks that work together to deliver content and functionality on the internet. Web technologies which we used are HTML, CSS, JAVASCRIPT, web browsers and frameworks like REACT [88].

6.2.1.2.1 HTML / CSS / JS

HTML, CSS, and JavaScript are the three foundational technologies used to create and design websites.
HTML (Hyper Text Markup Language): The standard language for creating web pages and structuring content on the internet. HTML defines the structure and layout of a webpage using tags.

CSS (Cascading Style Sheets) is used to enhance the presentation and layout of HTML elements. It allows developers to control the design aspects of a webpage, such as colors, fonts, spacing, and positioning. CSS is important for creating visually appealing and responsive websites.

JavaScript is a programming language that enables interactive and dynamic functionality on a webpage. It can be used to create animations, form validation, interactive maps, and other user interactions. JavaScript is essential for making websites more engaging and user-friendly.

6.2.1.2.2 React

React (also known as React.js or ReactJS) is a popular JavaScript library for building user interfaces. Developed by Meta (formerly Facebook) [89], React allows developers to create dynamic, interactive, and fast web applications [90].

One of the key features of React is its component-based architecture. Components are reusable building blocks that represent a part of the user interface. Each component encapsulates its own logic and state,

making it easier to manage and update different parts of the UI [91]. Components can be nested within each other to create complex interfaces.

React uses a virtual DOM (Document Object Model) to efficiently update and render components. When the state of a component changes, React only updates the necessary parts of the DOM, resulting in faster performance and improved user experience. This approach helps to minimize unnecessary re-renders and optimize the rendering process.

React also allows developers to create declarative and composable UI components. With React, developers can easily compose complex UI elements by combining smaller components, which helps in writing clean, maintainable, and modular code [92].

We decided to name our app "ضاد الرقمية" as it encapsulates how our app work with the Arabic language in a digital world, Figure 6.1 shows the home page design of our app.

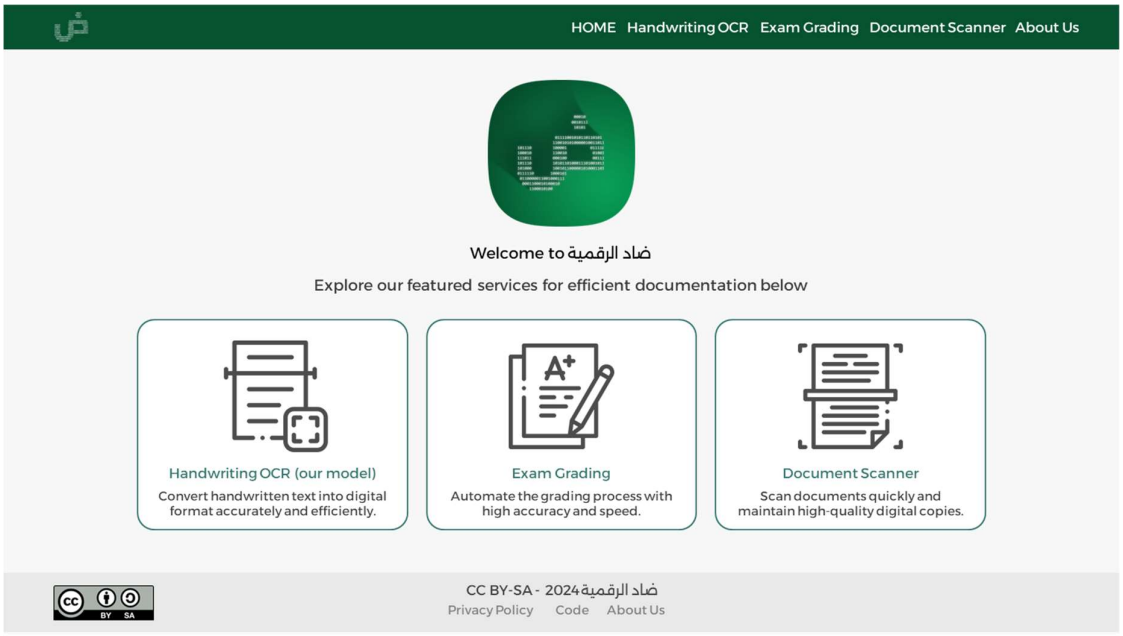


Figure 6.1: Screenshot of Main Page

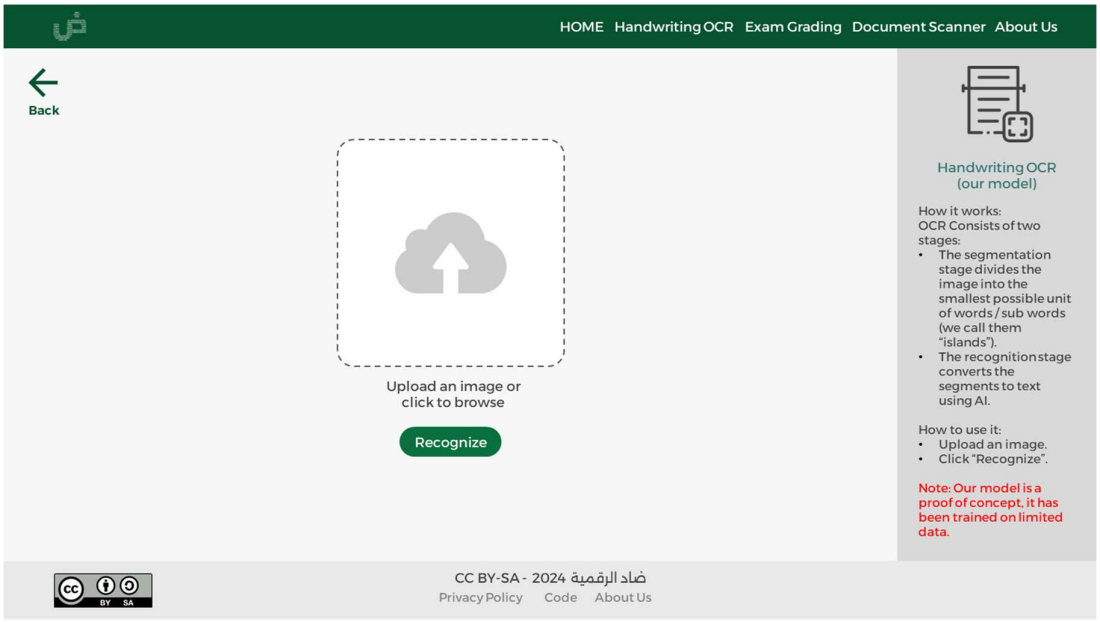


Figure 6.2: Screenshot of "our model" page

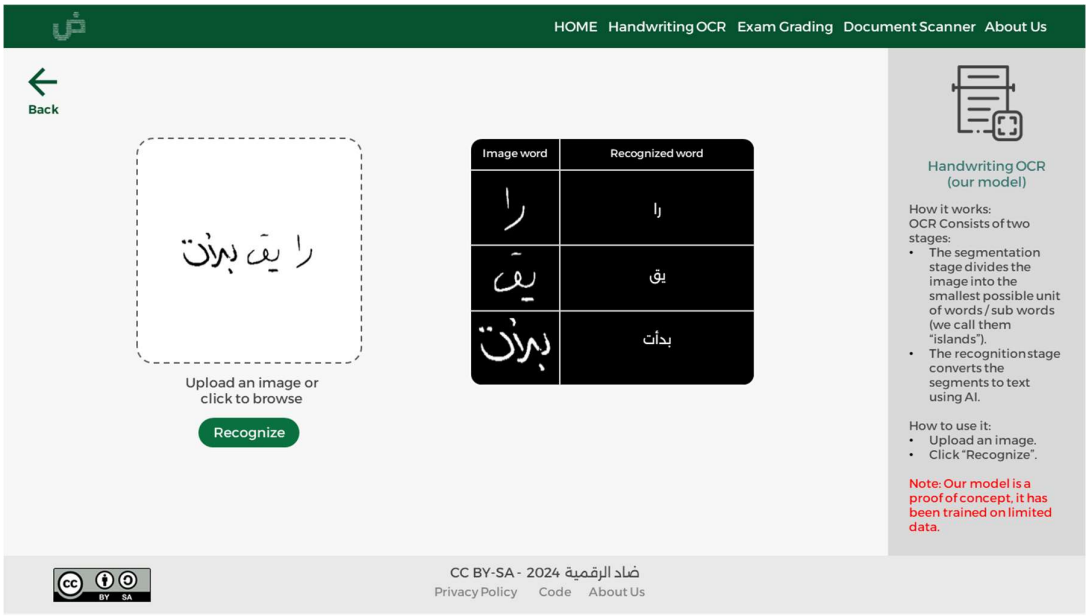


Figure 6.3: Screenshot of how results are displayed

6.2.2 Backend

The backend is the part of the system responsible for processing input images using the segmentation system and then passing these images to the model for recognition, ultimately returning the labels. It serves the crucial purpose of offloading the computationally intensive tasks from the frontend, ensuring that the user interface remains responsive and efficient.

6.2.2.1 Developing an API

In a frontend-backend model application, an API (Application Programming Interface) is essential for communication between the two parts. The API allows the frontend to send image data to the backend, where it can be processed, and then receive the recognized labels. This separation of concerns ensures that each part of the application can be developed and maintained independently, improving modularity and scalability.

6.2.2.2 Fake API

Initially, we developed a simple fake API that processed the image in an arbitrary way and returned predefined labels. This fake API was crucial for testing the API's functionality and determining which package to use for developing the final API. By simulating the backend's behavior, we were able to make informed decisions about the tools and frameworks we would employ.

6.2.2.3 FastAPI

FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.6+ based on standard Python-type hints. It is designed to be easy to use and deploy, offering automatic interactive API documentation.

Despite its advanced features, we found that FastAPI was more complex than necessary for our application, which primarily involved sending images and receiving labels. The additional features and complexity of FastAPI did not provide a significant advantage for our relatively straightforward requirements [93].

6.2.2.4 Flask

Flask is a lightweight WSGI web application framework in Python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Flask is simple and flexible, providing the essentials without requiring extensive boilerplate code.

We found Flask to be easy to work with, allowing us to define routes and handle requests with minimal effort. Its simplicity made it ideal for our needs, as we only needed to send images to the backend and receive recognized labels in return.

After experimenting with different frameworks, we decided to use Flask and implemented the entire backend system with it. Flask provided the right balance of simplicity and functionality, making it the optimal choice for our application.

6.2.2.5 Hardware Configuration

The backend was implemented in a local environment (self-hosted) rather than on the cloud. This decision was based on our need to have full control over the hardware and software environment, ensuring optimal performance and ease of maintenance.
The hardware configuration used in the backend includes:

- **Intel Core i3-12100F Processor:** A budget-friendly central processing unit (CPU) capable of handling the necessary computations for our OCR system.

- **16 Gigabytes of Random Access Memory (RAM):** Sufficient memory to manage the image processing and model inference tasks.
- **NVIDIA GeForce RTX 3060 Graphics Processing Unit (GPU):** A powerful GPU that significantly accelerates the processing of images and model computations, making the system efficient and responsive.

By using this hardware configuration, we ensured that our backend could handle the demands of the OCR processing effectively, providing quick and accurate results.

CHAPTER 7

AI TRAINING & RESULTS

CHAPTER 7 AI SYSTEM IMPLEMENTATION

7.1 Our Model's Methodology

Our model for Arabic handwriting recognition project as shown in Figure 7.1 begins with the `Image` input layer, which takes 64x64 images. These inputs are then fed into a `CNN` that extracts core features from the images. The model further processes these features through a `Dense Layer`, providing a meaningful representation of the input. Following this, a `Batch Normalization` layer stabilizes the network by normalizing the output of the dense layer. A `Dropout Layer` is used to reduce overfilling by ignoring a fraction of the input. The `Bi-LSTM (RNN)` layer is critical for analyzing sequences of the input image, making it ideal for recognizing the sequential nature of Arabic handwriting. The output of the Bi-LSTM is stabilized by another `Batch Normalization` layer, followed by another `Dropout Layer`. The model then uses a `Softmax Layer` to generate the predictive output, ensuring accurate transcription of the handwritten input. The `CTC Loss Layer` forms the final part of the network, handling the sequence and reducing the error percentage by accurately calculating the CTC loss and providing feedback for improved performance. The model's output is generated after the `CTC Loss Layer`, providing an accurate sequence of characters that can be used as the final output for the Arabic handwriting recognition.

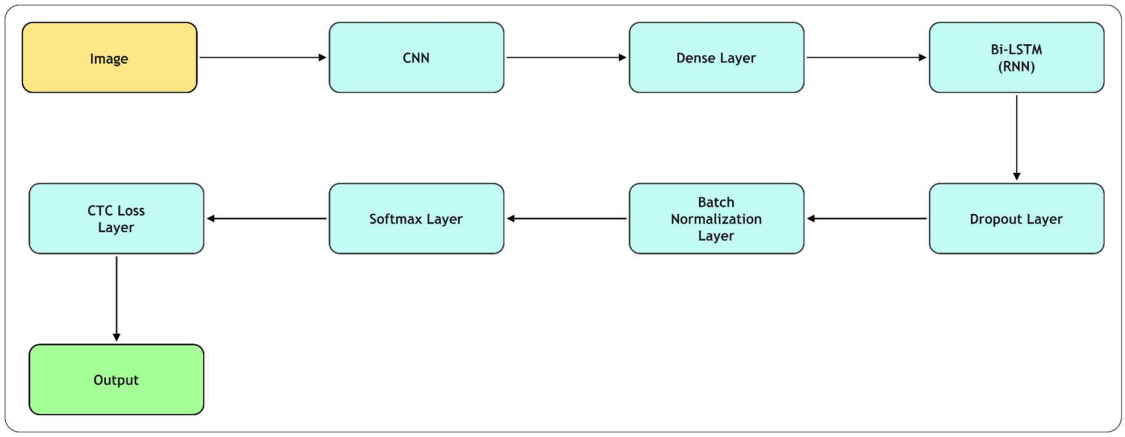


Figure 7.1: Our Model's Approach

7.2 Initial Experiments: Prototype Model

In the initial phase of our project, we focused on developing a prototype model using a smaller dataset to validate our approach and refine our techniques. This allowed us to quickly iterate and experiment with different Convolutional Neural Network (CNN) architectures. By testing various architectures, we aimed to identify the model that offered the best balance between accuracy, computational efficiency, and generalization capabilities. Experimenting with these different architectures enabled us to understand their performance in the context of our specific application, guiding us toward the most suitable model for further development and deployment.

7.2.1 Initial Dataset

For the initial model prototype, we utilized a subset of the KHATT dataset containing 5,159 samples, which we augmented to a total of 15,477 samples shown in Figure 7.2. This dataset covered 36 Arabic letters and included a variety of segmented words and sub-words. As part of our preprocessing steps, all images were resized to a uniform 32x64 resolution. This resizing process ensured consistency in input dimensions across the dataset, facilitating efficient training and evaluation of our models. By standardizing the image size, we aimed to

optimize computational resources while maintaining the integrity and quality of the dataset for accurate character recognition and model performance.

Figure 7.2: Samples of Initial Dataset Showing Arabic Words

7.2.2 Experiments And Results

In the pursuit of refining model performance, experimentation with various architectures plays a crucial role. This process involves exploring different configurations and designs of neural networks, particularly focusing on convolutional neural networks (CNNs) for tasks such as image recognition and classification. Each architecture variant may include adjustments in the number of layers, types of layers (e.g., convolutional, pooling, fully connected), activation functions, and regularization techniques. By testing multiple architectures, researchers aim to identify the configuration that optimizes model accuracy, efficiency, and generalization capabilities across diverse datasets and real-world scenarios. This iterative exploration allows for the discovery of architectures that strike a balance between complexity and performance metrics, ensuring robust and reliable outcomes in machine learning applications. In the initial prototype model using the initial dataset we experimented with different CNNs; EfficientNetB1 [94], VGG19 [95], and lastly ResNet152 [96].

7.2.2.1 EfficientNetB1

EfficientNetB1's technical foundation lies in its innovative application of compound scaling, a method that harmonizes the dimensions of depth, width (number of channels), and image resolution within the neural network architecture. By scaling these dimensions

jointly rather than independently, EfficientNetB1 optimizes both computational efficiency and model performance for tasks such as Arabic handwriting recognition. Increased depth allows the network to learn intricate hierarchies of features essential for interpreting diverse Arabic characters. Expanded width enhances its capacity to process information in parallel, crucial for capturing nuances in handwritten script. Resolution scaling ensures efficient utilization of computational resources by adapting input image sizes to maximize accuracy without unnecessary computational overhead. This integrated approach enables EfficientNetB1 to excel in recognizing Arabic handwriting by effectively balancing model complexity and efficiency, making it a robust choice for real-world applications requiring precise character recognition in Arabic script and the Figure 7.3 [97] shows the EfficientNetB1 architecture.

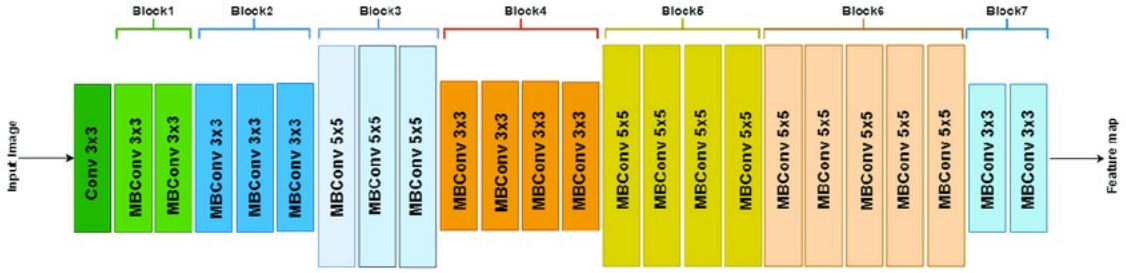


Figure 7.3: EfficientNetB1 Architecture

7.2.2.2 VGG19

VGG19 is a deep convolutional neural network architecture shown in Figure 7.4 [98], renowned for its simplicity and effectiveness in image recognition tasks. Developed by the Visual Geometry Group at Oxford, VGG19 consists of 19 layers, including 16 convolutional layers and 3 fully connected layers. Its design is characterized by using small 3x3 convolutional filters throughout the network, which allows it to capture intricate features in input images. VGG19's architecture emphasizes depth and homogeneous filter sizes, facilitating a straightforward interpretation and implementation. Despite its simplicity compared to

newer architectures, VGG19 remains highly effective due to its ability to extract detailed features from images, making it a strong contender for tasks requiring precise image classification and feature extraction, including applications in object detection and recognition.

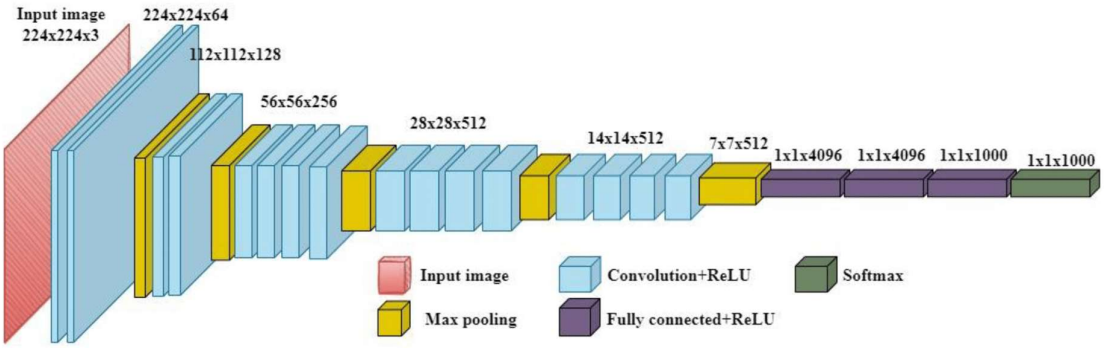


Figure 7.4: VGG19 Architecture

7.2.2.3 ResNet152

ResNet152, part of the Residual Network (ResNet) family, represents a significant advancement in deep learning architectures and its architecture shown in Figure 7.5 [99], particularly for tasks like image recognition and classification. Developed by Microsoft Research, ResNet152 is distinguished by its deep structure comprising 152 layers, including residual blocks that mitigate the vanishing gradient problem in deep networks. This architecture introduces skip connections, or shortcuts, that allow gradients to flow more directly during training, enabling efficient learning of complex features. ResNet152's design leverages residual learning principles to achieve state-of-the-art performance on various benchmarks, demonstrating robustness and scalability in handling large-scale datasets. Its depth and skip connections enhance model accuracy by facilitating the training of deeper networks without sacrificing computational efficiency, making ResNet152 a pivotal architecture in advancing the capabilities of deep learning models for challenging tasks in computer vision and beyond.

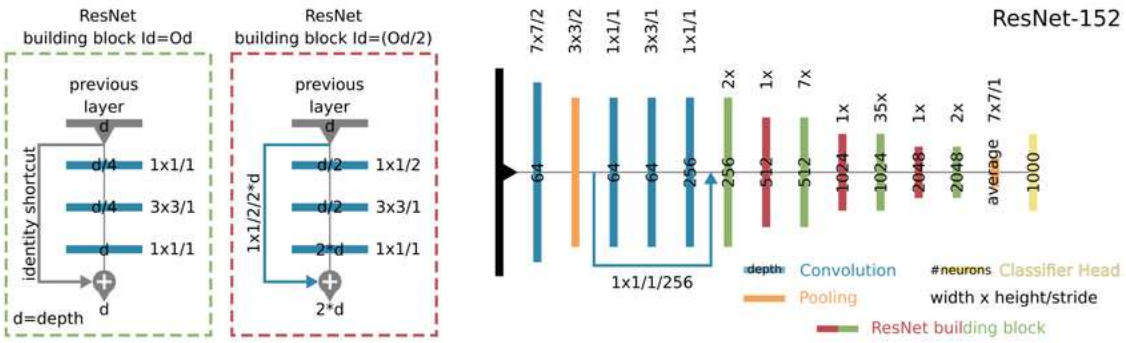


Figure 7.5: ResNet152 Architecture

7.2.2.4 Training & Results

During the training phase, we trained our models for 70 epochs, experimenting with different optimizers and hyperparameters to identify the best configuration. We utilized a learning rate scheduler to dynamically adjust the learning rate during training. The learning rate scheduler monitors the validation loss and, if there is no improvement over 10 consecutive epochs, it multiplies the learning rate by the factor of 0.2. This approach helps in fine-tuning the model's learning process, allowing it to converge more effectively and avoid getting stuck in local minima. By carefully managing the learning rate, the scheduler ensures that the model maintains a balance between learning efficiently and not overshooting the optimal solution, ultimately improving the model's performance and stability. Hyperparameters used for each model are shown in Table 1.

Parameters	VGG19	EfficientNetB1	ResNet152
Optimizer	ADAM	NADAM	NADAM
Learning Rate	0.0001	0.001	0.001
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999
ε	1×10^{-8}	1×10^{-8}	1×10^{-8}
Batch Size	128	128	128

Table 1: Hyperparameters Used for each Experimented CNN

After evaluating the performance of our models based on validation loss, character error rate (CER), and accuracy metrics, distinct outcomes were observed across different architectures:

- VGG19: Validation loss of 0.6, CER of 5.4%, and accuracy of 94.6%.
- EfficientNetB1: Validation loss of 0.78, CER of 7.3%, and accuracy of 92.7%.
- ResNet152: Superior results with a validation loss of 0.3, CER of 2.96%, and accuracy of 97.04%.

These findings highlight that ResNet152 outperformed both VGG19 and EfficientNetB1 across all evaluated metrics, making it the most effective model for our specific task.

The comparison between the models' performance is summarized in Table 2, emphasizing ResNet152's superiority. Additionally, Figure 7.6 depicting the CER and loss of the ResNet152 model throughout the epochs shows a noticeable decrease in error, particularly in the final epochs.

Model	Validation Loss	Character Error Rate	Accuracy
VGG19	0.6	5.4%	94.6%
EfficientNetB1	0.78	7.3%	92.7%
ResNet152	0.3	2.96%	97.04%

Table 2: Comparison Between CNN Architectures Performance

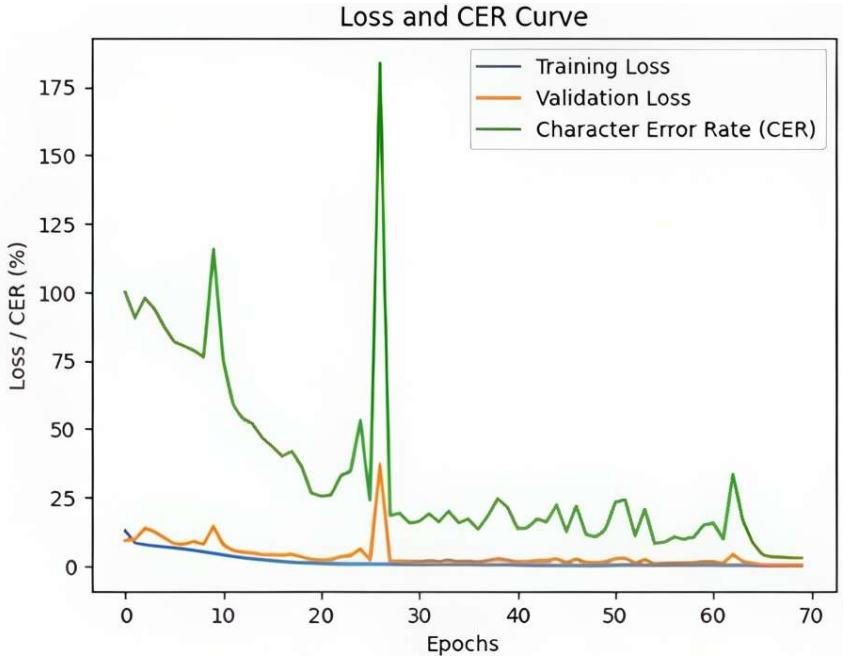


Figure 7.6: ResNet152 Performance Throughout the Epochs

7.3 Optimized Final Model

Moving from experimental prototype models to developing the final model involved a systematic approach to refine and optimize our initial findings. Initially, we explored various CNN architectures such as VGG19, EfficientNetB1, and ResNet152 using smaller datasets to gauge their performance and suitability. Upon identifying ResNet152 as the most promising due to its superior validation loss, character error rate, and accuracy metrics, we transitioned to a more rigorous development phase.

7.3.1 Larger Dataset

We curated an extensive dataset incorporating various sources, including a substantial portion of the KHATT Dataset originally comprising approximately 20,000 samples, which after meticulous cleaning and augmentation processes, expanded to 50,145 samples. Additionally, we integrated data from the AHAWP and Arabic Handwritten Characters datasets, also yielding a combined total of

58,474 samples post-cleaning and augmentation. This unified dataset was meticulously structured to encompass 36 Arabic letters, with all images standardized to a resolution of 64x64 pixels. This standardization ensured uniformity in input dimensions across the dataset, essential for training robust and accurate models capable of recognizing and interpreting Arabic characters across diverse handwritten styles and conditions. By incorporating a wide range of characters and maintaining consistent image quality, our dataset provided a comprehensive foundation for developing advanced machine learning models tailored for Arabic character recognition tasks.

7.3.2 Chosen Optimal Architecture: ResNet50V2

We opted for ResNet50V2 as our optimal model over ResNet152, despite the latter's larger size and higher resolution dataset, due to several considerations. ResNet50V2 strikes a balance between performance and computational efficiency, making it more feasible for scaling up to larger datasets with higher resolutions. To enhance its training efficacy, we implemented advanced techniques like the "**Cosine Learning Rate Scheduler**". This scheduler dynamically adjusts the learning rate during training according to a Cosine wave as shown in Figure 7.7 [100], potentially improving model convergence and generalization.

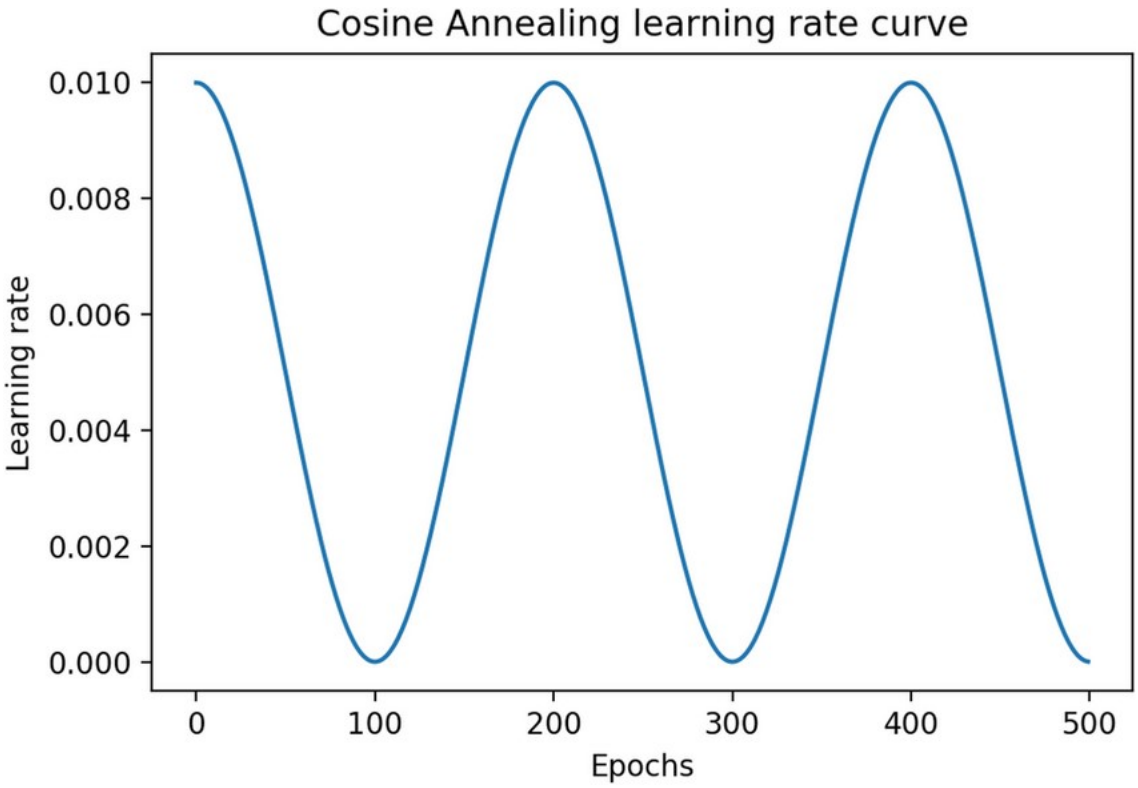


Figure 7.7: Example of Cosine Learning Rate Scheduler Effect

We used Nadam as the optimizer with an initial learning rate of 0.001, a batch size of 128, and trained the model for 70 epochs. The Nadam optimizer parameters included $\beta 1$=0.9, $\beta 2$=0.999, and ε=1 x e-08. Initially tested on a dataset with a resolution of 32x32, that test resulted in a model with a Character Error Rate (CER) of 4.62% corresponding to 95.38% accuracy. Performance of the model is shown in Figure 7.8.

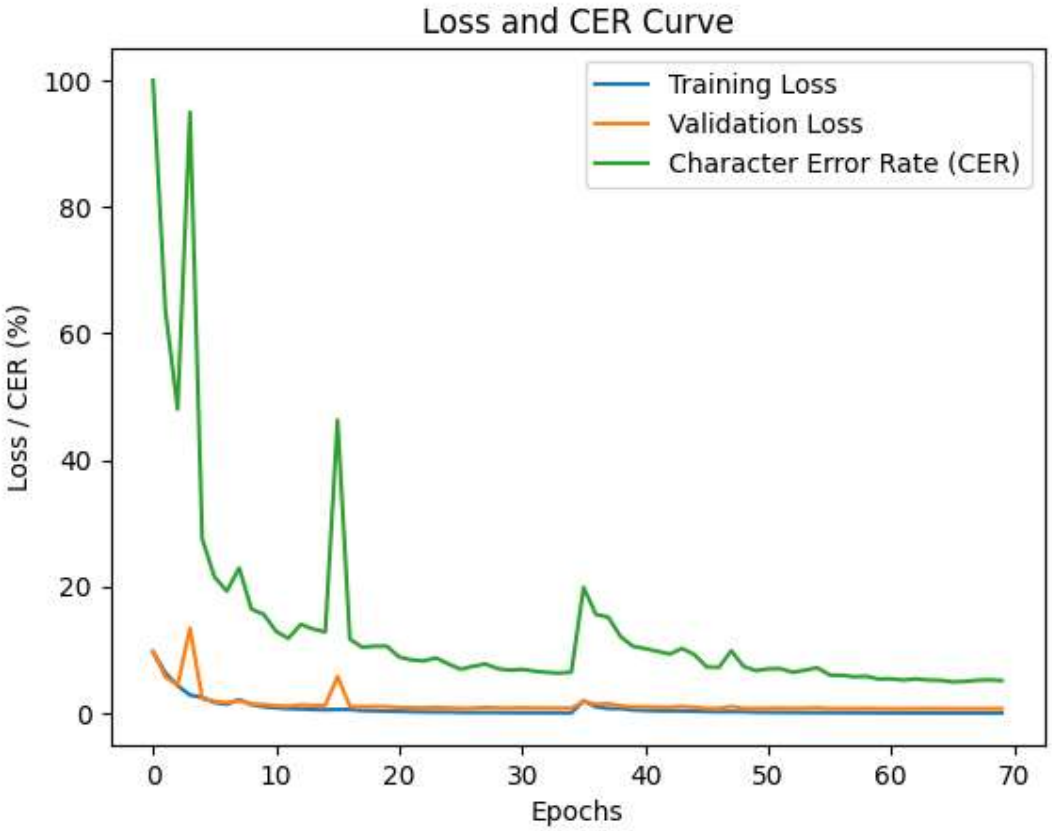


Figure 7.8: ResNet50V2 Performance on The Larger Dataset

Recognizing the need for further improvement, we employed transfer learning techniques to refine the model's performance, leveraging ResNet50V2's capabilities and our refined training approach to achieve robust performance and scalability necessary for our specific application.

7.3.3 ResNet50V2 Arabic Alphabet Transfer Learning

Transfer learning is a technique in machine learning where a model trained on one task is leveraged as a starting point for a new task as shown in Figure 7.9 [101]. This approach utilizes knowledge gained from solving one problem and applies it to a different but related problem, typically resulting in improved learning efficiency and performance. In our case, after initially training ResNet50V2 on an Arabic Alphabet Character dataset consisting of 58,474 samples, we easily achieved a

low CER, high accuracy model that's well trained to recognize Arabic Alphabetical Letters. This success prompted us to further enhance the model's capabilities by transferring its learned weights to the main KHATT Dataset, which comprises 50,145 samples. By fine-tuning the model on this larger and more diverse dataset, we aim to capitalize on the features and patterns learned from the Arabic Alphabet dataset, thereby improving the model's accuracy and robustness for recognizing handwritten Arabic text in the KHATT Dataset. This application of transfer learning allows us to efficiently adapt the model to new data while leveraging existing knowledge to achieve superior performance in a related domain.

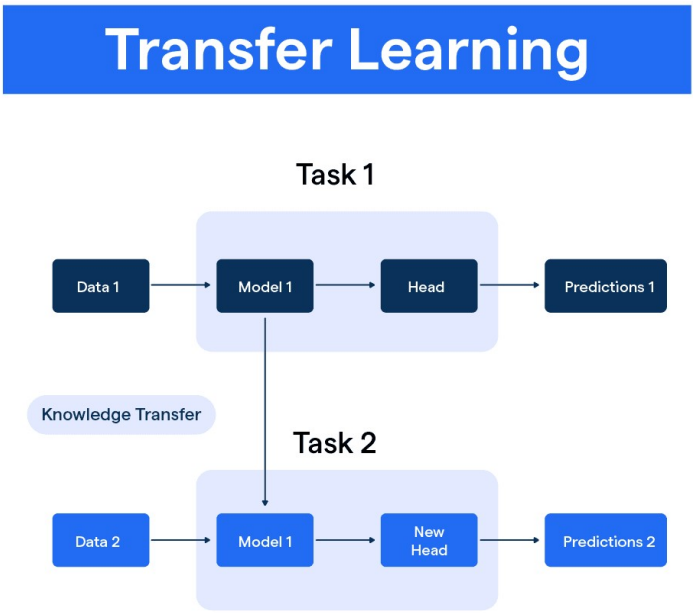


Figure 7.9: Demonstrating Transfer Learning Technique

7.3.4 Final Results

In our final approach, we loaded the ResNet50V2 model pre-trained on the Arabic Alphabet Character dataset and continued training on the main KHATT Dataset using advanced techniques and optimal hyperparameters. The model was trained with a cosine learning rate scheduler, Adam optimizer with an initial learning rate of 0.001, $\beta 1$ of 0.9, $\beta 2$ of 0.999, and ε set to 1 x e-7, across 70 epochs.

Combining the 58,474 samples from the alphabet dataset with the 50,145 samples from the KHATT Dataset, the model was trained on a total of 108,619 samples. Through this comprehensive training regimen, our model achieved outstanding results showing performance as in Figure 7.10, boasting a CER of only **3%** on the test set and an Accuracy of **97%**. These metrics underscore the model's robustness and accuracy in recognizing handwritten Arabic text, demonstrating its effectiveness in real-world applications.

This final model represents a culmination of rigorous experimentation, meticulous parameter tuning, and leveraging the power of transfer learning. Its high accuracy and low error rate position it as a reliable tool for applications requiring precise recognition of Arabic characters and text, validating its capability as our definitive and highly effective model.

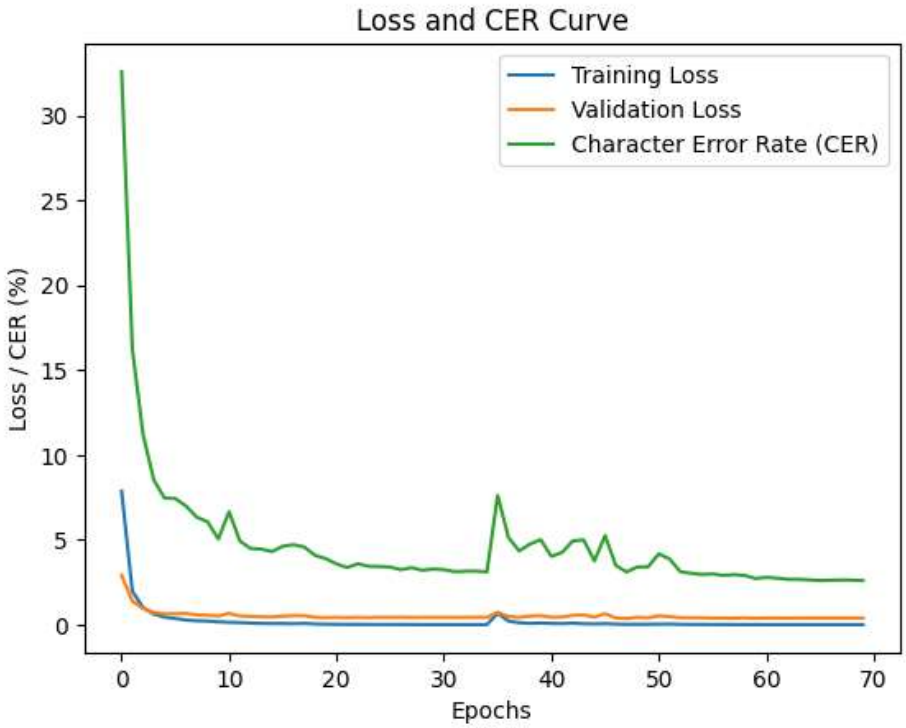


Figure 7.10: ResNet50V2 Trained on Alphabet from KHATT Dataset

CHAPTER 8

CONCLUSION & FUTURE WORK

CHAPTER 8 CONCLUSION AND FUTURE WORK

8.1 Conclusion

Driven by our passion for developing advanced text-processing techniques in Arabic, we recognized their potential to bolster various areas we mentioned earlier (education, government, data analysis, and information retrieval). This dedication led us to closely monitor advancements in Arabic handwriting recognition. We were particularly impressed by the recent research paper "Handwritten Arabic Bills Reader and Recognizer." It motivated us to push our own research, culminating in noteworthy results. Our final model, the ResNet50V2, pre-trained on the Arabic Alphabet Character dataset and further trained on the main KHATT Dataset using advanced techniques and optimal hyperparameters, yielded significantly better performance compared to the referenced research. This achievement, using a cosine learning rate scheduler, Adam optimizer with an initial learning rate of 0.001, β1 of 0.9, β2 of 0.999, and ε set to 1 x e-7, across 70 epochs, represents a substantial leap in accuracy (97%) and a low Character Error Rate (CER) of only 3% on the test set. This accomplishment fuels our commitment to further research in this domain, with the ultimate goal of achieving even more advanced levels of Arabic handwriting recognition.

8.2 Future Work

The Arabic language is rich in special characters and diacritics that can significantly alter the meaning of a word. As a result, future research will involve creating a more diverse dataset, and developing more sophisticated models that are specifically trained to recognize and interpret these characters accurately across a more diverse dataset. This could involve creating datasets from scratch; ones that focus on these special characters and incorporating more complex models that can capture the nuances of their usage and placement. We plan to focus on collecting and preparing more comprehensive and diverse datasets. This includes incorporating more datasets that represent different age

groups, writing styles, and handwriting conditions, such as varying levels of pressure, speed, and ink darkness, and if possible, across multiple mediums. The development of synthetic datasets or the augmentation of existing datasets with techniques like rotation, scaling, and noise could also help improve the model's robustness against real-world conditions and make it a powerful tool to be used in real-world applications.

8.2.1 Searchable Handwritten Note Taking App

A Searchable Handwritten Note Taking App is a mobile or desktop application that allows users to take notes by hand using a stylus or touch screen, and then search and access those notes later using keywords or text recognition.

8.2.1.1 Key Features

- Handwritten Note Taking: This is the core functionality, allowing users to write notes directly on the app's interface using a stylus or their finger.
- Ink-to-Text Conversion (OCR): The app uses Optical Character Recognition (OCR) technology to convert handwritten notes into digital text. This enables searching and indexing of the content.
- Search Functionality: Users can search for specific keywords or phrases within their handwritten notes using the converted text.
- Organization Tools: The app might offer features for organizing notes by tags, categories, or creating folders.
- Cloud Storage and Syncing: Some apps allow storing notes in the cloud and syncing them across multiple devices.
- Additional Features: Advanced apps might offer features like audio recording alongside handwritten notes, annotation capabilities on imported PDFs, or handwriting recognition in multiple languages.

8.2.1.2 Existing Examples:

- Notability
- GoodNotes
- Evernote (with handwriting recognition add-on)
- Microsoft OneNote
- Noteshelf

8.2.2 Automatic Exam Grading Software

Our future research aims to develop new techniques for recognizing mathematical equations within Arabic handwritten text. This includes training models to accurately parse and interpret mathematical symbols, expressions, and formulas, which are integral components of various subjects including mathematics, physics, and engineering. If successful, we envision integrating these advancements into a comprehensive system that combines Language Models (LLMs) with OCR models. This combined system will possess the capability to automatically grade handwritten Arabic exams. By leveraging the strengths of LLMs in understanding context and semantics alongside the precision of OCR in recognizing characters and symbols, we aim to create a powerful tool for educators to efficiently assess student performance in Arabic written subjects, including those involving mathematical components. This system could significantly streamline the grading process, reduce manual effort and human error, and provide timely feedback to both students and instructors. Automatic Exam Grading Software is a type of educational technology designed to streamline the process of evaluating student performance in multiple-choice, fill-in-the-blank, and similar standardized tests.

8.2.2.1 Core Features:

- Scannable Answer Sheets: The software works by scanning pre-formatted answer sheets marked by students. These sheets

typically involve filling in bubbles for multiple-choice questions or writing short answers in designated areas.

- Optical Mark Recognition (OMR): The software utilizes OMR technology to recognize marks or written responses on the answer sheets. OMR can translate these marks into digital data for further processing.
- Answer Key Integration: Educators can input the correct answers or answer patterns for each question into the software.
- Automatic Scoring: The software compares students' responses on the scanned sheets with the answer key, automatically calculating scores for each student and potentially the entire class.
- Data Analysis and Reporting: Some software may offer features for generating reports that summarize student performance and provide insights into class-wide understanding or areas where students might need additional support.

The use of Automatic Exam Grading Software is becoming increasingly common as technology advances. However, it's important to use it strategically, understanding its limitations and ensuring effective integration with other assessment methods.

REFERENCES

[1] H. P. E. D. LP, “OCR Document API.” 2016. [Online]. Available: https://dev.havenondemand.com/apis/ocrdocument

[2] Innovatrics, “OCR (Optical Character Recognition).” 2023. [Online]. Available: https://www.innovatrics.com/glossary/ocr-optical-character-recognition/

[3] J. W. T. Smith and Z. Merali, *Optical Character Recognition: The Technology and Its Application in Information Units and Libraries*. in Fact sheet / British Library. Research and Development Dept. British Library, 1985. [Online]. Available: https://books.google.com.eg/books?id=okGxAAAAIAAJ

[4] A. W. Services, “What is Optical Character Recognition (OCR)?” 2024. [Online]. Available: https://aws.amazon.com/what-is/ocr/

[5] C. S. Smith, “What Is OCR (Optical Character Recognition) Technology?,” *Forbes*, 2023, [Online]. Available: https://www.forbes.com/sites/technology/article/what-is-ocr-technology/

[6] V. Märgner and H. El Abed, *Guide to OCR for Arabic Scripts*. Springer Science & Business Media, 2024.

[7] M. Balat, Y. Mohamed, A. Heakl, and A. Zaky, “Arabic Handwritten Text for Person Biometric Identification: A Deep Learning Approach,” *ArXiv*, vol. 2406.00409v1, 2024, [Online]. Available: https://arxiv.org/html/2406.00409v1

[8] S. V Rice, G. Nagy, and T. A. Nartker, *Optical Character Recognition: An Illustrated Guide to the Frontier*. Springer Science & Business Media, 2024.

[9] N. Ramesh, A. Srivastava, and K. Deeba, “Improving Optical Character Recognition Techniques,” *International Journal of Engineering & Technology*, vol. 7, no. 2.24, p. 361, Jul. 2024, doi: 10.14419/ijet.v7i2.24.12085.

[10] B. Kunkel, “We Built one of the Top OCR Tools.” Jun. 2022. [Online]. Available: https://code.pieces.app/blog/top-ocr-tools

[11] Medium, “Evaluating Offline Handwritten Text Recognition: Which Machine Learning Model is the Winner?,” *Medium*, vol. 8, no. 2, pp. 63–78, 2024.

[12] L. Hamami and D. Berkani, “Recognition system for printed multi-font and multi-size Arabic characters,” *Arab J Sci Eng*, vol. 27, Apr. 2002.

[13] Conexiom, “The Advantages, Challenges, and Alternatives to OCR Solutions,” *Conexiom*, vol. 12, no. 4, pp. 45–57, 2024.

[14] Z. Shi, S. Setlur, and V. Govindaraju, “Pre-processing Issues in Arabic OCR,” in *Guide to OCR for Arabic Scripts*, London: Springer London, 2012, pp. 79–102. doi: 10.1007/978-1-4471-4072-6_4.

[15] I. Ahmad and G. Fink, “Handwritten Arabic text recognition using multi-stage sub-core-shape HMMs,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 22, Sep. 2019, doi: 10.1007/s10032-019-00339-8.

[16] V. Kulesh, K. Schaffer, I. Sethi, and M. Schwartz, “Handwriting quality evaluation,” in *International Conference on Advances in Pattern Recognition*, Berlin, Heidelberg, Mar. 2001, pp. 157–165.

[17] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber, “A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks,” 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:5668166

[18] H. Akouaydi, Y. Hamdi, H. Boubaker, M. Zaied, F. Alaya Cheikh, and A. Alimi, *Children’s Online Handwriting Quality Analysis.* 2021. doi: 10.36227/techrxiv.14776338.v1.

[19] M. Suganthi and R. Arun Prakash, “An offline English optical character recognition and NER using LSTM and adaptive neuro-fuzzy inference system,” *Journal of Intelligent & Fuzzy Systems*, vol. 44, pp. 3877–3890, 2023, doi: 10.3233/JIFS-221486.

[20] P. Ahmed and Y. Al-Ohali, “Arabic Character Recognition: Progress and Challenges,” *Journal of King Saud University - Computer and Information Sciences*, vol. 12, pp. 85–116, 2000, doi: https://doi.org/10.1016/S1319-1578(00)80004-X.

[21] "Arabic Language Day." Statistics Canada, 2024. [Online]. Available: https://www.statcan.gc.ca/o1/en/plus/2555-arabic-language-day

[22] Z. Alyafeai, M. S. Al-shaibani, M. Ghaleb, and Y. A. Al-Wajih, "Calliar: an online handwritten dataset for Arabic calligraphy," *Neural Comput Appl*, vol. 34, no. 23, pp. 20701–20713, 2022, doi: 10.1007/s00521-022-07537-2.

[23] M. Gudenburg, "Navigating the Challenges of OCR-based Research in the Arabic Script and Language." 2023. [Online]. Available: https://href.hypotheses.org/2516

[24] D. Simonnet and E. Anquetil, "Handwriting Quality Analysis of Block Letters and Cursive Words," *Handwriting Today, Journal of the National Handwriting Association*, no. 15, pp. 15–21, Dec. 2016, [Online]. Available: https://hal.science/hal-01484924

[25] 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, 2023, doi: 10.3390/app13074584.

[26] A. Zidouri, "ORAN: a basis for an Arabic OCR system," in *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, 2004, pp. 703–706. doi: 10.1109/ISIMP.2004.1434161.

[27] N. Nayef and J. M. Ogier, "Metric-based no-reference quality assessment of heterogeneous document images," in *Document Recognition and Retrieval XXII*, Feb. 2015, p. 94020L.

[28] H. Osman, K. Zaghw, M. Hazem, and S. Elsehely, "An Efficient Language-Independent Multi-Font OCR for Arabic Script." 2020.

[29] A. Mezghani, R. Maalej, M. Elleuch, and M. Kherallah, "Recent advances of ML and DL approaches for Arabic handwriting recognition: A review," *Int. J. Hybrid Intell. Syst.*, vol. 19, no. 1,2, pp. 61–78, Jul. 2023, doi: 10.3233/HIS-230005.

[30] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[31] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D*, vol. 404, p. 132306, 2020, doi: 10.1016/j.physd.2019.132306.

[32] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural Comput*, vol. 9, pp. 1735–1780, Mar. 1997, doi: 10.1162/neco.1997.9.8.1735.

[33] A. Mostafa *et al.*, *An End-to-End OCR Framework for Robust Arabic-Handwriting Recognition using a Novel Transformers-based Model and an Innovative 270 Million-Words Multi-Font Corpus of Classical Arabic with Diacritics*. 2022.

[34] N. I. Youssef and N. Abd-Alsabour, “A Review on Arabic Handwriting Recognition,” *Journal of Southwest Jiaotong University*, vol. 57, no. 6, pp. 746–756, Dec. 2022, doi: 10.35741/issn.0258-2724.57.6.66.

[35] M. S. Kasem, M. Mahmoud, and H.-S. Kang, “Advancements and Challenges in Arabic Optical Character Recognition: A Comprehensive Survey,” *ArXiv*, vol. 2312.11812v1, 2023, [Online]. Available: https://arxiv.org/html/2312.11812v1

[36] A. Alaei, V. Bui, D. Doermann, and U. Pal, “Document Image Quality Assessment: A Survey,” *ACM Comput. Surv.*, vol. 56, no. 2, Sep. 2023, doi: 10.1145/3606692.

[37] G. Cloud, “Use Cases for Optical Character Recognition (OCR) on Google Cloud.” Google. [Online]. Available: https://cloud.google.com/use-cases/ocr?hl=en

[38] R. Smith, “An Overview of the Tesseract OCR Engine,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[39] S. V Rice, F. R. Jenkins, and T. A. Nartker, “The Fourth Annual Test of OCR Accuracy.”

[40] G. C. Blog, “Announcing Tesseract OCR.” Aug. 2006.

[41] T. Kanungo, G. A. Marton, and O. Bulbul, “Performance evaluation of two Arabic OCR products,” in *Proc.SPIE*, Jan. 1999, pp. 76–83. doi: 10.1117/12.339809.

[42] A. Carrizo, J. Garbajosa, and G. Molina, “Requirements gathering: the journey,” *International Journal of Decision Systems*, vol. 23, no. 4, pp. 667–681, 2014.

[43] Z. Shi, S. Setlur, and V. Govindaraju, “Pre-processing Issues in Arabic OCR,” in *Guide to OCR for Arabic Scripts*, London: Springer London, 2012, pp. 79–102. doi: 10.1007/978-1-4471-4072-6_4.

[44] K. Safjan, “15 Tools for Document Deskewing and Dewarping,” *Krystian’s Safjan Blog*, 2022.

[45] Joel, “Document Scanner using OpenCV (HoughLines Approach).” Apr. 2020. [Online]. Available: https://medium.com/analytics-vidhya/document-scanner-using-opencv-houghlines-approach-eb276dd4a0a

[46] H. Feng, Y. Wang, W. Zhou, J. Deng, and H. Li, “DocTr: Document Image Transformer for Geometric Unwarping and Illumination Correction,” in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 273–281.

[47] H. Feng, W. Zhou, J. Deng, Q. Tian, and H. Li, “DocScanner: Robust Document Image Rectification with Progressive Learning,” *arXiv preprint arXiv:2110.14968*, 2021.

[48] H. Feng, S. Liu, J. Deng, W. Zhou, and H. Li, “Deep Unrestricted Document Image Rectification,” *IEEE Trans Multimedia*, 2023.

[49] S. Das, K. Ma, Z. Shu, D. Samaras, and R. Shilkrot, “DewarpNet: Single-Image Document Unwarping With Stacked 3D and 2D Regression Networks,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[50] R. Santos, G. Clemente, T. Ing Ren, and G. Cavalcanti, “Text Line Segmentation Based on Morphology and Histogram Projection,” *Document Analysis and Recognition, International Conference on*, vol. 0, pp. 651–655, Feb. 2009, doi: 10.1109/ICDAR.2009.183.

[51] V. Alkalai Mohamed and Sorge, “A Histogram-Based Approach to Mathematical Line Segmentation,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, G. Ruiz-Shulcloper José and Sanniti di Baja, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 447–455.

[52] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Higher Education, 2024.

[53] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, “A Systematic Literature Review of A* Pathfinding,” *Procedia Comput*

Sci, vol. 179, pp. 507–514, 2021, doi: https://doi.org/10.1016/j.procs.2021.01.034.

[54] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character Region Awareness for Text Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9365–9374.

[55] D. Madhugiri, "Extract Text from Images Quickly Using Keras-OCR Pipeline," *Analytics Vidhya*, 2022, [Online]. Available: https://www.analyticsvidhya.com/blog/2022/09/extract-text-from-images-quickly-using-keras-ocr-pipeline/

[56] M. PRATA, "Keras OCR Text Recognition," *Kaggle*, Sep. 20, 2022. [Online]. Available: https://www.kaggle.com/code/mpwolke/keras-ocr-text-recognition

[57] S. A. Mahmoud *et al.*, "KHATT: Arabic Offline Handwritten Text Database," in *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 449–454. doi: 10.1109/ICFHR.2012.224.

[58] S. A. Mahmoud *et al.*, "KHATT: An open Arabic offline handwritten text database," *Pattern Recognit*, vol. 47, no. 3, pp. 1096–1112, Feb. 2014, doi: 10.1016/j.patcog.2013.08.009.

[59] D. Powers, *PHP Solutions: Dynamic Web Design Made Easy*. Apress, 2024.

[60] K. W. Hon, "Artificial neural networks," in *Technology and Security for Lawyers and Other Professionals*, Edward Elgar Publishing, 2024, pp. 490–511.

[61] A. Hashemi, G. Orzechowski, A. Mikkola, and J. McPhee, "Multibody dynamics and control using machine learning," *Multibody Syst Dyn*, vol. 58, pp. 1–35, Feb. 2023, doi: 10.1007/s11044-023-09884-x.

[62] M. F. Ijaz and M. Woźniak, "A Survey of Deep Convolutional Neural Networks Applied for Prediction of Plant Leaf Diseases," *Sensors*, vol. 21, no. 14, p. 4749, 2021, doi: 10.3390/s21144749.

[63] C. Xiao and J. Sun, "Deep Neural Networks (DNN)," in *Introduction to Deep Learning for Healthcare*, Springer, Cham, 2021. doi: 10.1007/978-3-030-82184-5_4.

[64] IBM, “What are Recurrent Neural Networks?” 2024. [Online]. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks

[65] Wikipedia, “Recurrent Neural Network.” 2024. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network

[66] AWS, “What is RNN? - Recurrent Neural Networks Explained.” 2024. [Online]. Available: https://aws.amazon.com/machine-learning/what-is-rnn/

[67] DataCamp, “Recurrent Neural Network Tutorial (RNN).” 2024. [Online]. Available: https://www.datacamp.com/tutorial/recurrent-neural-networks-tutorial

[68] T. O’Shea, S. Hitefield, and J. Corgan, “End-to-End Radio Traffic Sequence Recognition with Deep Recurrent Neural Networks,” Mar. 2016.

[69] R. Najam and S. Faizullah, “Analysis of Recent Deep Learning Techniques for Arabic Handwritten-Text OCR and Post-OCR Correction,” *Applied Sciences*, vol. 13, no. 13, p. 7568, 2023, doi: 10.3390/app13137568.

[70] W. Albattah and S. Albahli, “Intelligent Arabic Handwriting Recognition Using Different Standalone and Hybrid CNN Architectures,” *Applied Sciences*, vol. 12, no. 19, 2022, doi: 10.3390/app121910155.

[71] T. Mullaney, “Building My Own Google Hangouts ‘Autocomplete’ with Char-RNN.” 2016. [Online]. Available: http://tommymullaney.com/projects/char-rnn-gchat

[72] P. W. Code, “CTC Loss Explained.” [Online]. Available: https://paperswithcode.com/method/ctc-loss

[73] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 369–376, 2006, [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.75.6306

[74] A. Hannun, “Sequence Modeling with CTC,” *Distill*, vol. 2, no. 11, 2017, doi: 10.23915/distill.00008.

[75] G. Agrawal, S. Taqvi, and R. Gulati, “Machine Learning with TensorFlow and PyTorch: A Comparative Analysis,” *RES MILITARIS*, vol. 10, no. 1, pp. 172–179, 2020.

[76] NVIDIA 2020, “NVIDIA GeForce RTX 3060 Family.”

[77] “Data-centric AI: A complete primer,” *Snorkel AI*, May 17, 2022. [Online]. Available: https://snorkel.ai/data-centric-ai-primer/

[78] M. A. Khan, “Arabic handwritten alphabets, words and paragraphs per user (AHAWP) dataset,” *Data Brief*, vol. 41, p. 107947, Jul. 2024, doi: 10.1016/j.dib.2022.107947.

[79] S. Raschka, “Training-validation-test split and cross-validation done right.” 2020. [Online]. Available: https://machinelearningmastery.com/train-test-split-and-cross-validation-for-model-evaluation/

[80] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*, vol. 6, no. 1, p. 60, 2019, doi: 10.1186/s40537-019-0197-0.

[81] Inc. Kolena, “Model Training in AI/ML: Process, Challenges, and Best Practices,” *Kolena*, 2023, [Online]. Available: https://www.kolena.com/blog/model-training-process

[82] Dataheroes.ai, “Machine Learning Model Training,” *Dataheroes*, 2023, [Online]. Available: https://www.dataheroes.ai/machine-learning-model-training

[83] ProjectPro.io, “How to Train a Machine Learning Model: The Complete Guide,” *ProjectPro*, 2023, [Online]. Available: https://www.projectpro.io/article/how-to-train-a-machine-learning-model

[84] Inc. Kolena, “Model Training in AI/ML: Process, Challenges, and Best Practices,” *Kolena*, 2023, [Online]. Available: https://www.kolena.com/blog/model-training-process

[85] T. with Kolena, “WER, CER, and MER: Measuring textual similarity with substitution, deletion, and insertion errors,” *Testing with Kolena*, 2024, [Online]. Available: https://docs.kolena.com

[86] M. Xiong *et al.*, *A Rate of Penetration (ROP) Prediction Method Based on Improved Dung Beetle Optimization Algorithm and BiLSTM-SA*. 2024. doi: 10.21203/rs.3.rs-4255057/v1.

[87] C. Pavlatos, E. Makris, G. Fotis, V. Vita, and V. Mladenov, "Enhancing Electrical Load Prediction Using a Bidirectional LSTM Neural Network," *Electronics (Basel)*, vol. 12, p. 4652, Jul. 2023, doi: 10.3390/electronics12224652.

[88] J. H. Noor, "The Effects of Architectural Design Decisions on Framework Adoption: A Comparative Evaluation of Meta-Frameworks in Modern Web Development," Aalto University, Espoo, 2024.

[89] Meta, "React | Meta Open Source." 2024.

[90] Kinsta, "What Is React.js? A Look at the Popular JavaScript Library." 2023.

[91] D. E. V Community, "ReactJS: A Comprehensive Overview of a Popular JavaScript Library." 2023.

[92] T. Rascia, "React Tutorial: An Overview and Walkthrough." 2023.

[93] "Comparison of FastAPI with Django and Flask," *GeeksforGeeks*, Nov. 04, 2023. [Online]. Available: https://www.geeksforgeeks.org/comparison-of-fastapi-with-django-and-flask/

[94] M. Tan and Q. V Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[95] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[96] 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.

[97] V. Adithya *et al.*, "EffUnet-SpaGen: An Efficient and Spatial Generative Approach to Glaucoma Detection," *J Imaging*, vol. 7, p. 92, May 2021, doi: 10.3390/jimaging7060092.

[98] T.-H. Nguyen, T.-N. Nguyen, and B.-V. Ngo, "A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of

Tomato Leaf Disease," *AgriEngineering*, vol. 4, pp. 871–887, Jul. 2022, doi: 10.3390/agriengineering4040056.

[99] T. Hoeser and C. Kuenzer, "Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends," *Remote Sens (Basel)*, vol. 12, Jul. 2020, doi: 10.3390/rs12101667.

[100] S. V. Iyer, "Implementing Analog Filters Digitally," in *Digital Filter Design using Python for Power Engineering Applications: An Open Source Guide*, S. V. Iyer, Ed., Cham: Springer International Publishing, 2020, pp. 71–111. doi: 10.1007/978-3-030-61860-5_5.

[101] BotPenguin, "Transfer Learning: Techniques & Algorithms," *BotPenguin*, Oct. 2023.

الملخص

تم تنفيذ هذا المشروع "التعرف على الكتابة اليدوية للغة العربية" من قبل فريق مكون من 11 طالب في كلية الهندسة قسم حاسبات وتحكم آلي بجماعة قناة السويس تحت إشراف د. خالد عبد السلام. الهدف الأساسي من المشروع هو تطوير نظام يستطيع التعرف على النصوص والعربية وتحويلها رقميًا. تم تصميم النظام لتحويل مختلف الأنواع من المقالات، مثل المقالات المصورة، ملفات PDF، أو صور تم التقاطها بالكاميرات الرقمية، إلى بيانات بإمكاننا تعديلها والبحث فيها بمرونة.

تضمن المشروع عدة مراحل، من تجميع البيانات اللازمة وتحضيرها، مرورًا بالتخطيط للنظام وتصميمه، ثم تطبيق هذا النظام، إلى مراحل تدريب الذكاء الاصطناعي وتحليل النتائج. استخدم الفريق العديد من التقنيات والتكنولوجيا، مثل معالجة الصور، التعرف على الحروف العربية، الشبكات العصبية الملتفة (CNN) ، الشبكات العصبية المتكررة (RNN)، والجوريزمات التعلم العميق.

على الرغم من مواجهة الكثير من التحديات مثل تعقيدية كتابة اللغة العربية، الاختلاف في طريق الكتابة والخطوط، وقلة توفر المصادر العامة والمفتوحة لبيانات الكتابة اليدوية باللغة العربية، استطاع الفريق تطوير نظام حاز على مستويات عالية من الدقة والمتانة في التعرف على الكتابة اليدوية للغة العربية. لا يساهم المشروع فقط في مجال "التعرف البصري للحروف (OCR)" ولكنه أيضًا يفتح آفاق جديدة للبحث المستقبلي والتطوير في تخصص التعرف على الكتابة اليدوية للغة العربي

جامعة قناة السويس
كلية الهندسة
قسم حاسبات وتحكم آلي

التعرف على الخط اليدوي للغة العربية

تقرير مقدم كمتطلب مشروع التخرج للحصول على درجة البكالوريوس في قسم الحاسبات والتحكم الآلي بكلية الهندسة جامعة قناة السويس

تحت إشراف

د. خالد عبد السلام

مقدم من

أيمن صابر جاد محمد
أحمد طه أحمد عبد الرحيم
أحمد نجاح محمد أحمد
محمد فتحي محمد محمد علي
أبانوب عايد خلف
ريم فؤاد محمد
روان جمال محمد
ندى محمود محمد
ندى حسين عصران
كيرلس سمير فتحي
محمد عبد الفتاح فتحي