Department of Computer Science
COMP4300- Graduation Project
Second Semester 2023/2024

KhattClarifier:

an Arabic Handwritten Offline Recognition System (AHR)

Izzat Almustafa, 1202444

Supervisor:

Dr. Ahmed Abusnaina

Title of Project:

KhattClarifier: an Arabic Handwritten Offline Recognition System (AHR)

Project No:

32

Supervisor:

Dr. Ahmed Abusnaina

Key Areas:

Artificial Intelligence, Computer Vision, Arabic Handwritten Recognition (AHR), Arabic Offline Intelligent Character Recognition (ICR), Deep Learning, Neural Networks.

Student Signature: Izzat Almustafa **Date Submitted:** 24.6.2024

First Supervisor Name: _____

First Supervisor Signature: _____

Date Approved: ____ / ____ / _____

KhattClarifier:
an Arabic Handwritten Offline Recognition System (AHR)

Key Areas:
Artificial intelligence, Computer Vision, Arabic Handwritten Recognition (AHR), Arabic Offline Intelligent Character Recognition (ICR), Deep Learning, Neural Networks.

Abstract

A website that uses an Arabic Handwritten Recognition (AHR) offline system is proposed. This web site will use a web application as a front that takes images of handwritten Arabic text and sends them to an Intelligent Character Recognition (ICR) system. This system in turn will extract any Arabic words found, convert them to printed text, and return the converted text to the web application. The ICR system used for this app is an enhanced implementation of the Optical Character Recognition (OCR) technology, where ICR is used to recognize handwritten text and OCR is used for typewriting.

The ICR system in AHR will use pre-processing techniques to remove excess noise in the image, segment the words and perform any necessary action before recognizing the word. The ICR system will use Neural Networks, an algorithm used in deep learning, to distinguish and recognize characters and ligatures. By using this app, those who require a product to convert handwritten Arabic text to printed text can easily complete their task by using this website.

The AHR system will rely on a Deep Convolutional Neural Network (DCNN) with a Bi-Directional Long Short-Term Memory (BiLSTM) neural network and Connectionist Temporal Classification (CTC) loss to perform the recognition of the Arabic words. Several variations of the AHR were tested to try to find the variation the yielded the highest accuracy. Finally, improvements on the AHR were discussed for opportunity of future expansion upon it.

Acknowledgment

I thank Dr. Ahmed Abusnaina for helping me with this project. He provided all the help I needed as soon as I needed it. I thank him for his patience and his insightful suggestions.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

Chapter 1 shows the aims and objectives of the project. In addition, the introduction and the motivation of the project. At the end of it, an overview of the report is provided.

1.1 Overview and Motivation

We live in a digital age, where the need for pattern recognition has become increasingly in demand. Arabic Language, as one of the oldest and most spoken languages in the world, has a rich handwritten heritage (Britannica, 2023). Despite that, a form of marginalization was produced towards the Arabic language for the benefit of English particularly in Technology and information retrieval. It is believed that directing attention to Technology will reduce this marginalization. For instance, an offline ICR system for handwritten Arabic Language documents can be an opportunity, given that the ICR systems are utilized by search engines that optimize Arabic resource access. However, the unique characteristics of the Arabic Language make it challenging for non-Arabic speakers to learn to read, let alone for a machine.

Arabic Language Characteristics

The Arabic Language has 28 letters, some of which have the same structure but differ in an added dot, two, or three, the position of those dots may also vary. Take For example the letters: ب، ت، ث (Ba'a, Ta'a, Tha'a). Each of the 28 letters has multiple ways of being written depending on its position in a word. Linked letters connect to the characters previous to and ahead of it, in the same word. For instance, the letter ب (Ba'a), when appears at the beginning of a word بحر (sea) or in the middle of a word حبر (Ink), or at the end of a word like in the word حب (Love). Separated letters are connected to the letters previous to it but not to the ones in front of it, in the same word. An example would be ر (Ra), where in the word حرب (War) it's connected to the letter ح (Hha) and separated from the letter ب (Ba'a).

Handwritten Arabic is naturally cursive. Since Arabic characters can be linked together and there are many variations for each character, some handwriting styles can make it difficult to distinguish each character from another. This can lead to characters overlapping vertically, which

means two characters may exist in the same vertical line as seen in Figure 1.1 below. In extreme cases, characters in different words may overlap. Due to these factors, there are 84 variations for Arabic characters.

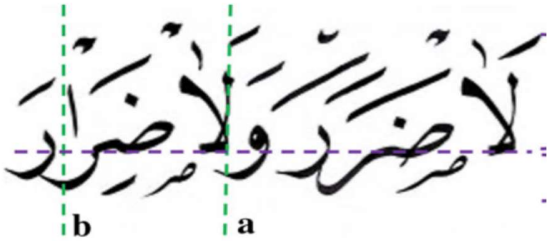

Figure 1.1: Vertical overlapping (a)between two words (b)in one word (Albahli & Alrobah, 2022)

Arabic Handwritten Recognition (AHR)

The technology used in this project is called ICR. The context of this system is Arabic Handwriting, hence the Technology of OCR cannot be used, because the latter focuses on typewritten recognition, while ICR is designed for handwritten text recognition. As presented above, the characteristics of the Arabic language make (ICR) challenging to implement.

ICR uses techniques such as recognition and segmentation. The recognition process is done by analyzing and converting the text in an image into machine-encoded text. Segmentation is the process of segmenting the units of a text: line, word, character, and ligatures. For example, the word عزات can be segmented into [عز,ا,ت], where each element of the set is a character or a subword called a ligature (also called Part of Arabic Words PAWs (Albahli & Alrobah, 2022)) such as عز.

KhattClarifer Software

Khatt (خط) is the Arabic word for writing style, hence the name of the software being KhattClarifier combining the words Khatt and Clarifier. KhattClarifier is a web application that is designed to be usable for a wide range of users from IT experts to those with no IT background. This app could be used in many fields for digitizing, documentation, and other uses.

For instance, teachers use our system to assist them in reading their students' handwritten papers. Some students struggle to understand lecture notes that they wrote in a hurry, this system will be able to help them analyze the notes written. A general usage of KhattClarifier is making the image searchable, which assists with information retrieval and is editable, saving time and effort.

1.2 Aim and Objectives

1.2.1 Aim

To develop a web application, with an Intelligent Character Recognition (ICR) System that performs Arabic Handwritten Recognition (AHR) and returns the recognized Arabic text in the form of printed text.

1.2.2 Objectives

1.2.2.a Research Objectives

1. **Arabic language Characteristics**: To address the Arabic Handwritten Language's challenging nature and characteristics.
2. **Relevant theories:** To investigate the relevant theories.
3. **Literature review:** To review the existing literature on the matter including papers about
    a. Arabic Handwritten datasets
    b. Arabic Handwritten Image Preprocessing
    c. Arabic Handwritten Image segmentation techniques.
    d. Arabic Handwritten Recognition (AHR)
    e. Arabic Handwritten recognized text post processing techniques.

1.2.2.b Deliverables Objectives

1. **Software Requirements Specifications Document**
    a. To specify system requirements: functional and non-functional.
    b. To describe the actors and the use cases.

c. To analyze the system including class, sequence, state chart, and activity diagrams.
d. To design the system including dividing the system into subsystems and deployment diagram.

2. **Web Application (KhattClarifier)**
    a. **Accessibility**: To develop a web application accessible to a wide range of users.
    b. **User Interface:** To design a user-friendly interface.
    c. **Testing:** To test the web application.

3. **Arabic Handwritten Recognition (AHR)**
    a. **Image preprocessing:** To implement Image preprocessing techniques in order to enhance the input image quality.
    b. **Image segmentation:** To implement Image segmentation techniques in order to identify the text subunits [lines, words, ligatures].
    c. **Model Training and Testing**: To train and test an AI model on the KHATT Dataset in order to achieve high recognition accuracy of AHR.
    d. **Recognition**: To obtain the recognized test from the model based on the user image input.
    e. **Integration**: To integrate the ICR system into the web application.

1.3 Overview of the Report

This report is divided into 6 chapters. In chapter one, the characteristics of the Arabic language, the AHR system, and KhattClarifier software as well as my aims and objectives are presented. Chapter two covers the project's background and literature review. Chapter Three will discuss our approach, resources, and system analysis. Chapter Four covers the methodology used for this project. Chapter Five covers the results of the AHR system. Chapter 6 covers the discussion of the project where future improvements are suggested.

Chapter 2 Background

Chapter 2 is the background chapter, in which the relevant theories are discussed and then the literature review on the matter is done. Lastly, the resources used in the project are shown.

2.1 Relevant Theory

Artificial intelligence (AI)

In his article "What is AI?" McCarthy defines AI as "*the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.*" (McCarthy, 2007)

But what is intelligence? "*Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals, and some machines.*" (McCarthy, 2007). AI is the actual backbone of the KhattClarifier web application, where **the intelligent task** is to mimic the human ability to read and recognize Arabic Handwritten images.

PEAS Representation

In AI, the PEAS (**P**erformance Measure, **E**nvironment, **A**ctuators, **S**ensors) representation is used to define the task environment. The performance measure describes what utility the agent tries to optimize. The environment describes where the agent acts and what affects the agent. The actuators and the sensors are the methods by which the agent acts on the environment and receives information from it (Sharma, 2023). Given the intelligent goal defined earlier, the PEAS definition is:

- **Performance Measure:** The accuracy of the AHR system in recognizing the handwritten Arabic text in images and converting them into machine readable.
- **Environment:** The users' context.
- **Actuators:** The User Interface (UI) elements like buttons, and the recognition model.

- **Sensors:** The screen and text inputs in the UI.

Computer Vision

If AI enables the computer to think, Computer Vision enables them to see, observe and understand. An ICR system is a well-known application in the field of Computer Vision. Computer Vision is considered to be a field of AI. It uses techniques that are used to interpret visual information and extract meaningful information from them (IBM, 2023). This being interpreting digital images of an Arabic Handwritten to extract printed texts in this project.

Machine Learning (ML)

ICR is also one of the Machine Learning applications. ML is a powerful AI tool. It starts when training a model on a dataset of labeled Arabic handwritten characters, to distinguish the features that define each of them. Once trained, the model is able to recognize handwritten Arabic. The most used in Character Recognition are based on ML specifically **deep learning** including Neural networks. In Deep Learning a neural network (NN) is used for detecting and recognizing the characters. It's composed of two categories: supervised and unsupervised learning. Supervised learning is when a machine learning model is given the correct answer for the data it is trained upon. Based on what it predicts and the actual correct answer it tries to figure out a pattern between the inputs and the output. Unsupervised learning is when a machine learning model is not given the correct answer for the data it is trained upon. Instead of knowing what the correct answer is, it tries to determine patterns based solely on the data it is given.

Neural Networks (NN)

A neural network is a supervised machine learning structure that is used for categorization which mimics the way a human's brain learns. A neural network consists of an input layer, an output layer and in most cases a hidden layer. Each layer is made up of neurons. A neuron is a small structure that takes in a value as an input, applies a formula to it and outputs the result. The layers are connected to each other by connections to each other's neurons, where each connection has a weight to it. As data passes from layer to another along to another layer, the formulas in the neurons are applied to data incoming from the previous layer and passed on

to the next layer, where the formula takes into account the weight of the current connection. A formula may look similar to this:

$$y = xw$$

Where x is the input data, w the weight of the connection and y the result that will be outputted. At the final layer, the output layer, each neuron will be given a category to represent. The final value exiting each of the final neurons will show how confident the neural network is that the data belongs to that category. A neural network can be visualized like this:

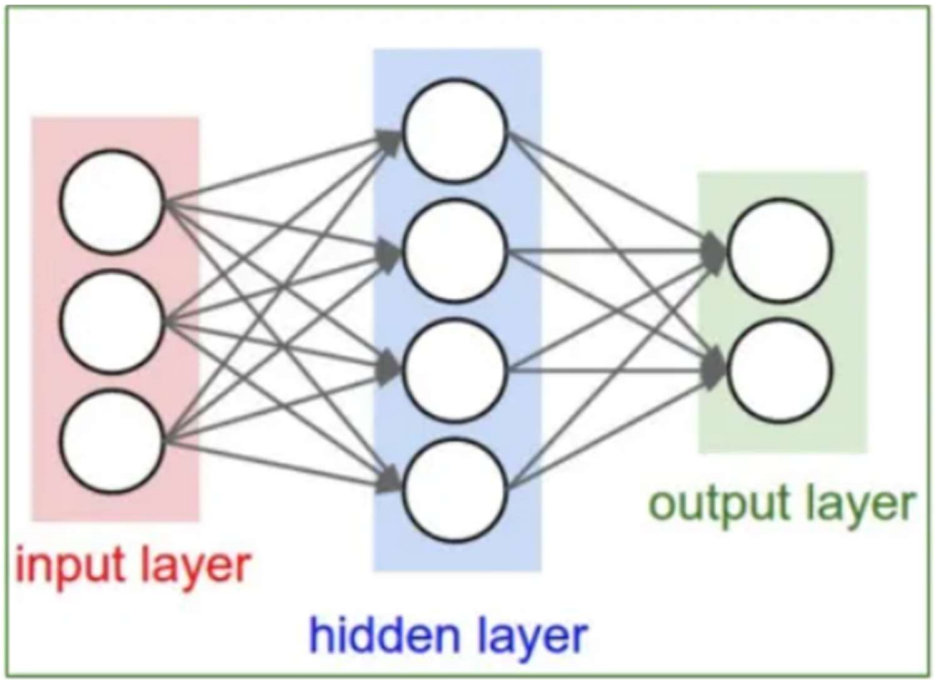

Figure 2.1: Visualization of a neural network

A neural network will optimize its structure so that it gets better at categorization through the backpropagation and gradient descent processes. A neural network with multiple hidden layers is called a deep neural network.

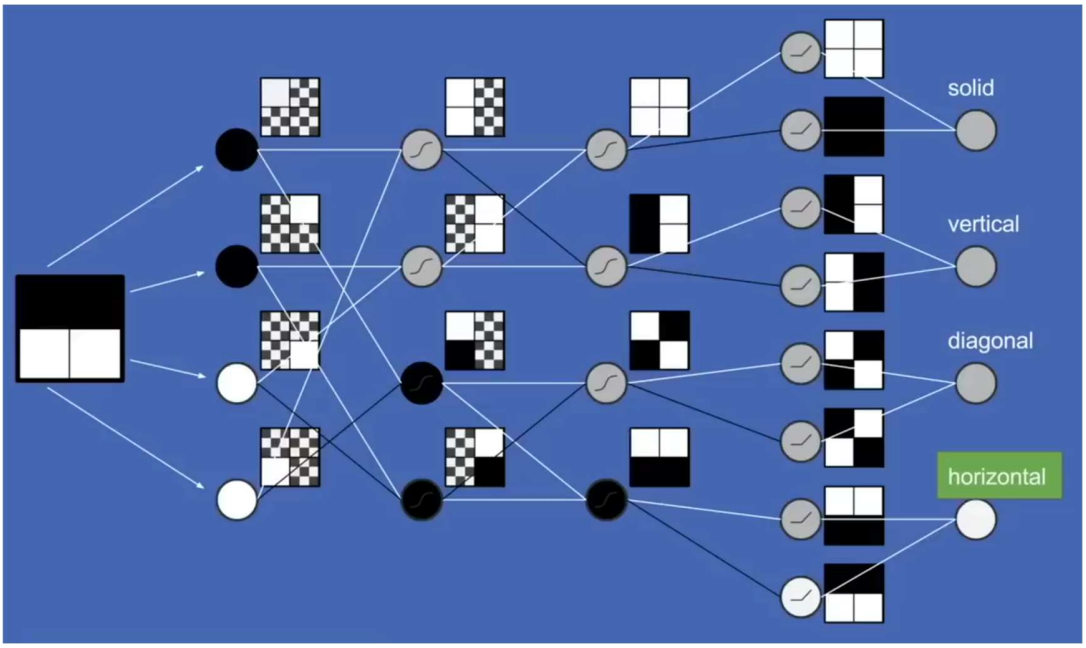

Figure 2.2: Example usage of a neural network

Backpropagation

During the training process of a neural network, data is passed throughout the network going through the designated neurons based on the connections. This process is called feed forward. The outputs of the neural network are compared to the ground-truth values based on an error function, where an error rate is outputted. The goal of the training process is to minimize the error. Changing the weights of the connections in the model can help with improving the accuracy of the neural network and reduce the error. The way the weights should be changed is calculated based on the change in error in regards to the change in weight. If the error increases when the weight is increased, then the weight should be decreased and vice versa. The change in error due to the change in weight is called a gradient, so it can be said the weights are changed based on the calculated gradient. It can be visualized as follows:

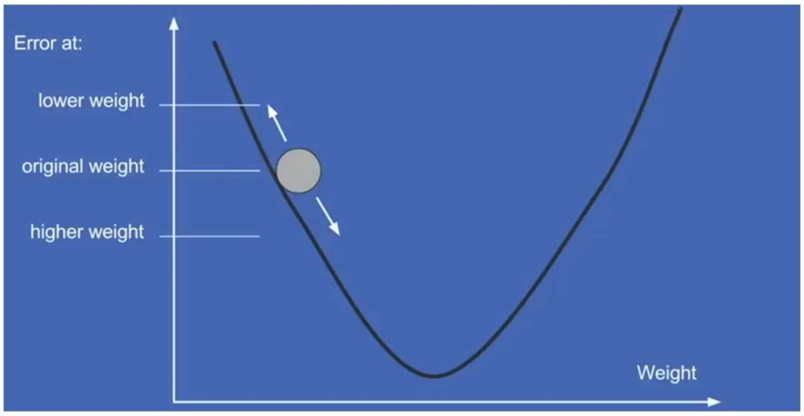

Figure 2.3: Minimizing the error of a neural network

Where the bottom of the plot is the optimal minimized error, the ball is the current weight and the arrows are possible gradients. Usually, the gradients are calculated by applying the chain rule to all the neurons throughout the entire neural network but there are other techniques too. In short, backpropagation is the process of finding the direction and how much the ball should move.

Gradient Descent

If backpropagation is finding where the ball should move, then gradient descent is the process of moving the ball. Gradient descent takes the gradients found during backpropagation and applies them to all of the weights in the neural network. The processes of backpropagation and gradient descent work hand in hand with the entire data set until the neural network is done training.

Deep Learning

Deep learning is the process of using deep neural networks to perform tasks that are too complex for normal machine learning models. Deep learning needs a lot more data and computational power than traditional machine learning. Most applications of neural networks in

the modern time use some form of deep learning such as Chat-GPT and others. Deep learning is considered to be a subset of machine learning.

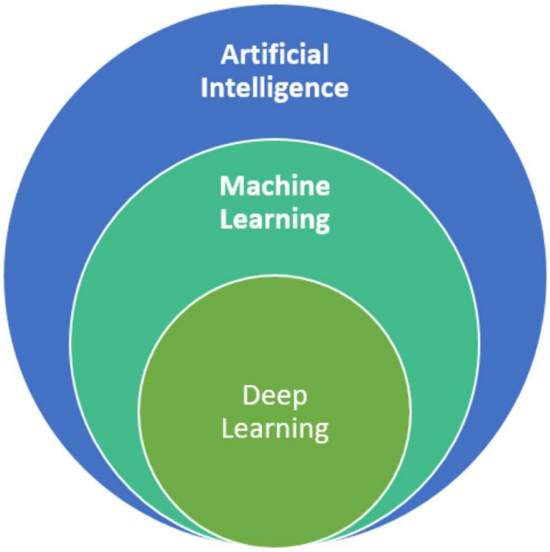

Figure 2.4: Diagram of how the different fields of AI are related to each other.

Convolutional Neural Network (CNN)

While normal neural networks are great for predicting a function's outputs or for simple problems, it's not ideal for images. When training with images, a neural network will memorize patterns belonging to that specific image. So, if the same image is transformed slightly, (rotated, cropped, shifted etc.) it will have difficulty recognizing that same image. A solution for this was to use a Convolutional Neural Network (CNN).

A CNN has the ability to recognize local patterns in 2D data, such as images. Instead of finding patterns in an overall image, creates a small matrix (usually 2x2 or 3x3) called a kernel. The kernel usually contains the information of a specific, local feature. The kernel is placed at the beginning of the image (usually top left) and performs a matrix multiplication with the values of the image and the values in the kernel. The results get stored. The kernel then shifts along the row it's on and performs the matrix multiplication again, until it reaches the end of the row. Once it finishes moving across the row, it shifts downwards and goes to the beginning of the image and repeats this process for the entire image. The process is called convolution. The result of the convolution process is a matrix of values that contains likely values corresponding to the

values in the kernel, also called a feature map, which indicate the most likely positions of a specific feature (stored in the kernel). The raises the question, how are the values in the kernel chosen? The values in the kernel are found during training and specifically through backpropagation. The layer in the responsible for performing the convolution process in a CNN is called the convolution layer. Most convolution layers use more than one kernel, which means that they usually lead to having lots of feature maps per convolution layer as each kernel generates a feature map.

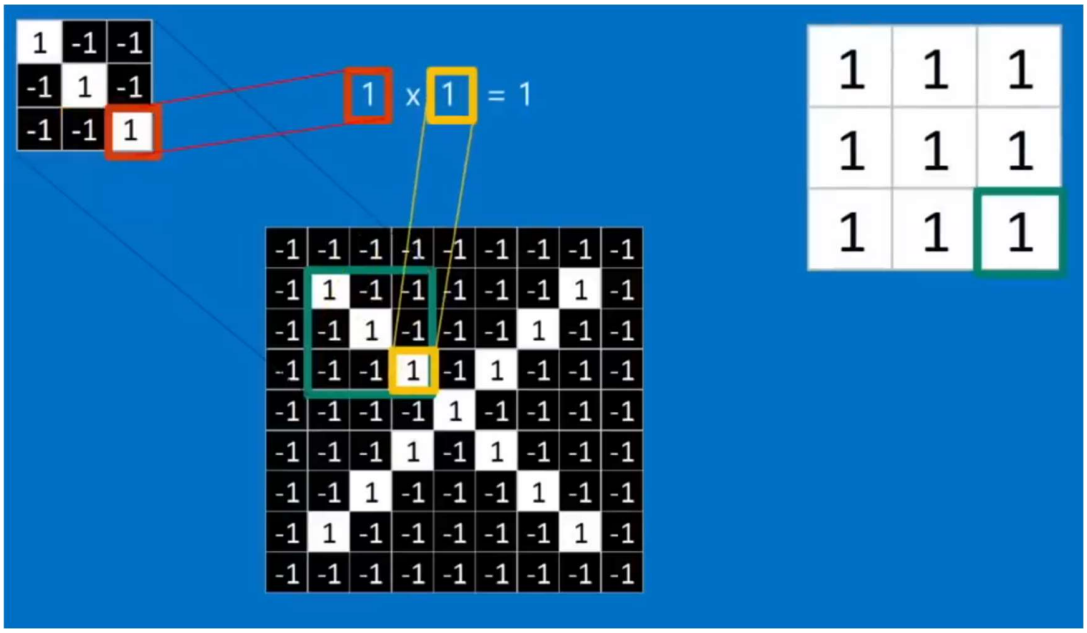

Figure 2.5: Example of how matrix multiplication is used during convolution

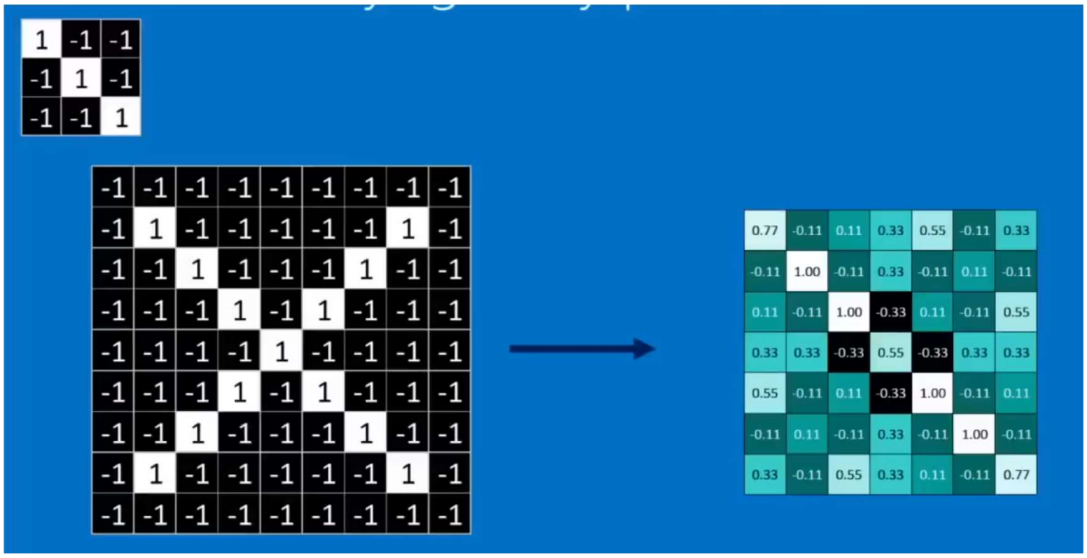

Figure 2.6: Example of how a feature map is generated after the convolution process

The next step in a CNN is to take the feature map and try to reduce it. This is done through a process called pooling. Pooling takes a small window (usually 2x2) and moves it along the feature map, similar to the convolution layer, where it performs one the pooling operations. There are 3 main types of pooling: max pooling, min pooling and average pooling. Max pooling takes the maximum value found in the window. Min pooling takes the minimum value found in the window. Average pooling calculates the average of the values in the window. Usually, max pooling is used. In the end, it results in an even smaller and more precise feature map, which can help indicate further the most likely positions of the feature. The pooling process is repeated for every feature map done for every kernel. The layer in the CNN responsible for doing the pooling process is called the pooling layer.

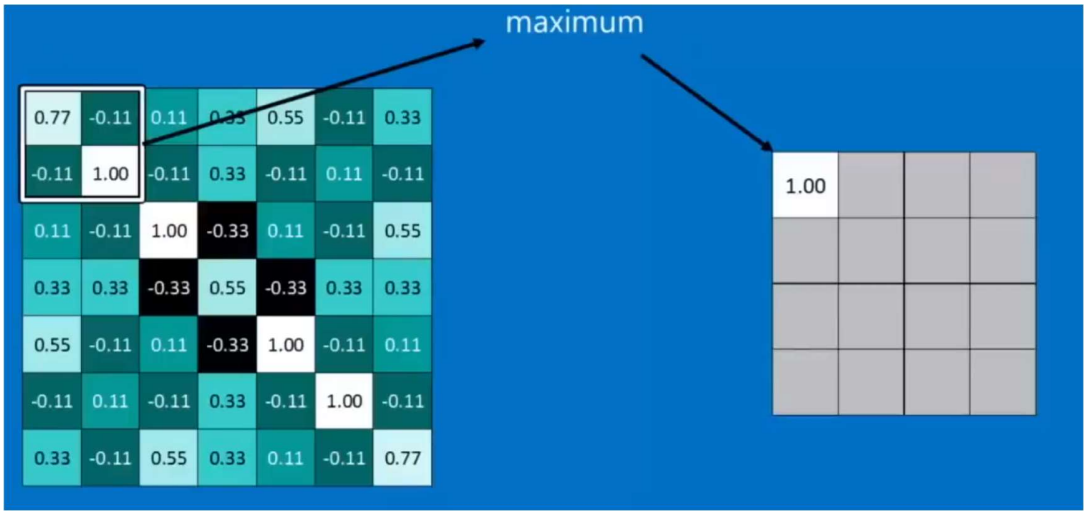

Figure 2.7: Example of how max pooling is applied to a feature map

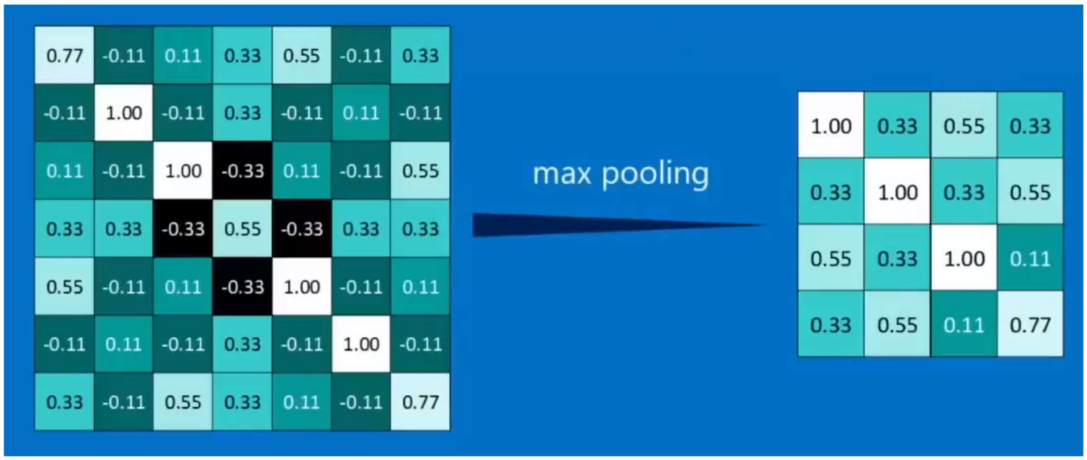

Figure 2.8: Example of the feature map outputted of the pooling process

The convolutional and pooling layers can be repeated as many times as needed in the CNN until the desired size of the feature maps is found. Having multiple convolutional and pooling layers is called deep stacking, it makes a normal CNN a Deep CNN (DCNN). Using certain activation functions such as “relu” or “sigmoid” between the convolution and pooling can help normalize the data, which helps with avoiding issues with numeric values. The final feature map is taken to a normal neural network for classification.

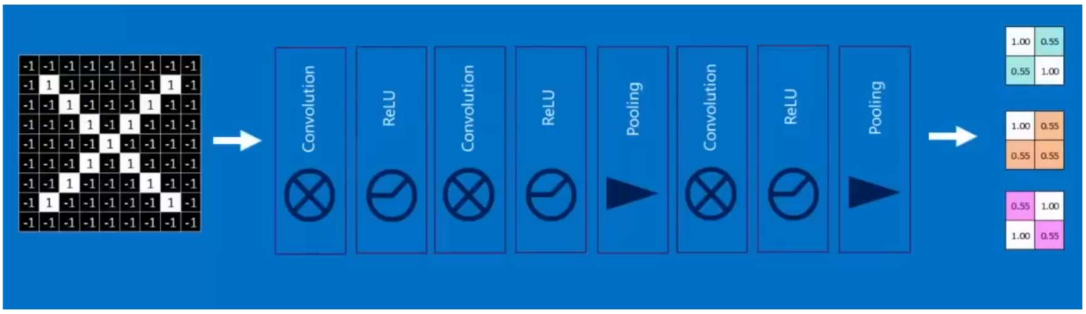

Figure 2.9: Example of deep stacking

Recurrent Neural Network (RNN)

While NNs are great for general classification and CNNs are great for images, they both fail at tasks that require sequential translation, such as time-based actions and using languages. RNNs were created to solve this issue. They are primarily used for sequence-to-sequence translation such as Text-to-Speech or language translation. An important version of RNN is the Long-Short Term Memory (LSTM) which will be discussed in the next section. RNNs are basically a neural network that work by taking new information with old predictions to make a prediction with an activation function (usually the tanh function) at the end for data normalization. This allows for learning sequential patterns in data.

RNNs can be utilized to predict the next action in a sequence given the previous and current actions are given. Utilzing this, an entire sequence can be generated by unrolling a RNN so that it can use predictions from the far past to generate the prediction for the action that is supposed to come after it up until the current prediction for the next action. During this process, the same weights and biases are used in the RNN. The process of utilizing information from the far past to predict the current next action is called unrolling. The unrolling process is done only during the training process so the RNN can learn the relationships between the previous and the current action better. The number of times an RNN is unrolled depends on the context of the output, for example, if a sentence of 3 words is to be predicted, then the RNN would unroll 3 times.

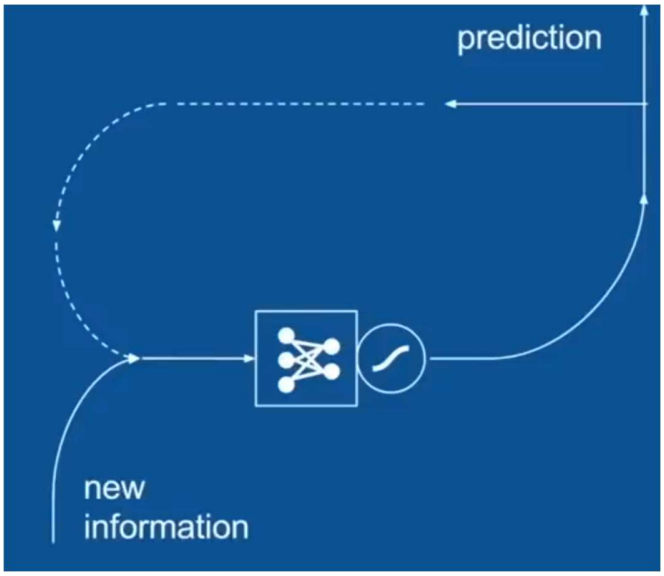

Figure 2.10: Basic Idea of a RNN structure

Basic RNNs only take in a single past prediction. But modern RNNs utilize something called memory to in order to take into account all previous predictions when predicting new information. Memory is basically another neural network that learns what is important to remember, with an activation function at the end of it (usually the sigmoid function) to normalize the data. The output of this neural network is combined with a gate to be able to forget the useless information, usually an OR gate is used. The remaining memory is combined with another gate with the prediction in order to create the final prediction, usually an AND gate is used here.

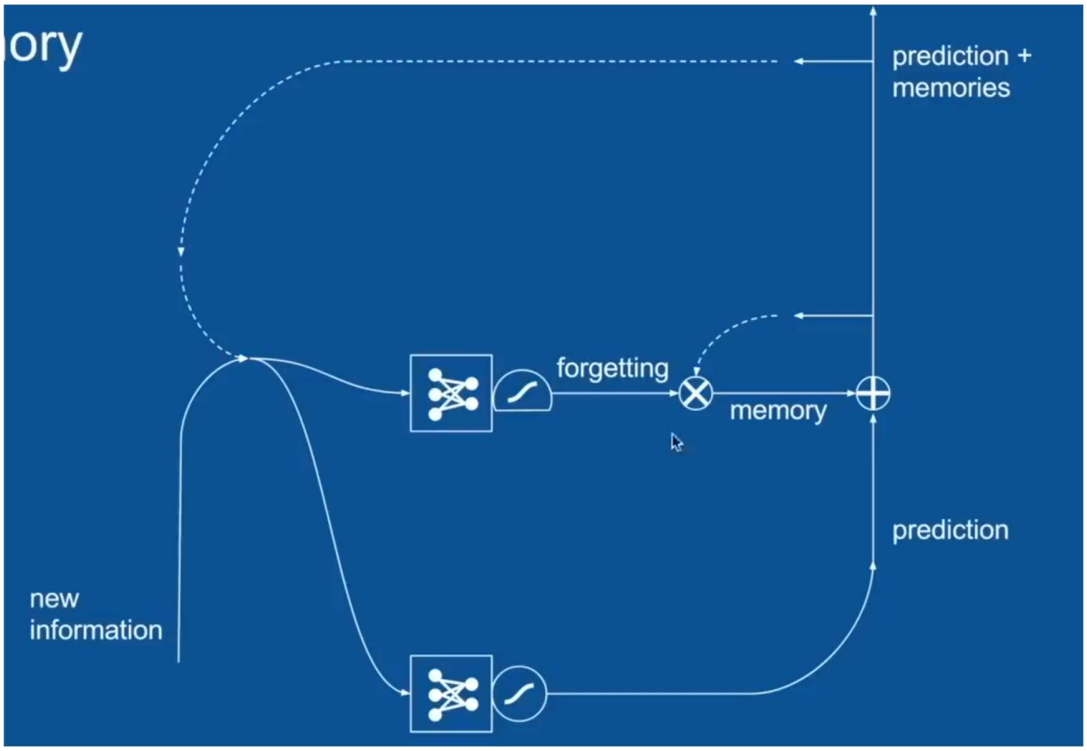

Figure 2.11: Basic illustration of an RNN that utilizes memory in its structure

But a problem arose with the memory system in an RNN. Gradients responsible for the memory, and in particular the unrolling parts, tend to increase or decrease exponentially. This is known as the exploding-vanishing gradients problem. Exploding gradients happen when weights are higher than 1, causing gradients to increase exponentially as the unrolling becomes longer and longer, making the initial data have too much effect on the prediction. The huge gradients make it almost impossible for the weights to converge. Vanishing gradients occur when the opposite happens, the weights are less than 1, causing gradients to decrease exponentially as the unrolling becomes longer, making initial data barely have any effect on the prediction. The tiny gradients make it almost impossible for the weights to converge before the maximum steps for RNN to finish training have been reached. There are many solutions to these issues, but the main one used was to move to LSTM networks.

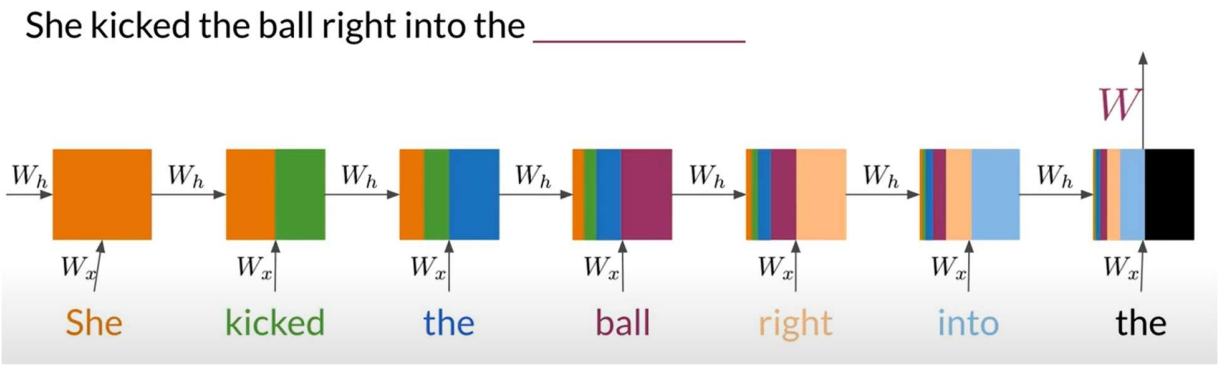

Figure 2.12: Example of the effect of vanishing gradients

Long-Short Term Memory (LSTM)

The difference between the LSTM and RNN occurs in the unrolling process. There are two major additions, first, instead of using a single path to transfer data between each cell in the feedback loop in the RNN, there are two new paths that the LSTM utilizes: the long-term memory path and the short term-memory path. Second, instead of relying on a simple cell like in the RNN, the LSTM uses a more complex cell with three main gates.

Figure 2.13: The two paths of LSTM, green path being long-term memory and the blue path being the short-term memory.

The new complex cell introduces these three gates: forget gate, input gate, output gate. Each of these three gets are connected to the short-term memory, long-term memory and input

through weights connections, biases on those connections, sum gates, multiplication gates and different activation functions. The forget gate is responsible for determining how much of the long-term memory should be forgotten, based on values of the short-term and input. The input gate is responsible for finding the potential long-term memory that can be added to the current long-term memory as well as the percentage of the potential long-term memory to remember. The output gate is responsible for determining the potential short-term memory to remember as well as the percentage of that long term member to remember.

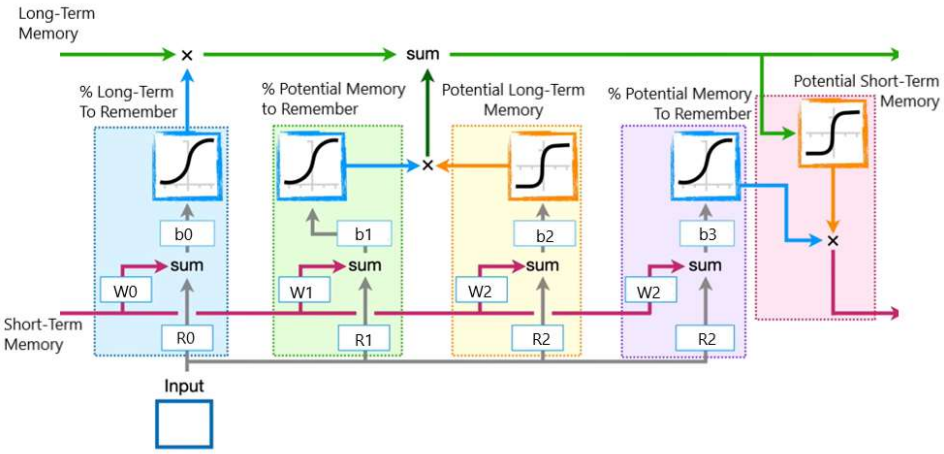

Figure 2.14: Diagram of the LSTM cell, blue being the forget gate, green and yellow the input gate and the purple and pink being the output gate.

A special version of the LSTM is the Bi-directional LSTM (BiLSTM). While LSTMs are great for predicting information given past information, it fails to take into account future information. Take for example the following scenario:

Figure 2.15: BiLSTM vs LSTM scenario

An LSTM would only take in the first part of the sentence "Apple is" to make a guess and disregard any information after it, which would lead to making a guess that in most likelihood would be incorrect. However, BiLSTMs take the first part as well as the second part of the sentence into account while making a prediction, thereby generating a more likely prediction.

BiLSTMs accomplish this by utilizing two LSTMs; one LSTM is used to learn patterns in the forward direction while the second LSTM learns patterns in the backwards direction. The outputs of each LSTM are aggregated before being sent to a Softmax layer for assessment.

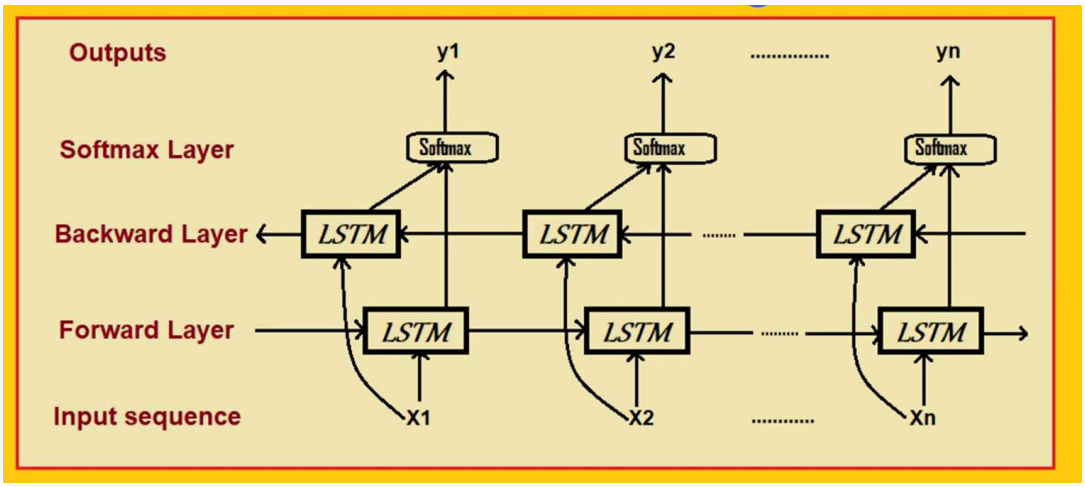

Figure 2.16: Basic Architecture of BiLSTM

Connectionist Temporal Classification (CTC)

CTC classification is used to match one sequence to another, especially when the two sequences don't match in length. The first sequence has to be longer than the second sequence. CTC is usually used for audio to text dictation. The basic idea behind CTC is that the first sequence is split into windows of a certain size that are each matched to an output. Sometimes, several windows will match to a singular output, so those windows outputs are collapsed into a single output. Other times, the collapsed output should collapse into several outputs instead of just one. So, a blank character was introduced to separately same outputs from being collapsed into one. An example of which is the following, where ε is the blank character:

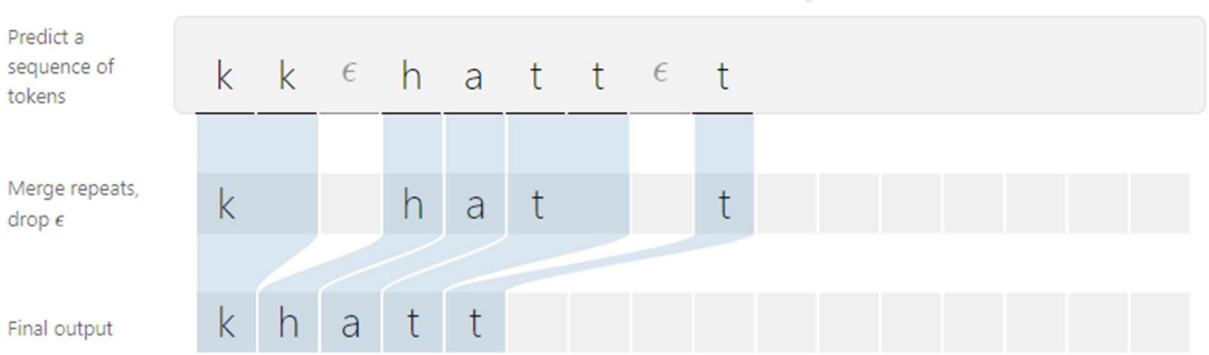

Figure 2.17: CTC example usage

The math behind, in short, basically generates every possible form of sequences using the alphabet of the sequence (including the blank character), with each sequence being considered a path, and finding the best possible path from the beginning of the sequence to the end of the sequence that matches the ground truth. Using dynamic programming, it can be optimized very highly. The goal of CTC training is to increase the possibilities of generating some path that collapses into the ground truth. The equation used to find the most likely path is for a given sequence is:

$$\theta = argmax \sum_{i=1}^{N} \log \sum_{\pi \in B-1(z)} p(\pi|xi;\theta)$$

Where π is the set of possible paths, xi is a sequence of inputs, B^{-1} is a function that maps label sequence z to a set of all possible label sequences (paths in π) that would eventually collapse to z and N is the number of paths. The formula used to calculate the CTC loss of a neural network is the following:

$$\theta = argmin - \sum_{i=1}^{N} \log \sum_{\pi \in B-1(z)} p(\pi|xi;\theta)$$

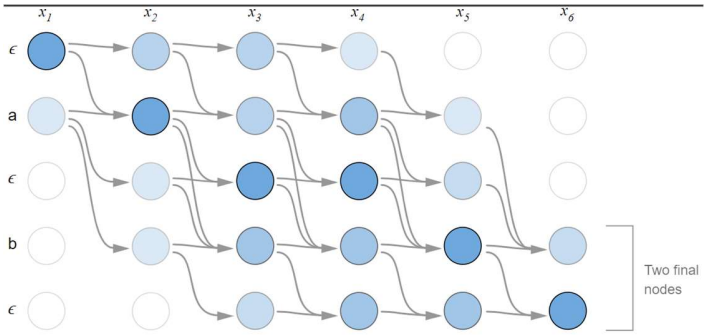

Figure 2.18: Finding all paths that lead to the ground truth before finding the best path.

At the end, the best possible path is found through the greedy search or the beam search, with the greedy search not always guaranteeing the best sequence of possible characters while the beam search being better but more computationally intensive. The best possible path is then collapsed and has its blank characters removed to output the final prediction.

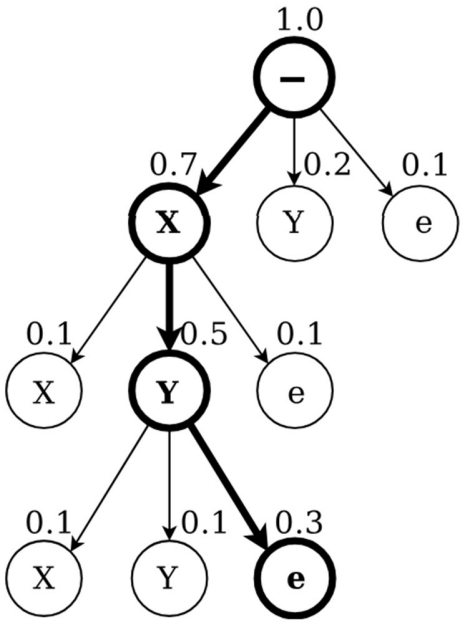

Figure 2.19: Using greedy search to find the best possible path.

Dilation

Dilation is a convolutional operation on an image used to increase the bright spots in that image. It first creates a kernel of a certain size, which is a matrix of ones. The kernel is placed over the image where the center of the kernel is over the first pixel. Each pixel is multiplied to the respective matrix entry the kernel is over, and the center pixel's value is to the maximum value found in the newly multiplied matrix. The kernel then slides to the next pixel and the same process is repeated. The kernel keeps sliding and repeating the process until it does this process over the entire image.

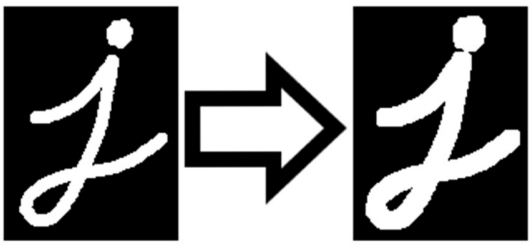

Figure 2.20: Example of how dilation affects an image.

Blurring

Blurring, also known as smoothing, is a convolutional operation used for removing noise and, using certain blurring techniques, make edges more profound. The most common blurring techniques are the bilateral filter and the gaussian filter. The bilateral filter works by taking a kernel of a certain size and placing its center over the pixel. The filter then takes the weighted mean of its neighboring pixels, giving different weights to neighboring pixels. It also does second calculation taking into account the differing intensities of the neighboring pixels.

Figure 2.21: Example of Bilateral Filter blurring

Thresholding (Binarization)

Thresholding refers to the process of converting an image to a black and white image. It's the simplest form of image segmentation. The term "threshold" comes from how it's decided which pixel is converted to a black or white pixel, where a certain value is chosen such that if a pixel's value is less then it the pixel's value will be set to 0 and if it's equal or greater than the value it's set to 255. The value is called the threshold. There are two main types of thresholding, global thresholding and adaptive thresholding. Global thresholding refers to setting a threshold value for all pixels of the entire image. Adaptive thresholding refers to setting a threshold value over a local value, which can change from one pixel to another. An example of adaptive thresholding is the mean adaptive thresholding, which creates a kernel and calculates the mean

value of the pixels within that kernel. The mean value becomes the threshold for the pixel. The kernel moves from pixel to pixel until all pixels are covered. Adaptive thresholding is used when there are differing bright and dark spots in the image.

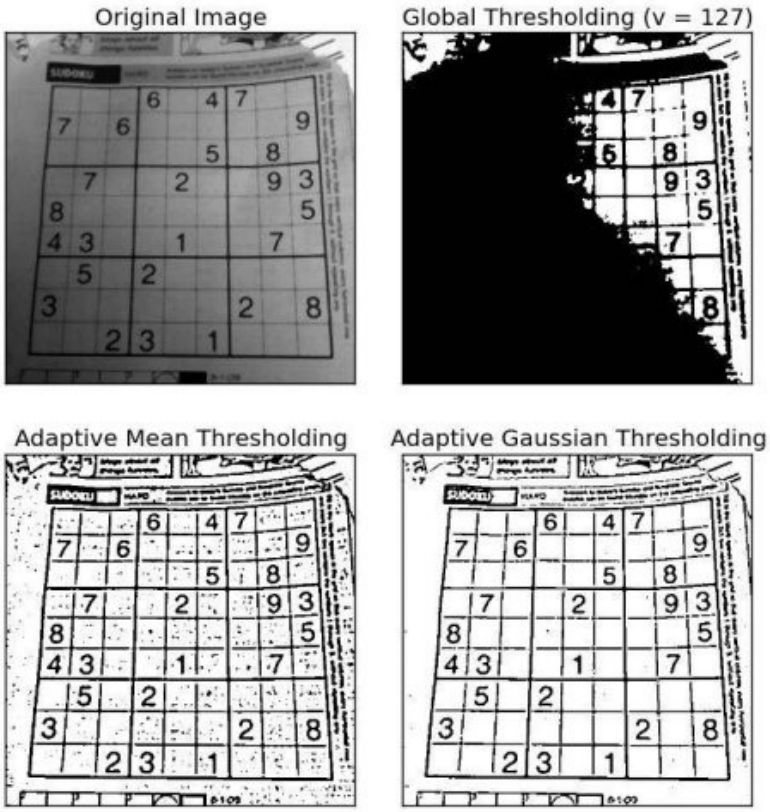

Figure 2.22: Examples of how different thresholding techniques work.

Activation Functions

Activation functions are math functions that take a certain value, and try to squash it down so that it can be within a certain range. Activation functions are great preventing neural networks from working with values that are too large, which helps reduce the computational overhead during training. Examples of activation functions are the hyperbolic tan (tanh), sigmoid and rectified linear units (ReLU) functions. The equations of each of these activation functions are:

Table 2.1: Activation functions' equations

Activation Function	Equation
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$
tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
ReLU	$f(x) = \max(0, x)$

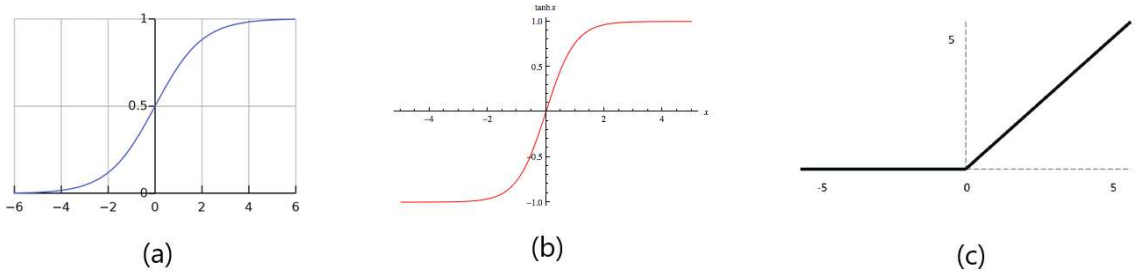

Figure 2.23: Plots of the activation functions, (a) being sigmoid, (b) tanh and (c) ReLU

Two-Stage VS End-to-end ICR

ICR systems can be categorized into (Two-Stages ICR) and (End-to-end ICR). The two-stage ICR is performed where the image-preprocessing is done, then the recognition step is separated into another following stage. On the other hand, an end-to-end ICR performs both the image preprocessing and the recognition step in one stage.

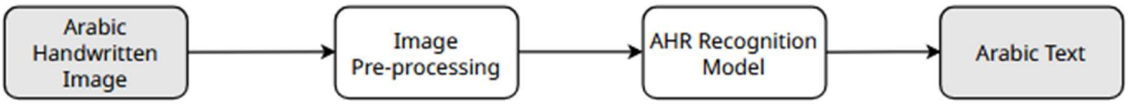

Figure 2.24: Two-Stages ICR

Offline VS Online ICR

The main differences between offline vs online ICR:

- Offline ICR does recognition on images, and takes time.
- Online ICR does recognition on text being written as soon as it's being written.

This means that offline ICR can be applied to document recognition, image recognition, recognition of videos etc. While Online ICR is applied to live hand writing, as in recognizing hand writing as it's being written, which are usually done on tablets that come with a pen.

ICR VS Traditional OCR

It is worth mentioning that Traditional Optical Character Recognition (OCR) and ICR are two technologies that significantly differ from each other. The following table shows the differences between the two: (Klippa, 2022)

Table 2.2: ICR VS Traditional OCR

ICR	Traditional OCR
Recognizes handwritten characters	Only recognizes characters in printed documents
Does not require templates	Template-Based
Uses AI and ML to extract and interpret data	Does not use AI and ML to extract data, relies on algorithms.
It is flexible and trained to recognize multiple formats	ideal for fixed structured documents
Upgrades its learning processes every time new data is being applied	Uses a specific format for data entry and doesn't have learning abilities
Can comprehend details and information	Can't comprehend information

2.2 Literature Review

i) 2.2.1 Arabic Handwritten Datasets

The IFN/ENIT database was the most cited among all. It contains 26,459 images of Tunisian city names and handwriting samples (Elb Abed & Margner, 2007) The problem with this database is the lack of variations, as it only includes names.

Another database that was reviewed is the CENPARMI (Al-Ohali, Cheriet, & Suen, 2003) database for Arabic Cheques. They were able to gather about 3000 real-world Arabic Cheque images. It includes the Arabic legal and courtesy vocabulary. This database also lacks variations of vocabulary since it is domain-specific. In addition, the SUST-ARG names dataset was reviewed. It only contained male Arabic names.

KHATT (Mahmoud, Ahmad, Al-Khatib, & Alshayeb, 2014) dataset stood out as they collected 1000 forms from 46 different resources covering 11 subjects. Each form contained two paragraphs, so in total they gathered 2000 paragraphs. The subjects varied ranging from Art, economy, society, technology, etc. This dataset had the highest number of writers reaching 1000 hence 1000 writing styles. They also varied in other factors such as Left-handed, right-handed, elementary, high school, female, male, and from different places: Palestine, Saudi Arabia, etc.

ii) 2.2.2 Arabic Handwritten Image Preprocessing

Regarding the image acquisition phase, it was found that collecting an image under certain environments could challenge the quality of the Arabic handwriting recognition process. Some images could have low resolutions or have surrounding white spaces. As shown in Figure 2.2.2.1 below, this input has low contrast, in binarization, it loses data.

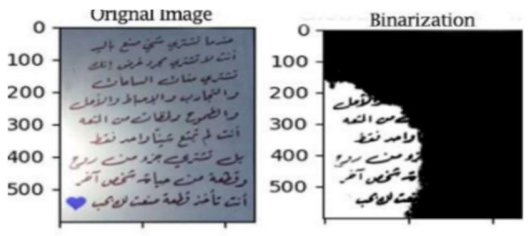

Figure 2.25: High contrast Arabic handwriting image (Boraik, Manjunath, & Saif, 2022)

Researchers (Boraik, Manjunath, & Saif, 2022) have used filtering techniques including Gaussian Filter, Median Filter, Low pass Filter, etc. But the results were not satisfying due to the unstable lightning, in contrast to when stable lightning was applied. KHATT database has binarization images, they transformed the images to a more useful form, which overcame the problem.

One of the interesting image preprocessing techniques found (Boraik, Manjunath, & Saif, 2022) was Signature Elimination and removal from an image. It was done depending on the connected components and finding out the average threshold.

iii) 2.2.3 Arabic Handwritten Segmentation

A survey of OCR in the Arabic Language (Faizullah, Ayub, Hussain, & Khan, 2023) was conducted and revealed the cruciality of the segmentation phase of an OCR system. This phase is concerned with segmenting the text into subunits including line, word, and character segmentation. It was found that there are various approaches to implementing that.

A study (Mousa, Sayed, & Abdalla, 2017) used a **projection-based technique** and was evaluated successfully, but it was limited to Arabic Printed text. In another study, **the divide and conquer algorithm** (Qaroush, et al., 2022) was used to segment the image into lines and process them line by line.

As for word segmentation, it was revealed to be challenging due to the cursive nature of Arabic, its overlapping, and its touching nature. Different techniques were used including **the spacing method, dictionary-based method, and Character-based**. (Faizullah, Ayub, Hussain, & Khan, 2023). Another approach was the **Hybrid approach**. This approach used different operators including **Morphological operators** (used when the image needs erosion and dilation when converting to binary) and **k-means** (machine learning). (Boraik, Manjunath, & Saif, 2022)

iv) 2.2.4 Arabic Handwritten Recognition

Various Recognition models were explored since it is a core phase in an ICR system. A comprehensive survey was conducted that covered 21 different studies 5 of which used Convolutional Neural Networks (CNN) for feature extraction and 16 used classical Machine Learning (ML) for feature extraction. The studies relied on 4 main databases: IFN/ENIT, AHDB, SUST-ARG names, and KHATT. This survey's introduction helped provide an idea of the basic phases this system must implement. It also provided the main features in Arabic letters that this system will need to analyze compared to other languages. The studies in this paper discussed CNN, long short-term memory networks (LSTM), and Hybrid architectures. According to this paper, models with the highest accuracies were achieved by models that used CNN. (Albahli & Alrobah, 2022)

In (Pozanski & Wolf, 2016), a model that uses CNN in addition to binary attributes to recognize words was discussed. The CNN that was used was a VGG network that is made of small 3x3 convolution filters, they started with a 100x32 input image, and a network of 12 layers was used. Their system was used to read French, English, and Arabic words. They relied on the IFN/ENIT database for Arabic handwritten words. They achieved different results based on the different test scenarios' accuracies that ranged from 94.09%-99.29%. In the discussion, they claimed that the dominating success of Recurrent Neural Networks (RNNs) in the ICR field was due to its precedence a few years earlier than when CNN was beginning to be used. They expect CNNs to be implemented more in the future.

In (Musa & Elbashir, 2020), a model that uses CNN to recognize Arabic handwritten words Holistically was discussed. They especially focused on reading Arabic male names. Data was preprocessed by extracting the name from the scanned paper, surrounding white spaces were removed, downscaling the image to 28x56 pixels, and changing the background to black and the foreground to white. The CNN they used worked with batch normalization to speed up the training process. They also used down sampling to combat the problem where slight changes in a feature would cause drastic differences in its classification. This paper used the SUST-ARG names dataset. Using this model, they have achieved an accuracy of 99.14%. Unfortunately, 50% of the names appearing in the dataset were among a group of 20 names, making the data set

primarily comprised of 20 classes. They stated they believed if the data set were made up of more classes, the accuracy would decrease.

v) 2.2.5 Arabic Handwritten Postprocessing

It was found that some of the papers included postprocessing in their ICR system's stages while others did not. For instance, a study (Neto, Toselli, & Bezerra, 2020) used alternative spelling correction techniques for text post processing. They eliminate the linguistic dependence between the optical model and the decoding stage. They developed an encoder–decoder neural network architecture with a training methodology to achieve the goal of spelling correction.

It was found that post-processing methodologies can be categorized into four (Hussain, S. 2023): First **Spell-checking** including Error correction, text enhancement, and restoration. Second **Contextual Analysis** involves Analyzing the surrounding words based on the specific context. Third **Confidence Scoring** Assigns scores to words—a higher score means more accuracy. The fourth is **the Language Model** which uses a large corpus of text to guess the best word in context.

2.3 Resources Used

One of the Machine learning principles says: “garbage in, garbage out”, highlighting the importance of datasets in high-accuracy results. As mentioned in the literature review, it was found that the KHATT dataset will ensure a higher accuracy due to its diversity of subjects and writers. Therefore, the KHATT dataset will be used under its license terms.

Regarding code versioning, GitHub is the preferred platform. It supports the software development model, as mentioned above Incremental process. In terms of documentation, Google Drive has been used to store related documents and will continue to do so; it enables easy access and collaboration. As for sketching the diagrams in this report Draw.io and Visual Paradigm are used. For designing the Graphical User Interface Framer is used.

Chapter 3 System Analysis

Chapter 3 conducts a detailed System analysis. First, the product description, objectives, constraints, and requirements. Then the functional decomposition is presented. Finally, the systems models and the system architecture are included.

3.1 Product Description

In this section, the KhattClarifier web application description is presented in detail. that include objectives, and main features, functional, non-functional requirements, and both the user and system requirements.

i) 3.1.1 System Objectives

To display the extracted Arabic text from an Arabic Handwritten inputted image.

ii) 3.1.2 System Main Features

Users can recognize the Arabic text from images.

Users can make edits to the recognized text.

iii) 3.1.3 Operating Environments

1) Any android capable of running a browser application.
2) Any apple mobile product with IOS 16 or greater
3) Python to develop the ICR system.

iv) 3.1.4 Constraints

- Due to some browser limitations with IOS devices, devices running IOS 15 or lower cannot make use of this application.
- The app will need to use the Internet in order to transfer data from the website to the Python backend.

v) 3.1.5 Functional Requirements [1]

KhattClarifier is a web application that allows users to extract Arabic text from an image with handwritten Arabic in it. When the user first runs the web application, the application asks them for image input to be acquired. The image submission can be done by uploading the image. The user is then redirected to a page where they can crop which parts of the image they want to use exactly.

The AHR system first loads the recognition model, then performs image-preprocessing techniques on the image. The user can then obtain the recognized text as printed text. The application then enables the user to copy the recognized text to do with what they please.

vi) 3.2.6 Non-Functional Requirements

1) **Usability**: The average user from different age groups can use our system without any training time because the user shall be able to determine the functional features of the system with no assistance.
2) **Portability**: The web application runs on Android tablets and phones.
3) **Accuracy**: The ICR system has a recognition accuracy of 87%.
4) The user can extract the text within 5 - 10 seconds ± 2 seconds.
5) **Robustness**: The system should recover within 5-10 seconds.

vii) 3.2.7 System and User Requirements

Image handling

UR1 The user shall be able to submit an image from multiple resources.

[1] The functional requirements are written following the guidelines and writing style of (Bruegge & Dutoit, 2010).

SR1.1 The system should allow the user to choose the **input method** of their image.

SR1.1.1 The system should grant the user to browse their system for a photo.

SR1.1.2 The system should allow users to drag and drop the image.

SR1.2 The system shall perform **image-preprocessing** techniques to enhance the quality of the acquired image.

SR1.2.1 The system shall implement resizing of the image where the image will always have a width of 1280, but maintain it's aspect ratio.

SR1.2.2 The system should provide a bilateral filter to help remove noise and highlight the edges of the text.

SR1.2.3 The system should use adaptive mean thresholding to binarize the image (convert to black and white).

SR1.3 The system shall apply **image segmentation** techniques to the enhanced image.

SR1.3.1 The system shall use a large dilation filter, (20,100), to detect where the lines are.

SR1.3.2 The system shall use a small dilation filter, (25,25), to detect where the words are.

Arabic Handwritten Text Recognition

UR2 The user shall be able to obtain the recognized text of the acquired image.

SR2.1 The system shall recognize handwritten text and convert it to printed text.

SR2.2 The system will return the printed text back to the user.

Status notifications

UR3 The user shall be able to receive notifications regarding the success or failure of different operations they select to do.

SR3.1 The system shall be able to notify the users about the failure or the success.

SR3.2 When the notification is received, the system shall be able to provide a time stamp since it was first sent.

SR3.3 In case the handling options are available for success or failure, as in keeping going to the next step or go back to redo the current step.

3.2 Functional Decomposition

In this section, the use cases of the system are described in text and diagrams. The description includes actors, pre-conditions, the sequence or the flow of events, and alternative flows.

i) 3.2.1 Actors Description

Table 3.1: Actors Descriptions

Actor	**Description**
User	The primary Actor of the system, their actions determine the event's flow. - Open the application - Submit an Image by dragging and dropping. - Submit an Image by browsing for it and uploading it. - Obtain the recognized text document.
AHR system	The core system performs - the recognition based on the provided image and sends it back to the mobile application.

ii) 3.2.2 Use case descriptions

Use Case 1: Image submission

Table 3.2: Image acquiring use case

Actor	User
Description	The user can provide the image using dragging and dropping the image or upload it from their local device.
Pre-Conditions	- The user has opened KhattClarifier web application.
Events Flow	1. The user opens the website. 2. The user selects the browse option as the image input method. 3. The user selects the image they want. 4. The user crops the image. 5. The user clicks the submit button to submit the image.
Alternative Flows	If the user chooses to drag and drop the image as the image input method. 1. The user chooses the intended photo. 2. The user drags and drops the image into the website. 3. The user crops the image. 4. The user clicks the submit button to submit the image

Use case 2: Obtain the recognized text

Table 3.3: Obtain Recognized Text use case

Actor	User
Secondary Actors	Web application (KhattClarifier website) AHR system
Description	The user can obtain the recognized text by the AHR system using the web site.
Pre-Conditions	- The user has opened the KhattClarifier website. - The user has gone through the Image submission flow of events.
Events Flow	1. The AHR system pre-processes the image. 2. The AHR system processes the image. 3. The AHR System returns the recognized text to the website. 4. The KhattClarifier website displays the recognized text.
Alternative Flows	If the image quality caused a low accuracy and the user chooses to retake it: 1. The user visually validates the recognized text. 2. The user goes back to the main page. 3. Repeat the image submission flow events until successful.
	If the user wishes to cancel the recognition operation: 1. The user hits the back option in their browser. 2. The user goes back to the previous step.

	If the recognition operation encountered an error: 1. The system alerts the user of the error. 2. The use case ends with canceling the recognition.

iii) 3.2.3 Use Cases Diagram

The following figure provides the Use Case diagram, in which the possible interactions between the User and the environment actors are presented. It shows the various use cases:

- Image Submission
- Obtain Recognized Text

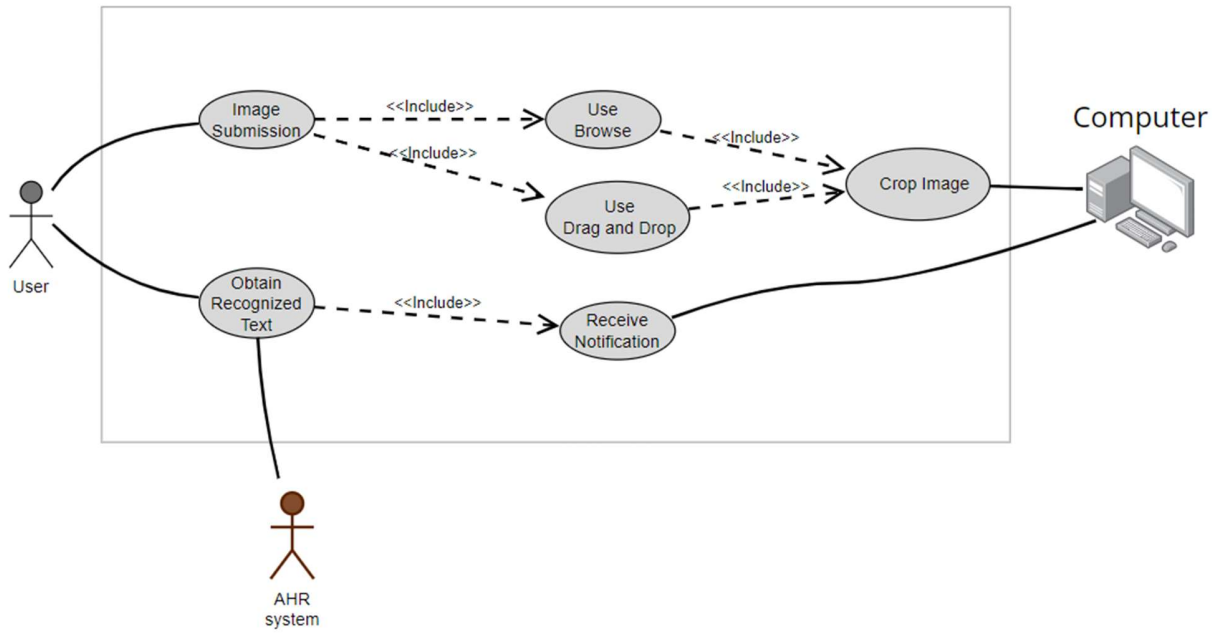

Figure 3.1: Use Case Diagram

3.3 System Models

In this section, the system models are shown. These include: Class diagram, Sequence diagram of each use case, state chart diagram, and activity diagram.

iv) 3.3.1 Class Diagram

The following diagram describes the structure of the project. It shows the system's classes, their attributes, operations, and the relationships among objects.

- **Image:** The image submitted by the user.
- **Preprocessing:** The class in charge of preprocessing the submitted image.
- **Segmentation:** The class in charge of segmenting the inputted image into smaller images.
- **ICR Model:** The AHR model used for this project
- **Recognized Text:** The final results of the recognized text generated by the ICR model.

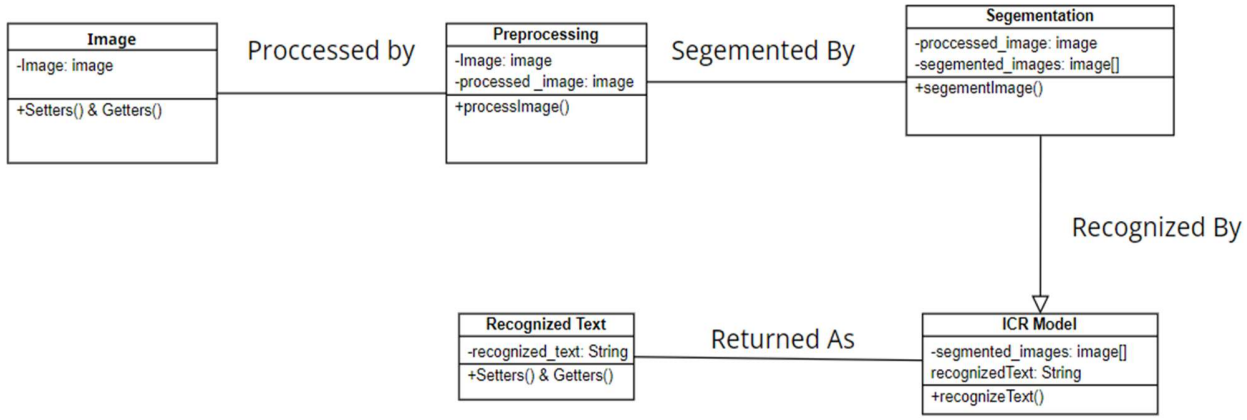

Figure 3.2: Class diagram

v) 3.3.2 Sequence Diagrams

In this section, the sequence diagrams for each use case of the system are presented. Where shows the interactions of the user, the exchanges between the classes.

Use case 1: Image submission: the following Sequence diagram shows how the user interacts with the system objects in the Use case of Image Submission.

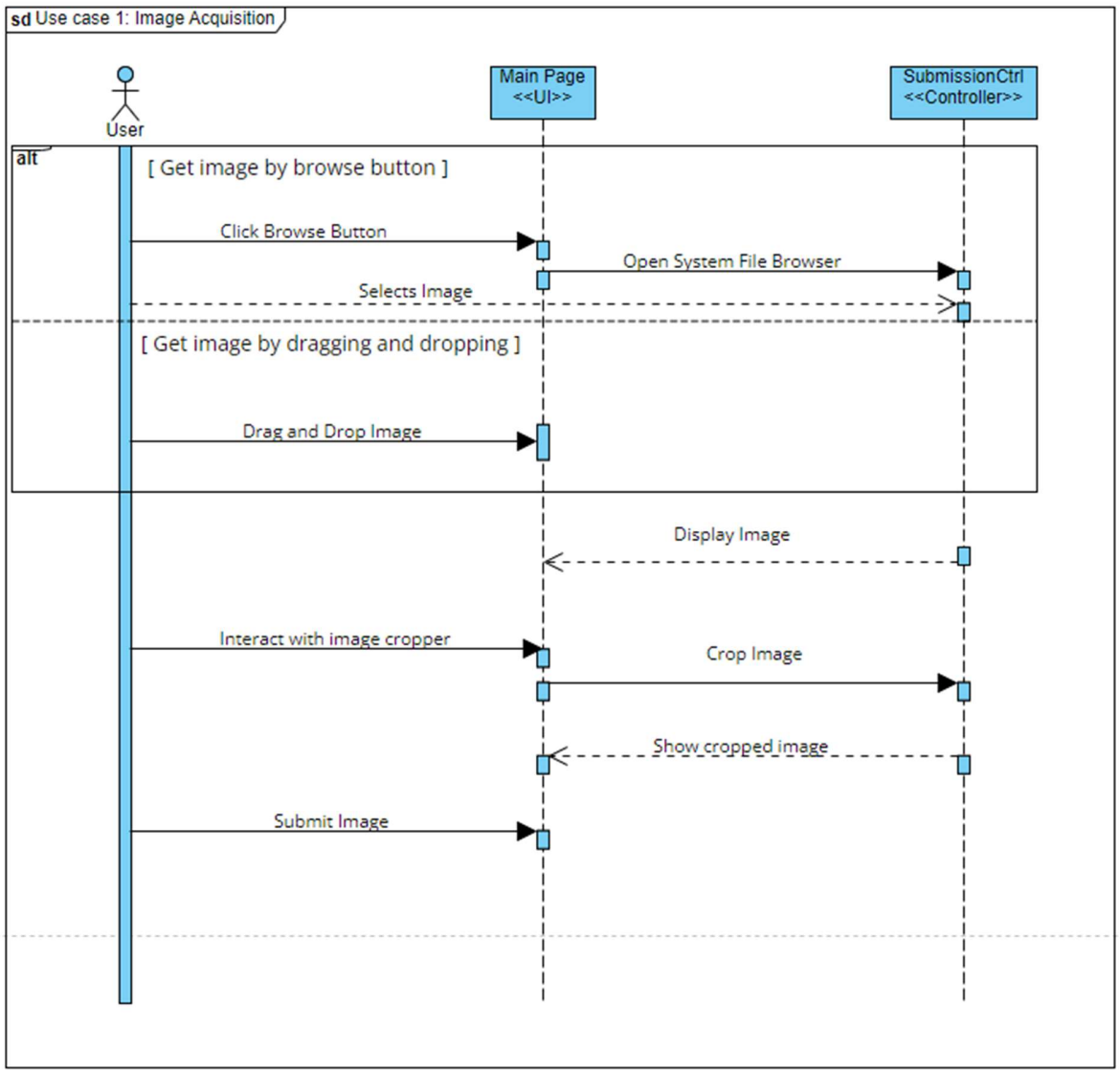

Figure 3.3: Sequence Diagram for Use Case 1

Use case 2: Obtain the recognized text: the following sequence diagram shows the User interacts with the system objects in the Use case of Obtaining the recognized text:

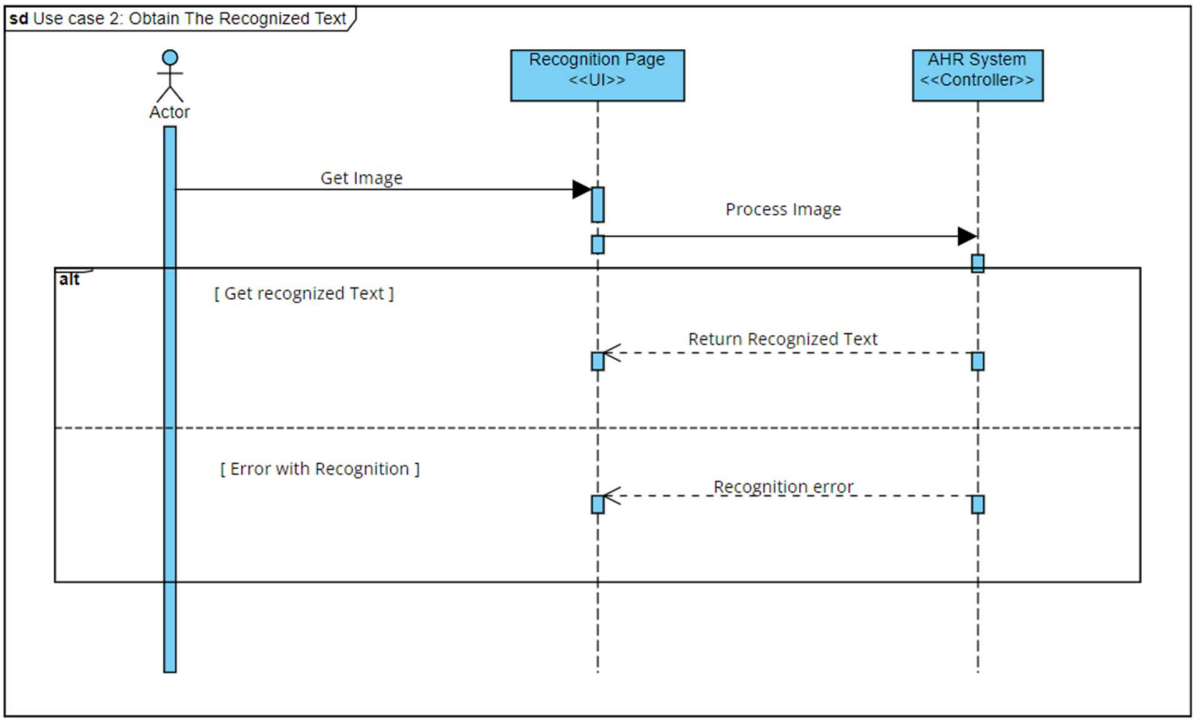

Figure 3.4: Sequence Diagram for Use Case 2

vi) 3.3.3 State Chart Diagram

The following diagram is used to model the different states of the system that changes by the events during its lifetime.

States

- Image acquiring,
- Obtaining Arabic Text Recognized Documents.

Transitions

- Open the app
- Image Accepted or Image Rejected.
- AHR system.

Termination states:

- Permission not granted
- The Recognized Text document is obtained.

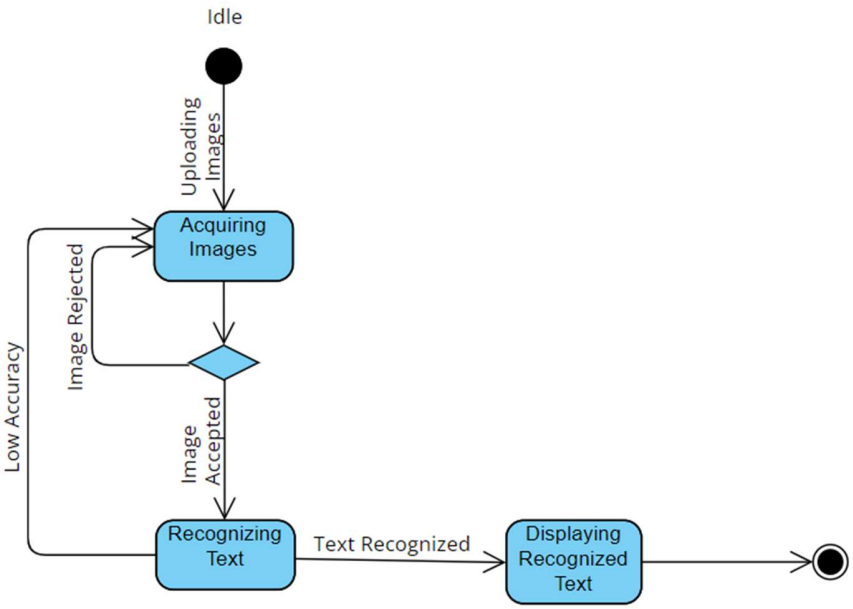

Figure 3.5: KhattClarifier State Chart Diagram

vii) 3.3.4 Activity Diagram

The following diagram is used to model the workflow of activities and actions:

- **Image Acquisition Activity:** Acquire the photo from drag and drop or browse.
- **Obtain Arabic Text Activity.**
- **Recognition Activity:** Do image pre-processing, Do image segmentation and Arabic text recognition.

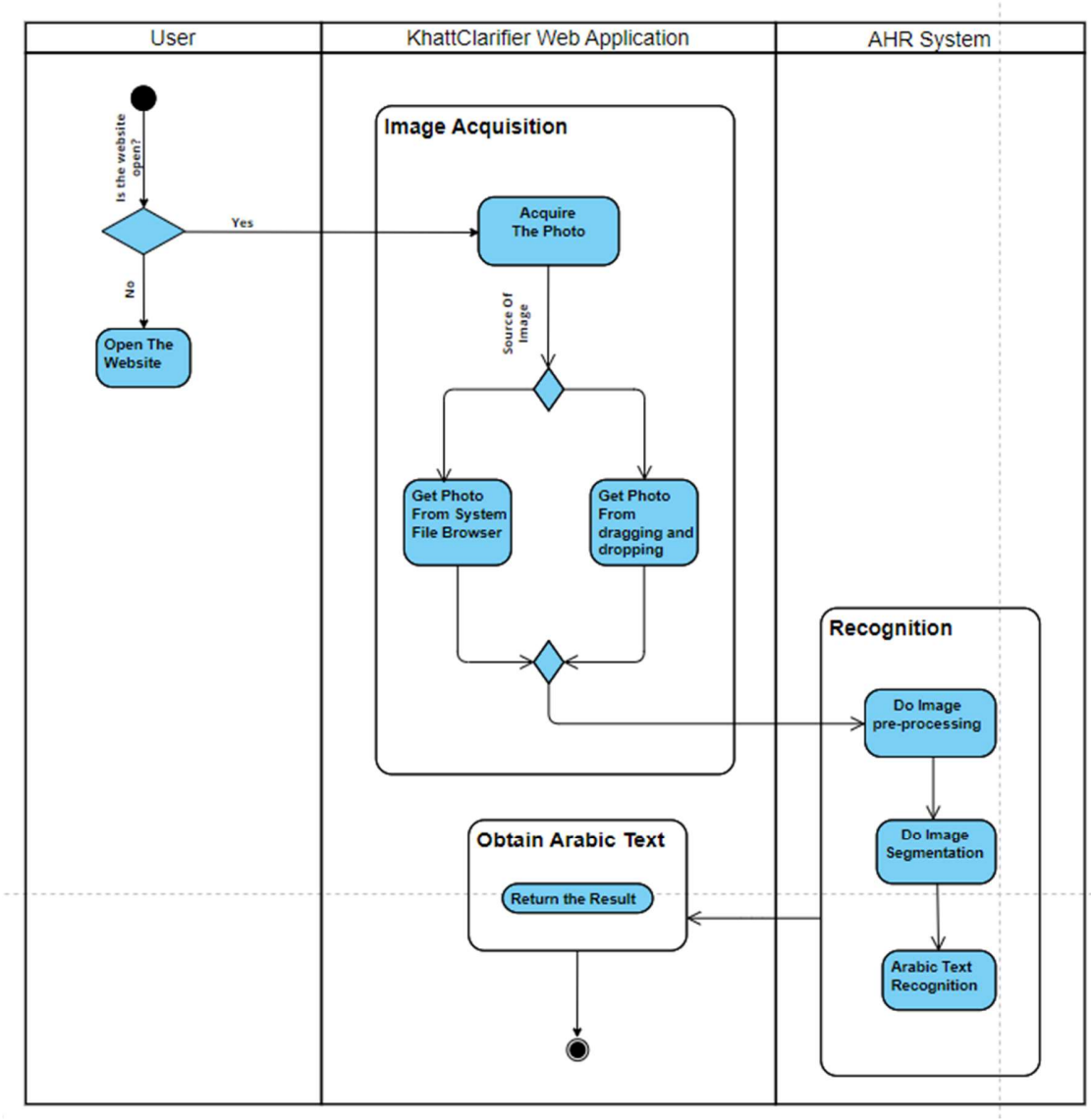

Figure 3.6: KhattClarifier Activity Diagram

3.4 System Architecture

In this section, the system architecture is presented. That includes: sub-systems, software architecture diagram and the deployment diagram.

viii) 3.4.1 Sub-Systems

1) **Image Acquisition Subsystem**: This subsystem includes functionalities to upload an image from the drag and drop or from the system's file browser and then upload that image to the Recognition subsystem.

   Components:

   - File Browser
   - Drag and Drop

2) **Character Recognition Subsystem**: The functionalities in this subsystem will conduct image preprocessing, image segmentation, and recognition of the Arabic text.

   Components:

   - Image Preprocessing system
   - Segmentation system
   - Recognition model (AHR)

ix) 3.4.1 Software Architecture

The following diagram shows the Architecture of the system. It includes different layers of User Interface (UI), Application, and AHR system layers.

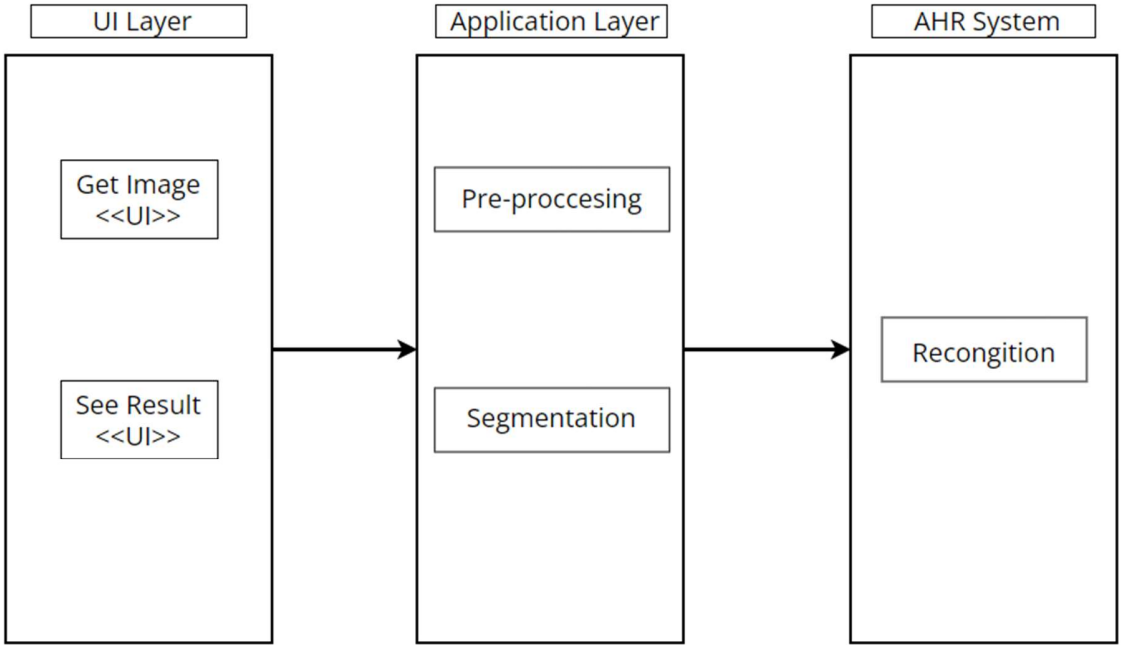

Figure 3.7: Software Architecture

x) 3.4.2 Deployment Diagram

The following diagram describes the deployment of the system, including different nodes.

Nodes:

- **Desktop** that had the KhattClarifier Web Application Artifact.
- **AHR System** that represents the ICR model Artifact.

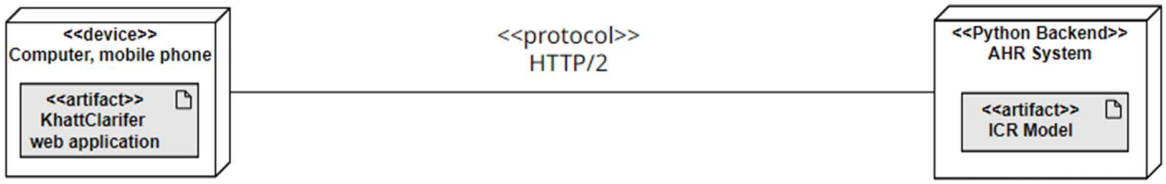

Figure 3.8: Deployment Diagram

Chapter 4 Methodology

This chapter will go through the techniques used for the AHR system and the website.

4.1 Website

4.1.1 Overall Information

The website was written in Python using the Flask framework, with some assistance for certain functionalities in JavaScript. As soon as the user opens the website, they are greeted with the main page that allows them to upload their image either by dragging and dropping the image or by selecting it from the system's file browser. The page then redirects to the user to an image cropper where the user can crop the image to how they see fit. Once the user submits their final image, the image is sent to the backend to be pre-processed and then recognized by the AHR system. The recognized text is sent back to a page in the website in a text area to be done with whatever the user desires. If the user does not like the results or would like to try another image, they can go back to the main page by hitting the back button on the browser.

4.1.2 Cropper

The cropper used is called Cropper.js (Cropper.js, 2015). It is an open-source JavaScript package that allows users to crop images on within their browser. The benefit of using this cropper, is that the user is able to crop their image many times without needing to refresh the browser, which makes it a lot more convenient for users.

4.2 Pre-processing Pipeline for Website

4.2.1 Overall Pre-processing used

The preprocessing pipeline was mainly done using the OpenCV library (OpenCV, 2024) and the TensorFlow library (TensorFlow, 2024). The image is first resized, then blurred using a filter, converted to gray scale and binarized, dilated twice for line and word segmentation, binarized once more and finally, converted to a TensorFlow data object for final resizing. Take for example this image:

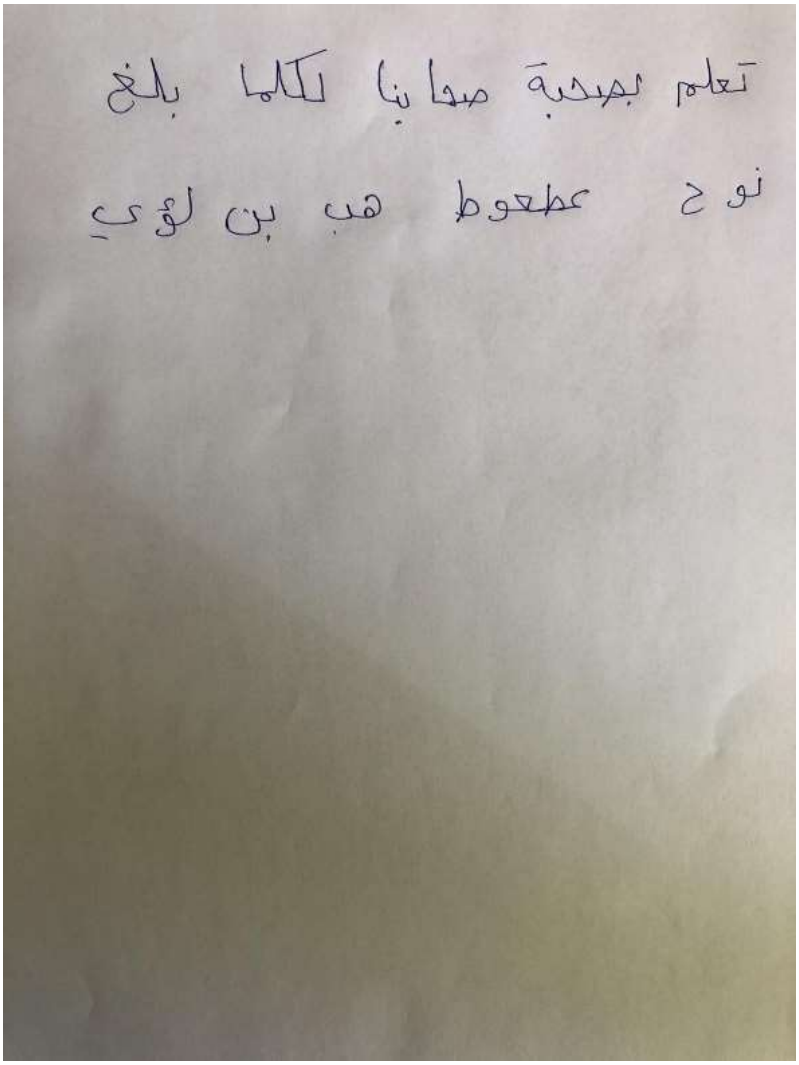

Figure 4.1: Example Image Before Pre-processing

4.2.2 Resizing the Image

The input image is resized to have a new width of 720, regardless of its old width. The new height will be calculated based on the original aspect ratio in order for the aspect ratio to be maintained. This is done so that the resized image doesn't become distorted due to stretching. To maintain the quality of the image, the inter cubic interpolation is used. After resizing, the example image will look like this:

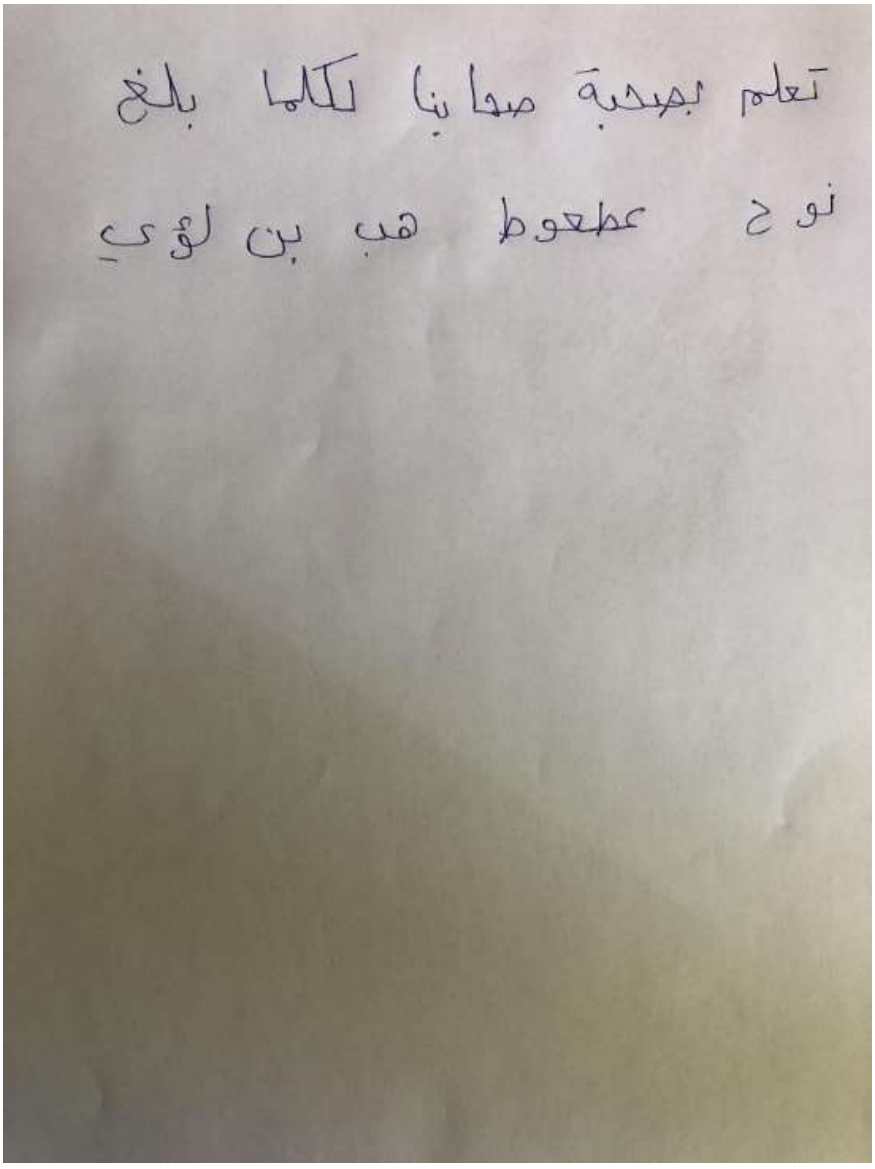

Figure 4.2: Example Image After Resizing

4.2.3 Blurring the Image

After resizing, the image is then blurred using a bilateral filter. Bilateral filtering is used as opposed to other filtering techniques because it specializes in reducing noise while retaining the edges in the image. The parameters used were: 101 for surrounding pixels, 30 for deviation of color space, 30 for deviation of coordinate space. After blurring, the example image will look like this:

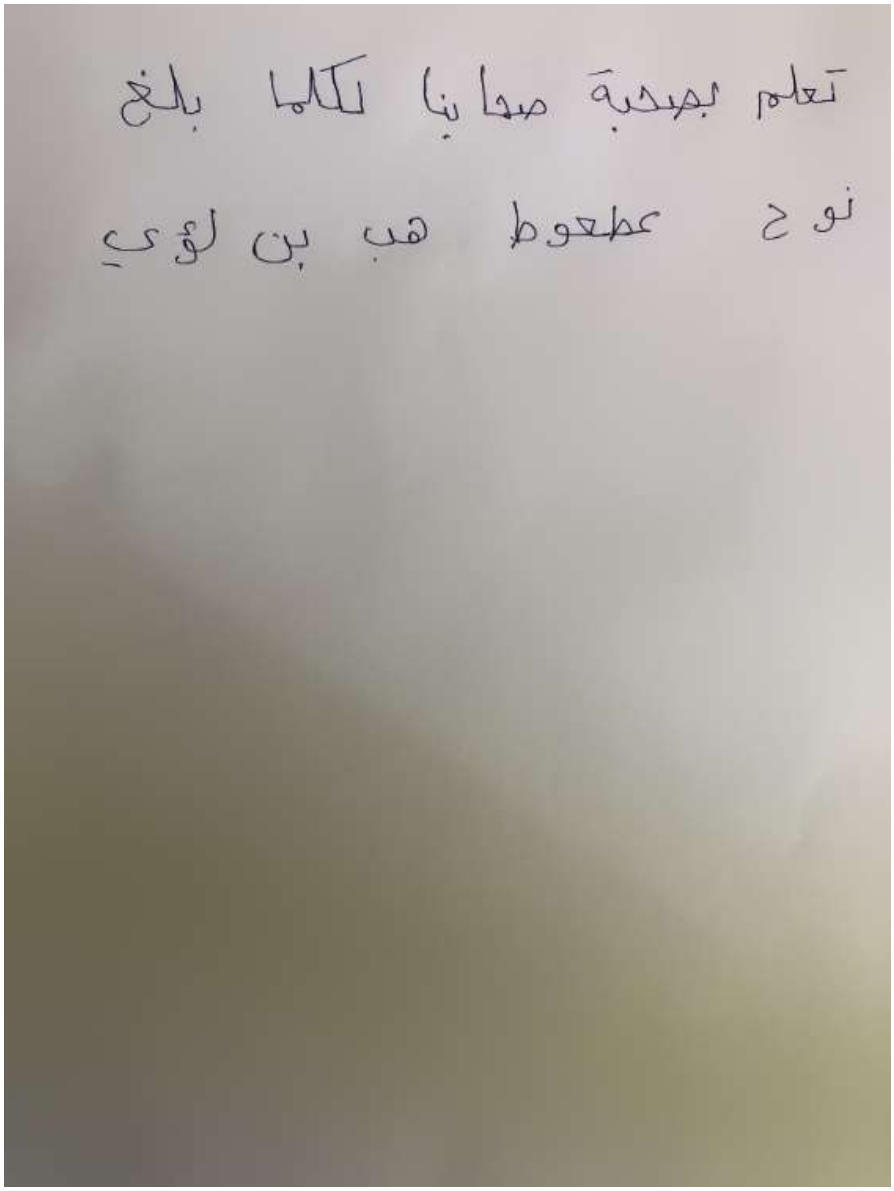

Figure 4.3: Example Image After Blurring

4.2.4 Thresholding the Image

After blurring, the image is converted to a binary image using an adaptive thresholding technique. But first, it is converted to a gray scaled image. The adaptive thresholding technique used was the adaptive mean thresholding. The parameters used for the adaptive thresholding were: Maximum value of 255, threshold binary inversed, a block size of 17 and a C value of 20. The purpose of this initial thresholding is to be able to dilate the image for segmentation. After grey scaling and thresholding, the example image will look like this:

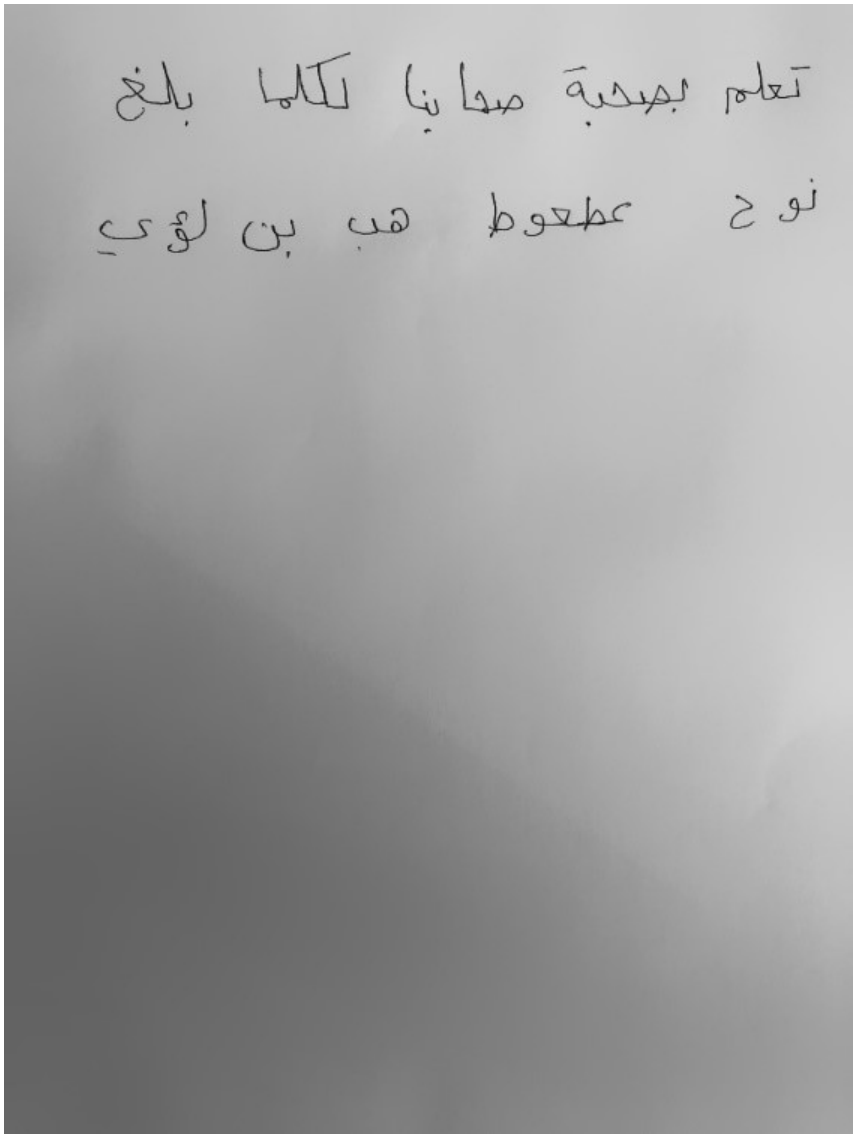

Figure 4.4: Example Image After Gray Scaling

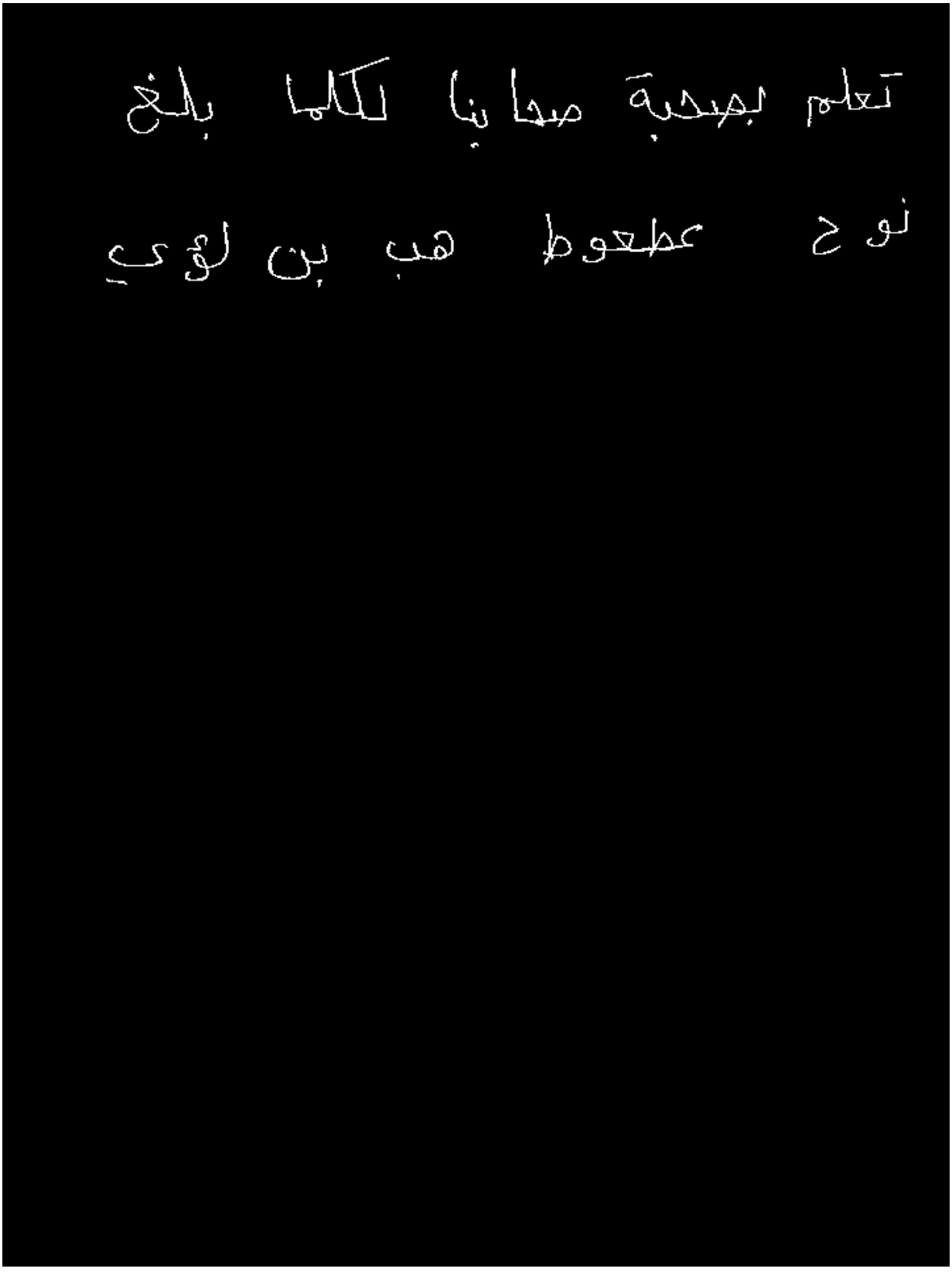

Figure 4.5: Example Image After Thresholding

4.2.5 Segmenting Lines and Words

Dilation was used to segment both the lines and the words. First, a large dilation is used to dilate the text largely horizontally and slightly vertically, in order to segment the lines from any non-line object in the image. The parameters used for line dilation were: kernel of 20 rows by 100 columns of ones and a single iteration. The dilation turns the example image to this:

Figure 4.6: Example Image Dilated for Lines

Using contouring, the lines detected in the image can be viewed here:

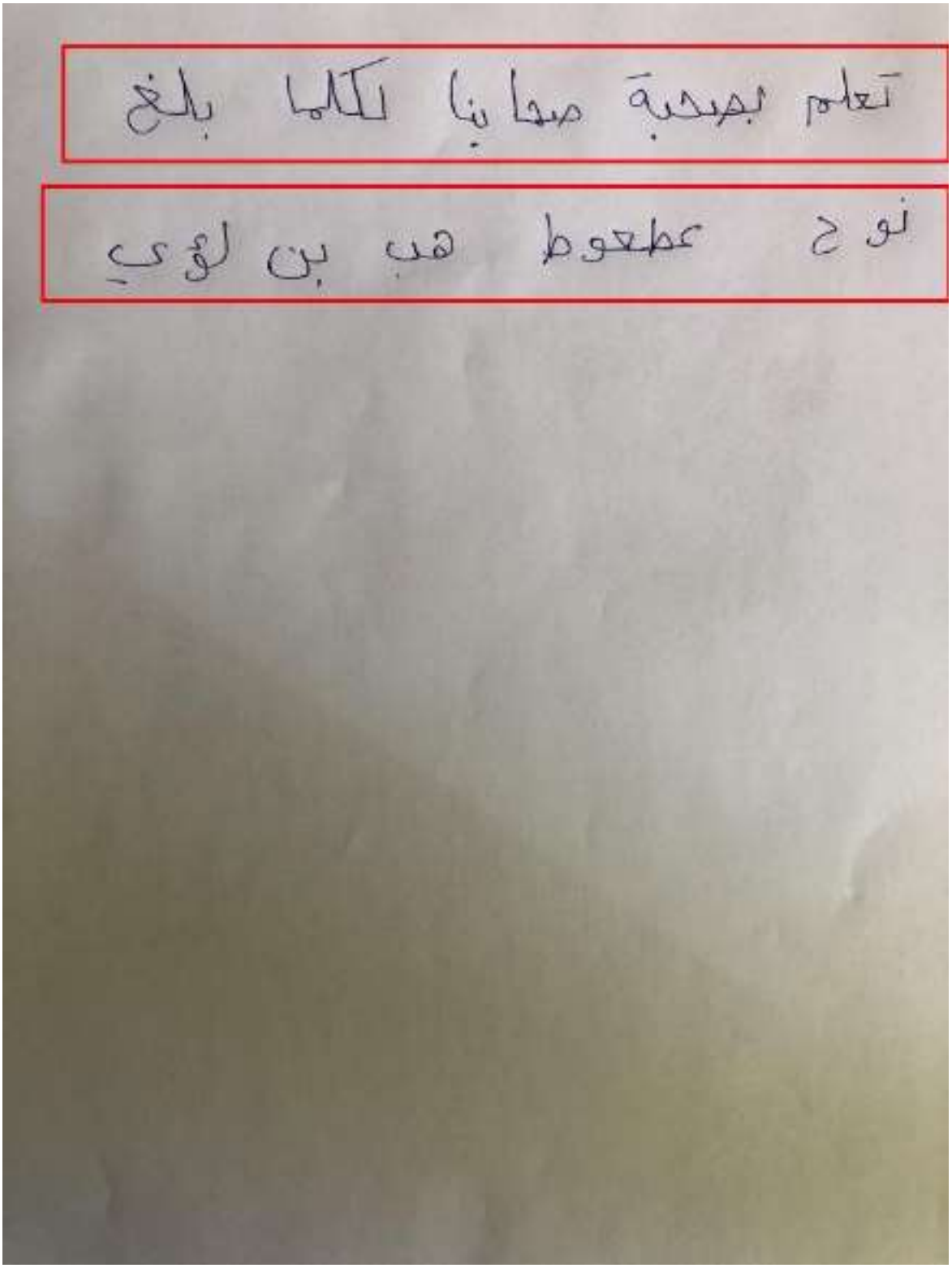

Figure 4.7: Example Image Lines Detected

The next dilation that is done is smaller dilation. This dilation is done to detect the words in the image. The small dilation uses the following parameters: a kernel of 25 rows and 25 columns of ones and a single iteration. The example image after the smaller dilation looks like this:

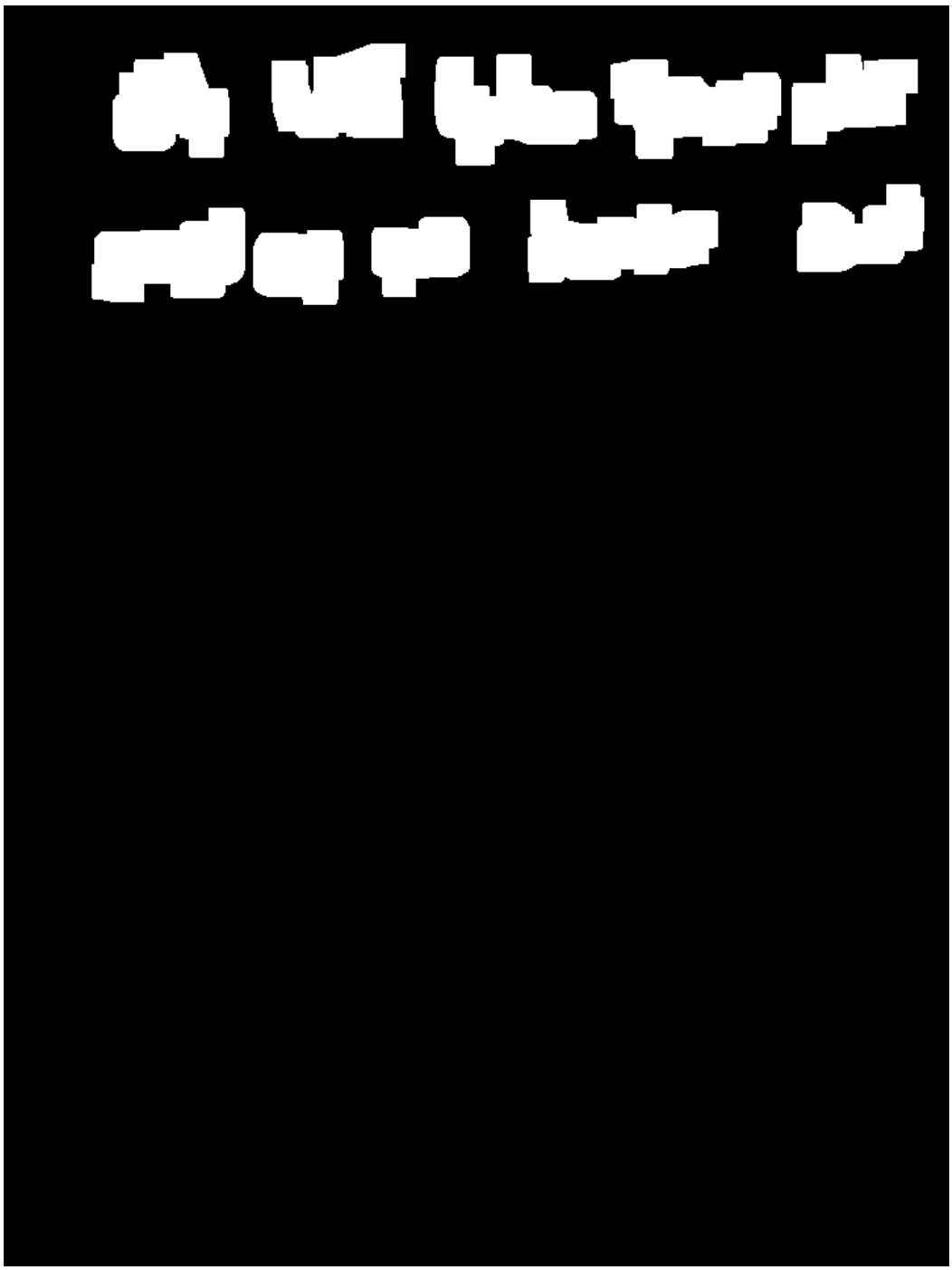

Figure 4.8: Example Image After Smaller Dilation

Using contouring, the words detected in the image can be viewed here:

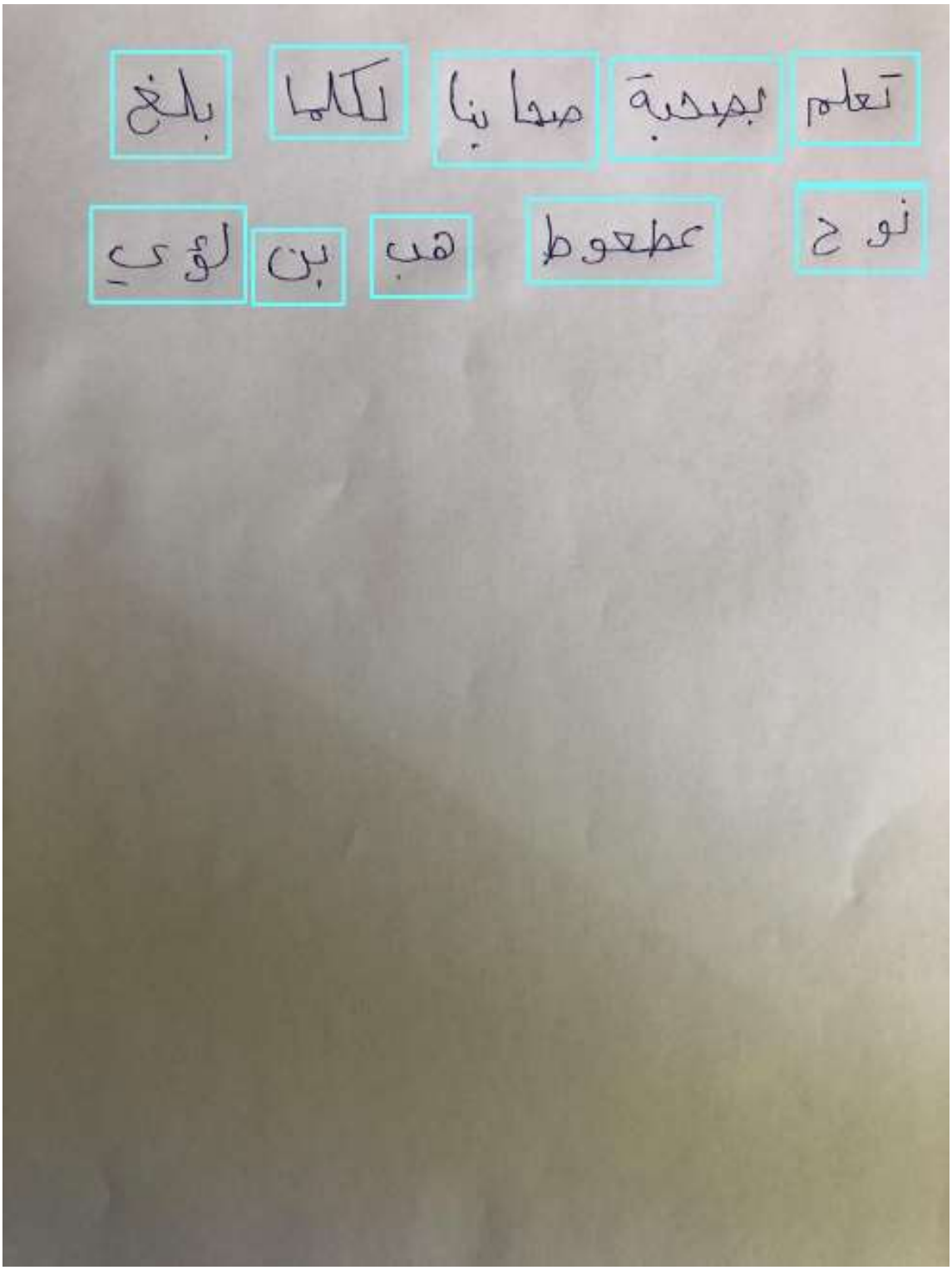

Figure 4.9: Words Detected from Example Image After Smaller Dilation

From the contours found based on the word and line dilations, each word can be cropped from the main image and stored based on which line they are located in.

4.2.6 Final Pre-processing Measures

After the words have been cropped and stored separately, they are binarized once more. The words then converted to a TensorFlow image data object, where they are resized to fit the model's input of 64x32 sized picture. The words are then thresholded manually for the final time where the threshold is 80.

4.3 Pre-processing the Training and Test Data

While the pre-processing of the dataset was very similar to the website, it was not exactly the same. This is due to the nature of the quality of the images found in the dataset as compared to the quality of images usually used by users (dataset doesn't have any noise, mostly binarized etc.). The dataset used rarely had any issues with quality as it was mostly pre-processed before being used. All that was needed was to binarize, resize and convert to a TensorFlow image data object as mentioned in 4.2.6.

As for labels, a different processing approach was needed so that it can work with this model. First, the labels need to be converted into a vector. The dataset's labels were searched through to find all the characters that are in the dataset. A total of 39 unique characters were found. Each character was given a unique integer identifier. Second, the labels were searched through to find any issues and fixed them, e.g. the spaces in the labels were removed. Third, the label with the longest characters was searched for and it was found the longest sequence with 7 characters long. For CTC to work a padding token needs to be introduced, that padding token was given an integer identifier of 99 to avoid conflicts with the other characters' identifiers.

4.4 Dataset Used

The dataset used was a modified version of the KHATT dataset. As mentioned previously, the KHATT dataset is a collection handwritten Arabic paragraphs. This modified version uses specific segmentation techniques designed to segment the paragraphs into words, sub words and punctuation as well as binarization to binarize all the images. From the original KHATT dataset, less than 100 images were used, but 5161 words were extracted from only these images. Utilizing the rest of the images was found to be difficult due to the images having issues with the structure of the lines, such as being highly skewed, lots of overlapping between words and lines. The dataset used can be expanded upon further once more robust segmentation techniques are discovered (such as utilizing deep learning).

4.5 Architecture Used

The architecture used for the model was a combination of a CNN, LSTM and NN. The first layer is the input layer that contains a shape of 64x32x1. The CNN's first layer is a convolutional 2D layer that has 32 kernels each with a size of (3,3) an activation function of "relu", a kernel initializer of "he_normal" and a padding of "same. The second layer is a max pooling 2D layer with a size of (2,2). The third layer is a batch normalization layer. The fourth layer is a convolutional 2D layer with 64 kernels with a size of (3,3), an activation function of "relu", kernel initializer of "he_normal" and a padding of "same". The fifth layer is a max pooling 2D layer with a size of (2,2). The sixth layer is a batch normalization layer. The seventh layer is a reshaping layer that reshapes the input into a shape of (512,256). The eighth layer is a dense layer with 16 neurons with an activation function of "relu". The ninth layer is a batch normalization layer. The tenth layer is a bidirectional LSTM layer with 512 LSTM units, 256 LSTM units going in each direction, with a dropout rate of 0.35 and return sequences. The eleventh and final layer is a dense layer of 41 neurons (number of total characters found in the dataset +2), with an activation function of "softmax". The loss function used was the CTC loss function. An initial learning rate of 0.01 was used with decay steps of 10,000 and a decay rate of 0.9 was used with the Adam optimizer. The model was trained for 450 epochs. Of course, several

variations of the model were used with differing hyper parameters before this optimal architecture was found.

Table 4.1: Model's architecture

Layer (type)	Parameters	Activation Function	Output Shape
image (input layer)	None	None	[(None,64,32,1)]
Conv1 (Conv2D)	32 kernels, (3,3) stride, "same" padding	ReLU	[(None,64,32,32)]
pool1 (MaxPooling2D)	(2,2) stride	None	[(None,32,16,32)]
batch_normalization_1 (BatchNormalization)	None	None	[(None,32,16,32)]
Conv2 (Conv2D)	64 kernels, (3,3) stride, "same" padding	ReLU	[(None,32,16,64)]
pool2 (MaxPooling2D)	(2,2) stride	None	[(None,16,8,64)]
batch_normalization_2 (BatchNormalization)	None	None	[(None,16,8,64)]
reshape (Reshape)	(16,512)	None	[(None,32,256)]
dense1 (Dense)	16 neurons	ReLU	[(None,32,16)]
batch_normalization_3 (BatchNormalization)	None	None	[(None,32,16)]
bidirectional_1 (Bidirectional LSTM)	1 LSTM layer each way, 256 cells, return_sequences=True, 0.35 dropout	None	[(None,32,512)]
dense2 (Dense)	41 neurons	Softmax	[(None,32,42)]

4.6 Training

As mentioned previously, the model was trained using 450 epochs. It was trained using an NVIDIA RTX 4050 GPU and took 4 and a half hours to train. The dataset was augmented to include an additional 5161 images of the images rotated counter-clockwise 15 degrees and another 5161 images were augmented to be rotated clockwise 15 degrees. That brings the total of images used for training is 15483. The model was trained in google colab using a local host runtime, which requires the installation of google docker and WSL.

Chapter 5 Results

5.1 Calculation of Error Rate

The error rate was calculated by subtracting the character error from 1. The character error rate is calculated by taking the any extra character insertions, deletions or substitutions in a prediction and dividing it by the total character count. In a formula it can be written like this:

$$Character\ Error\ Rate\ (CER) = \frac{insertions + substitutions + deletions}{total\ character\ count}$$

5.2 Accuracies of the different models trained:

The hyper parameters and the layers were modified on the model to see how the model's accuracy can be improved. The same base model was used. The modifications and accuracies can be found in the table below:

Table 5.1: Results of Training

Modification Number	Modifications made	Accuracy
1	learning rate of 0.0001	80%
2	learning rate of 0.001	85%
3	Modification 1 + BiLSTM of 128 cells per LSTM	83%
4	Modification 1 + remove of Conv2 and pool2 layers	83%
5	Modification 1+4+3	85%

6	learning rate of 0.00001	74%
7	Modification 1 + decay steps 1000	83%
8	decay steps 1000	82%
9	removal of conv2 and pool2 layers	85%
10	Modification 1+ removal of cov2 and pool2 layers BiLSTM of 128 cells per LSTM without data augmentation	68%
11	Modification 1+ removal of cov2 and pool2 layers BiLSTM of 128 cells per LSTM without data augmentation 1000 epochs	73%
No Modifications	Same model in 4.6	87%

Chapter 6 Discussion and Conclusion

6.1 Summary

This project was able to successfully create a user-friendly website to translate their handwritten Arabic images into printed text. Users ranging from teachers to students are able to use this project The AHR system of this project has been able to successfully get a character recognition accuracy of over 80%. The AHR system was able to be created using an architecture of a deep CNN, BiLSTM with CTC loss. Future improvements will be able to make KhattClarifier even better.

6.2 Future Improvements

Although the results are positive, the entire KHATT dataset was not used. The KHATT dataset's images have some issues with skewing, as a great number of these images have lines that are not entirely nor somewhat rotated around a point, they tend to round off as they go on. These lines need to be de-skewed manually or using advanced de-skewing techniques. There are also issues with segmentation of lines and words in the images as a great number of images in the KHATT dataset have words that overlap into each other as mentioned in 1.1. Words will sometimes partially overlap with sentences below it as well.

As for the model, only a few modifications and tests can be made on the model as training time is very computationally and time exhausting on the current hardware available. More variations of the hyperparameters as well as different values should be tested with better hardware to lower the training time.

References

Albahli, S., & Alrobah, N. (2022). Arabic Handwritten Recognition Using Deep Learning: A Survey. *Arabian Journal for Science and Engineering*.

Al-Ohali, Y., Cheriet, M., & Suen, C. (2003). *Databases for recognition of handwritten Arabic cheques.* Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S003132030200064X

Boraik, O., Manjunath, R., & Saif, M. (2022). *Characters Segmentation from Arabic Handwritten Document Images: Hybrid Approach.* Retrieved from International Journal of Advanced Computer Science and Applications: https://www.researchgate.net/publication/360343135_Characters_Segmentation_from_Arabic_Handwritten_Document_Images_Hybrid_Approach

Britannica. (2023). Retrieved from https://www.britannica.com/topic/Arabic-language

Bruegge, B., & Dutoit, A. H. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java™ (3rd ed.).* Technical University of Munich Department of Computer Science Munich, Germany.

Cropper.js. (2015). *Github*. Retrieved from https://fengyuanchen.github.io/cropperjs/

Elb Abed, H., & Margner, V. (2007). *The IFN/ENIT-database Its Applications.* Retrieved from https://link.springer.com/chapter/10.1007/978-1-4471-4072-6_8

Faizullah, S., Ayub, M. S., Hussain, S., & Khan, M. A. (2023). *A Survey of OCR in Arabic Language: Applications, Techniques, and Challenges.* Retrieved from https://www.researchgate.net/publication/369775787_A_Survey_of_OCR_in_Arabic_Language_Applications_Techniques_and_Challenges

IBM. (2023). *What is computer vision?* Retrieved from https://www.ibm.com/topics/computer-vision

Klippa. (2022, November 30). *What is ICR, and why do you need it?* Retrieved from https://www.klippa.com/en/blog/information/what-is-icr/

Mahmoud, Ahmad, Al-Khatib, & Alshayeb. (2014). *KHATT: An open Arabic offline handwritten text database.* Retrieved from Pattern Recognition: https://www.sciencedirect.com/science/article/abs/pii/S0031320313003300

McCarthy, J. (2007). *What is Artificial Intelligence.* Retrieved from https://www-formal.stanford.edu/jmc/whatisai.pdf

Mousa, M. A., Sayed, M. S., & Abdalla, M. A. (2017). *Arabic Character Segmentation Using Projection Based Approach with Profile's Amplitude Filter.* Retrieved from https://www.researchgate.net/publication/318205989_Arabic_Character_Segmentation_Using_Projection_Based_Approach_with_Profile%27s_Amplitude_Filter

Musa, M., & Elbashir, M. (2020). *A Deep Learning Approach for Handwritten Arabic Names Recognition.* Retrieved from International Journal of Advanced Computer Science and Applications 11: https://www.researchgate.net/publication/339025964_A_Deep_Learning_Approach_for_Handwritten_Arabic_Names_Recognition

Neto, A., Toselli, A., & Bezerra, B. (2020). *Towards the Natural Language Processing as Spelling Correction for Offline Handwritten Text Recognition Systems.* Retrieved from Applied Sciences: https://www.researchgate.net/publication/345166167_Towards_the_Natural_Language_Processing_as_Spelling_Correction_for_Offline_Handwritten_Text_Recognition_Systems

OpenCV. (2024). Retrieved from https://opencv.org/

Pozanski, A., & Wolf, L. (2016). *CNN-N-Gram for HandwritingWord Recognition.* Retrieved from IEEE Conference on Computer Vision and Pattern Recognition: https://ieeexplore.ieee.org/document/7780622

Qaroush, A., Abdalkarim, A., Abualsound, H., Mohammad, K., Jaber, B., & Hasheesh, A. (2022). *Learning-free, divide and conquer text-line extraction algorithm for printed Arabic text with diacritics.* Retrieved from https://www.researchgate.net/publication/360522477_Learning-free_divide_and_conquer_text-line_extraction_algorithm_for_printed_Arabic_text_with_diacritics

Sharma, N. (2023). Introduction to Artifical Intelligence. CS department.

TensorFlow. (2024). Retrieved from https://www.tensorflow.org/

Report.docx

ORIGINALITY REPORT

13%	10%	9%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	github.com Internet Source	1%
2	www.mdpi.com Internet Source	1%
3	strongfemaleprotagonist.com Internet Source	1%
4	www.csc.villanova.edu Internet Source	1%
5	arxiv.org Internet Source	1%
6	dokumen.pub Internet Source	<1%
7	inst.eecs.berkeley.edu Internet Source	<1%
8	ksascholar.dri.sa Internet Source	<1%
9	gifu-u.repo.nii.ac.jp Internet Source	<1%

10	manuscript.springernature.com Internet Source	<1 %
11	dspace.rri.res.in:8080 Internet Source	<1 %
12	jearq.riau.ac.ir Internet Source	<1 %
13	Hum Nath Parajuli, Galymzhan Bakhtiyarov, Bikash Nakarmi, Ikechi Augustine Ukaegbu. "Chapter 15 Interference Mitigation in Multi-radar Environment Using LSTM-Based Recurrent Neural Network", Springer Science and Business Media LLC, 2024 Publication	<1 %
14	Alanazi, Mohammad S.. "The Use of Computer-Assisted Translation Tools for Arabic Translation: User Evaluation, Issues, and Improvements.", Kent State University, 2020 Publication	<1 %
15	Lecture Notes in Computer Science, 2006. Publication	<1 %
16	hdl.handle.net Internet Source	<1 %
17	repository.uinsi.ac.id Internet Source	<1 %
	www.sciedupress.com	

18	Internet Source	<1 %
19	Aziz Qaroush, Bassam Jaber, Khader Mohammad, Mahdi Washaha, Eman Maali, Nibal Nayef. "An efficient, font independent word and character segmentation algorithm for printed Arabic text", Journal of King Saud University - Computer and Information Sciences, 2019 Publication	<1 %
20	www.nature.com Internet Source	<1 %
21	scholar.uwindsor.ca Internet Source	<1 %
22	Guide to OCR for Arabic Scripts, 2012. Publication	<1 %
23	www.politesi.polimi.it Internet Source	<1 %
24	www.theseus.fi Internet Source	<1 %
25	Naseem Alrobah, Saleh Albahli. "Arabic Handwritten Recognition Using Deep Learning: A Survey", Arabian Journal for Science and Engineering, 2022 Publication	<1 %
	etd.uwc.ac.za	

26	Internet Source	<1 %
27	Omar Ali Boraik, M. Ravikumar, Mufeed Ahmed Naji Saif. "Characters Segmentation from Arabic Handwritten Document Images: Hybrid Approach", International Journal of Advanced Computer Science and Applications, 2022 Publication	<1 %
28	www.eecs.northwestern.edu Internet Source	<1 %
29	www.oregon.gov Internet Source	<1 %
30	www.researchgate.net Internet Source	<1 %
31	Handbook of Document Image Processing and Recognition, 2014. Publication	<1 %
32	Nguyen Viet Hung, Tran Thanh Lam, Tran Thanh Binh, Alan Marshal, Truong Thu Huong. "Efficient Deep Learning-Based Viewport Estimation for 360-Degree Video Streaming", Advances in Science, Technology and Engineering Systems Journal, 2024 Publication	<1 %
33	mountainscholar.org Internet Source	<1 %

34	Arthur Flor de Sousa Neto, Byron Leite Dantas Bezerra, Alejandro Héctor Toselli. "Towards the Natural Language Processing as Spelling Correction for Offline Handwritten Text Recognition Systems", Applied Sciences, 2020 Publication	<1 %
35	Richard Pinčák, Kabin Kanjamapornkul, Erik Bartoš. "Forecasting Laurent Polynomial in the Chern–Simons Current of V3 Loop Time Series", Annalen der Physik, 2019 Publication	<1 %
36	scholarworks.uark.edu Internet Source	<1 %
37	de Carvalho, José Pedro Ferreira Pinheiro. "Deep Reinforcement Learning Methods for Cooperative Robotic Navigation", Universidade do Porto (Portugal), 2024 Publication	<1 %
38	vdoc.pub Internet Source	<1 %
39	eprints.kfupm.edu.sa Internet Source	<1 %
40	www.fedoa.unina.it Internet Source	<1 %
41	dspace.lboro.ac.uk	

	Internet Source	<1%
42	technodocbox.com Internet Source	<1%
43	www.mandiant.com Internet Source	<1%
44	www.sciencepg.net Internet Source	<1%
45	Janke, Jonathan Gabriel Martin. "Analysis of the Proficiency of Fully-Connected Neural Networks in the Process of Classifying Digital Images : Benchmark of Different Classification Algorithms on High-Level Image Features from Convolutional Layers", Universidade NOVA de Lisboa (Portugal), 2024 Publication	<1%
46	assets.researchsquare.com Internet Source	<1%
47	kyutech.repo.nii.ac.jp Internet Source	<1%
48	pubmed.ncbi.nlm.nih.gov Internet Source	<1%
49	www.freepatentsonline.com Internet Source	<1%

50	www.iprjb.org Internet Source	<1 %
51	Seung Jin, Jung Cho, Jae Jeon. "FPGA based Passive Auto Focus System using Adaptive Thresholding", 2006 SICE-ICASE International Joint Conference, 2006 Publication	<1 %
52	Thakur, Abhishek K.. "Atomistic Simulations of Interstellar Carbon Nanostructures and Multiscale Modeling of Spinodal Decomposition in Metal Alloys", The University of Arizona, 2023 Publication	<1 %
53	cdigital.uv.mx Internet Source	<1 %
54	coou.edu.ng Internet Source	<1 %
55	hal.archives-ouvertes.fr Internet Source	<1 %
56	ouci.dntb.gov.ua Internet Source	<1 %
57	www.britannica.com Internet Source	<1 %
58	"Digital Technologies and Applications", Springer Science and Business Media LLC, 2023	<1 %

	Publication	
59	1library.net Internet Source	<1 %
60	Freitas, Diogo Domingos Lopes de. "Multi-temporal Optimal Power Flow Including Storage", Universidade do Porto (Portugal), 2024 Publication	<1 %
61	Phan, Thaibao Peter. "Snapshot Hyperspectral Polarization Imaging", Stanford University, 2023 Publication	<1 %
62	Udumulla, Niranga Mahesh. "Predicting Tropical Cyclone Intensity from Geosynchronous Satellite Images Using Deep Neural Networks", Florida Atlantic University, 2021 Publication	<1 %
63	espace.etsmtl.ca Internet Source	<1 %
64	etheses.bham.ac.uk Internet Source	<1 %
65	prism.ucalgary.ca Internet Source	<1 %
66	www.coursehero.com Internet Source	<1 %

67	www.psychosocial.com Internet Source	<1 %
68	Yogendra Narayan Pandey, Ayush Rastogi, Sribharath Kainkaryam, Srimoyee Bhattacharya, Luigi Saputelli. "Machine Learning in the Oil and Gas Industry", Springer Science and Business Media LLC, 2020 Publication	<1 %
69	"Artificial Neural Networks and Machine Learning – ICANN 2023", Springer Science and Business Media LLC, 2023 Publication	<1 %
70	"Deep Learning and Convolutional Neural Networks for Medical Image Computing", Springer Science and Business Media LLC, 2017 Publication	<1 %
71	"Proceedings of International Conference on Artificial Intelligence and Applications", Springer Science and Business Media LLC, 2021 Publication	<1 %
72	Alhag Alsayed, Chunlin Li, Ahmed Fat'hAlalim, Mohammed Hafiz, Jihad Mohamed, Zainab Obied, Mohammed Abdalsalam. "The Impact of Various Factors on the Convolutional	<1 %

Neural Networks Model on Arabic Handwritten Character Recognition", International Journal of Advanced Computer Science and Applications, 2024
Publication

73	M. Mohandes, M. Deriche. "Image based arabic sign language recognition", Proceedings of the Eighth International Symposium on Signal Processing and Its Applications, 2005., 2005 Publication	<1 %
74	Tasie-Amadi, Kinikanwo Ayodele. "Model-Checking of UML Models Using CSP", The University of Manchester (United Kingdom), 2023 Publication	<1 %
75	X. Li, R. Ramachandran, M. He, Sunil Movva, J. Rushing, S. Graves, W. Lyatsky, Arjun Tan, G. Germany. "Comparing different thresholding algorithms for segmenting auroras", International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004., 2004 Publication	<1 %

Exclude quotes	Off	Exclude matches	< 4 words
Exclude bibliography	Off		