People's Democratic Republic of Algeria
Ministry of Higher Education and
Scientific Research
Larbi Tebessi University - Tebessa
Faculty of Exact Sciences and Natural and Life Sciences
Department of Mathematics and Computer Science
End of studies dissertation
in view of obtaining the Master's degree
**Field:** Mathematics and Computer Science
**Sector:** Computer Science
**Option:** Systems and Multimedia
Thesis

# Analytic study of the preprocessing methods impact on historical document analysis and classification

Presented by:

**Souahi Houssem**

The Jury Members:

| | | | |
|---|---|---|---|
| Menassel Rafik | MCA | University Larbi Tebessi – Tebessa | President |
| Bourougaa salima | MCA | University Larbi Tebessi – Tebessa | Examiner |
| Akram Bennour | MCA | University Larbi Tebessi – Tebessa | Supervisor |

Academic Year: **2022/ 2023**

# Dedication

First and foremost, we thank **ALLAH** the Almighty for giving us strength and patience to overcome the difficulties and to finish what we have started in the study trip. We ask **ALLAH** to help us in the future.

I would like to express my deepest thanks to my supervisor
Dr Akrem Bennour, for the supervision, the trust and for the support he gave me during all this work.

His scientific skills have always been a source of enrichment which have enabled me to carry out this work. Thanks for all his advice and patience.

A big thank to all the members of the Jury who do me the honor of accepting to read and judge my work.

I thank all the teachers and students of the computer science department of the University of Tebessa.

I would also like to present my thanks to my mother and brothers for all their sacrifices in favor of my education, my friends for their support which allowed me to work in the best conditions.

# *Abstract*

While historians face various challenges in the process of dating the texts of historical manuscripts, computer scientists face multiple difficulties in automating these texts. To address this problem, deep learning techniques that have proven effectiveness in other fields have been used. Of this study presents the various pre-processing methods used in character recognition systems, which cater to a wide range of image types, as these images include simple handwritten forms and documents with colorful and complex backgrounds and varying intensity. Basic pre-processing techniques are comprehensively discussed, including aberration detection and correction, contrast stretching for image optimization, binary encoding, noise removal methods, normalization, segmentation, and morphological processing techniques.

**Keywords:** dating, deep learning, Binarization, Character Recognition, Noise Removal, Normalization, Preprocessing Techniques, Segmentation.

## الخلاصة

بينما يواجه علماء التاريخ تحديات مختلفة في عملية تأريخ نصوص المخطوطات التاريخية ، يواجه علماء الكمبيوتر عقبات متعددة في أتمة هذه النصوص. لمعالجة هذه المشكلة ، تم استخدام تقنيات التعلم العميق التي أثبتت فعاليتها في مجالات أخرى. حيث تقدم هذه الدراسة طرق المعالجة المسبقة المتنوعة المستخدمة في أنظمة التعرف على الأحرف ، والتي تلبي مجموعة واسعة من أنواع الصور، كما تشتمل هذه الصور على أشكال بسيطة مكتوبة بخط اليد ومستندات ذات خلفيات ملونة ومعقدة وكثافة متفاوتة. فتناقش بشكل شامل تقنيات المعالجة المسبقة الأساسية ، بما في ذلك الكشف عن الانحراف وتصحيحه ، وتمديد التباين لتحسين الصورة ، والترميز الثنائي ، وطرق إزالة الضوضاء ، والتطبيع ، والتجزئة ، وتقنيات المعالجة الصرفية.

**الكلمات الرئيسية**: المواعدة ، التعلم العميق ، الثنائية ، التعرف على الحروف ، إزالة الضوضاء ، التطبيع ، تقنيات المعالجة المسبقة ، التجزئة.

## Table of Contents

## List of Figures

# General Introduction

# General Introduction

Historical documents are of great importance in informatics. There are also many libraries around the world that digitize their documents to make them available to people and to preserve their funds.

It also requires access to its contents by applying document image processing algorithms, since its raw [1] form is not machine-readable. Analysis in document planning is the process of identifying the foreground and classifying the different areas in the image, it's an important pre-processing step for many document images.

The use of deep learning algorithms has also provided state of the art to many areas of computer vision, because of its learned advantages, it is considered to be one of the most powerful analyses in deep learning.

We present a method for the schematic analysis of historical handwritten documents using a Siamese grid [2], it consists of two identical convolutional neurons.

If it is an example then you should write: CNN networks, for instance insert a pair. It is like inserting a pair of images or points, extracting their features and arranging their similarities. Looking at the historical pages, they are divided into the same size and also we train the Siamese as a network model. Using it, we build a space matrix between the patches of each test page. Then we use the distance matrix to divide the spots into three categories: main text, side text, and background [3].

Where we are evaluating work on challenging the historical manuscripts dataset, filtering in the main text partition as well as updating the partition and reducing the power function. They advance higher Probabilistic approximation and mapping of component pairs for the same nomenclature.

Where each pixel is represented as vector features based on the color of the image. Then Gaussian Mix Modeling (GMM), Multilayer Perception (MLP) and SVM to classify pixels into ornamentation, Background pixels, sides and text. It turns out that SVM and MLP are the ones that generally outperform GMM in hashing [4].

Instead, all of the work excels, representing every pixel with more features. They use more Texture and color features such as Smoothness, Laplacian, Dominant Gabor Orientation Graph, and local binary patterns, In addition, they select the many features, and use the algorithm to remove the irrelevant features [5].

# Chapter 01:

# Theorical

# Conception

# Introduction

The Historical Document Pre-processing System is essentially an artificial simulation of human reading, designed to process historical documents that include handwritten or machine-written text, graphics, videos, and other elements, with the aim of making them machine-readable. This system may leverage neural networks and artificial intelligence technologies as well.

# 1 Image processing

## 1.1 Image definition

An image is a visual representation or depiction of an object, scene, or concept. In the context of digital technology, a digital image refers to a binary representation of visual data that can be stored, manipulated, and displayed using electronic devices. Digital images can take various forms such as photographs, graphics, and individual frames of video [6].

There are two common ways to describe and represent images: vector graphics and raster graphics. In vector graphics, images are defined mathematically using geometric primitives such as points, lines, and curves. This allows for scalability and precise rendering at any size. On the other hand, raster graphics represent images as a grid of pixels, where each pixel carries color and intensity information. Raster images are also referred to as bitmaps [7].

An image map is a file or data structure that associated different regions or locations within a specified image with hyperlinks or other interactive elements. This allows users to interact with specific areas of an image, such as clicking on different parts to access related information or navigate to different web pages [8].

## 1.2 Image types

There are several common image types or formats used in digital imaging. Here are some of the most widely used image types:

- JPEG (Joint Photographic Experts Group): JPEG is a widely used image format that uses lossy compression. It is suitable for photographs and complex images with many colors and details. JPEG files have the extension ".jpg" or ".jpeg".
- PNG (Portable Network Graphics): PNG is a lossless image format that supports transparency. It is commonly used for graphics, icons, and images with sharp edges. PNG files have the extension ".png".

- GIF (Graphics Interchange Format): GIF is a compressed image format that supports animation and transparency. It is often used for small animations, logos, and simple graphics. GIF files have the extension ".gif".
- BMP (Bitmap): BMP is a simple image format that stores pixel data without compression. It is commonly used in Windows-based applications. BMP files have the extension ".bmp".
- TIFF (Tagged Image File Format): TIFF is a versatile image format that supports lossless compression and can store multiple images in a single file. It is often used in professional settings and for archival purposes. TIFF files have the extension ".tiff" or ".tif".
- RAW: RAW is a file format that contains unprocessed and uncompressed data directly from a camera's image sensor. It retains the most information and provides flexibility for post-processing. RAW files have various extensions depending on the camera manufacturer.
- SVG (Scalable Vector Graphics): SVG is an XML-based vector image format that can be scaled without losing quality. It is widely used for scalable graphics and logos on the web. SVG files have the extension ".svg".
- EPS (Encapsulated PostScript): EPS is a file format commonly used for vector graphics and illustrations. It supports both vector and raster elements and is widely supported in graphic design applications. EPS files have the extension ".eps".
- PDF (Portable Document Format): PDF is a versatile file format used for documents, including images. It supports various image types, text, and interactive elements. PDF files can be viewed and printed on different devices while preserving the original formatting. PDF files have the extension ".pdf".
- PSD (Photoshop Document): PSD is the native file format of Adobe Photoshop, a popular image editing software. It supports layers, masks, and various image editing features. PSD files are primarily used for editing and preserving editing capabilities. PSD files have the extension ".psd".
- WEBP: WEBP is a modern image format developed by Google. It provides both lossy and lossless compression options and supports transparency and animation. WEBP files are commonly used on the web to optimize image loading and performance. WEBP files have the extension ".webp".
- JP2 (JPEG 2000): JP2 is an image compression standard that offers superior compression efficiency and quality compared to traditional JPEG. It supports lossy and lossless compression, as well as advanced features like region of interest coding. JP2 files have the extension ".jp2" or ".j2k".

- ICO (Icon): ICO is a file format used for icons in Windows operating systems. It supports multiple image sizes and color depths, allowing icons to be displayed at different resolutions. ICO files have the extension ".ico".

## 1.3 Image processing

Image processing refers to the computational processing and analysis of digital images with the aim of extracting valuable information. In its processing, it is treated as two-dimensional signals and various Group are applied to them using signal processing techniques [9].

Common types of image processing include:

- Pattern recognition: involves measuring and analyzing patterns in an image. It can be used for various tasks such as selecting shapes, textures, or objects based on predefined patterns.
- Object Recognition: Object recognition is focused on detecting and recognizing objects within an image. It includes algorithms and techniques for identifying specific objects or classes of objects based on their features [10].
- Image retrieval: Image retrieval is a process of searching and browsing through a large image database to find images similar to a given query image. It may rely on content-based indexing and retrieval techniques to match features and similarities.
- Image sharpening and restoration: Sharpening and restoration techniques aim to improve image quality and detail. These methods can reduce noise, improve contrast, and restore missing or degraded parts of an image.
- Visualization: Visualization techniques are used to reveal hidden or invisible information in an image. It involves enhancing specific features or highlighting certain aspects to make them more visible and interpretable [11].

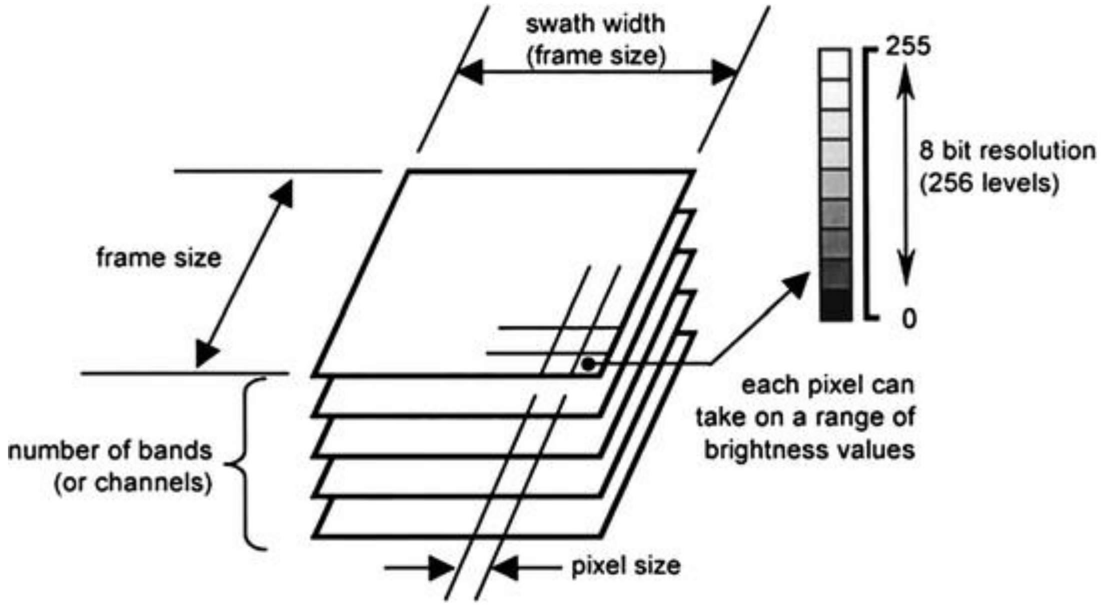


Fig1 digital image processing [88]

## 2 Historical documents

Historical documents refer to any type of written or printed material that provides information about past events, people, or cultures. They can include diaries, letters, manuscripts, newspapers, government records, and many other types of documents. Historical documents are valuable resources for researchers, historians, and other scholars who seek to understand and interpret the past.

Here are some definitions of historical documents from scholarly sources:

- "Historical documents are the written evidence of human society that have been preserved and transmitted through time." [12]
- "Historical documents are primary sources of information about the past that are created or produced at the time under study." [13]
- "Historical documents are documents that were created in the past and provide evidence of historical events, people, and cultures." (The National Archives, n.d.)
- Historical documents are important because they provide direct evidence of past events and can offer unique insights into the beliefs, values, and experiences of people from different times and places [14].

Fig2.historical documents [89]

## 2.1 Tasks on historical documents

The tasks associated with historical documents can vary depending on the specific document, here are some common tasks associated with historical documents:

**2.1.1 Preservation:** The preservation of historical documents is an important task that involves taking steps to protect the physical documents from damage or deterioration.

This can involve measures such as temperature and humidity control, proper handling techniques, and storage in archival materials. For more information on preservation techniques for historical documents [15].

Fig 3 Preservation Resources [90]

**2.1.2 Digitization:** The digitization of historical documents involves converting physical documents into digital form, which can make them more accessible and easier to preserve. For more information on digitization techniques for historical documents [16].

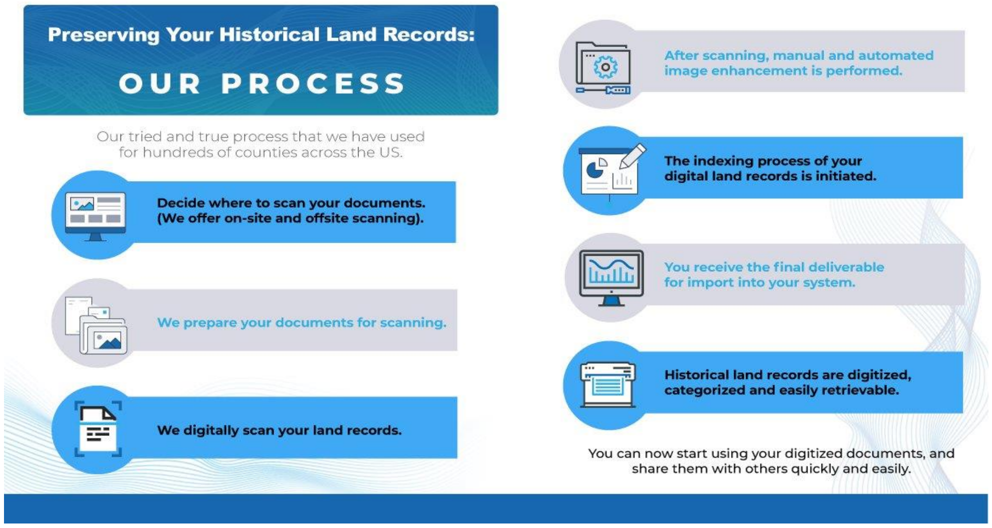

Preserving Your Historical Land Records:
OUR PROCESS
Our tried and true process that we have used for hundreds of counties across the US.
Decide where to scan your documents. (We offer on-site and offsite scanning).
We prepare your documents for scanning.
We digitally scan your land records.
After scanning, manual and automated image enhancement is performed.
The indexing process of your digital land records is initiated.
You receive the final deliverable for import into your system.
Historical land records are digitized, categorized and easily retrievable.
You can now start using your digitized documents, and share them with others quickly and easily.

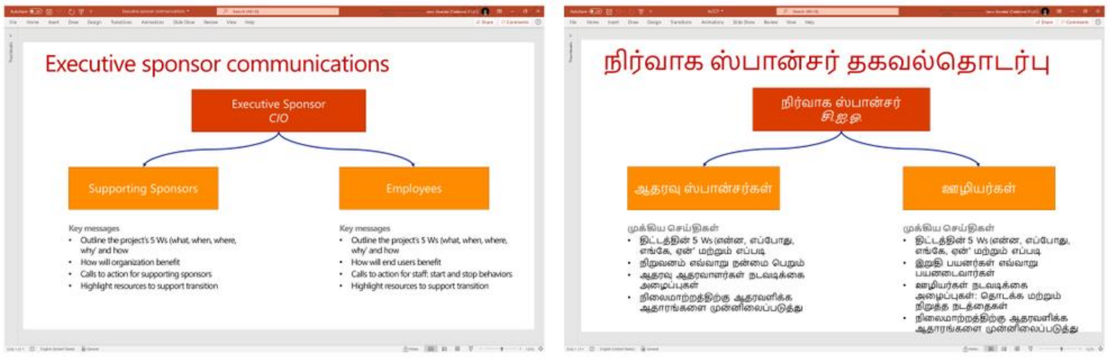
Fig4.digitization on historical documents [91]

**2.1.3 Transcription:** Transcription refers to the process of transforming handwritten or printed text into a format that can be read and understood by machines.

This can be necessary when dealing with historical documents that are difficult to read due to variations in handwriting, spelling, and punctuation. For more information on transcription techniques for historical documents [17].

Fig 5 online OCR Transcription Services [92].

**2.1.4 Translation:** Translation involves the process of converting text from one language to another. This can be necessary when dealing with historical documents that are written in languages that are no longer widely used. Here is an example: [18].



## Executive sponsor communications

Executive Sponsor
*CIO*

Supporting Sponsors

Key messages
- Outline the project's 5 Ws (what, when, where, why' and how
- How will organization benefit
- Calls to action for supporting sponsors
- Highlight resources to support transition

Employees

Key messages
- Outline the project's 5 Ws (what, when, where, why' and how
- How will end users benefit
- Calls to action for staff: start and stop behaviors
- Highlight resources to support transition

## நிர்வாக ஸ்பான்சர் தகவல்தொடர்பு

நிர்வாக ஸ்பான்சர்
*சி.ஐ.ஓ*

ஆதரவு ஸ்பான்சர்கள்

முக்கிய செய்திகள்
- திட்டத்தின் 5 Ws (என்ன, எப்போது, எங்கே, ஏன்' மற்றும் எப்படி
- நிறுவனம் எவ்வாறு நன்மை பெறும்
- ஆதரவு ஆதரவாளர்கள் நடவடிக்கை அழைப்புகள்
- நிலைமாற்றத்திற்கு ஆதரவளிக்க ஆதாரங்களை முன்னிலைப்படுத்து

ஊழியர்கள்

முக்கிய செய்திகள்
- திட்டத்தின் 5 Ws (என்ன, எப்போது, எங்கே, ஏன்' மற்றும் எப்படி
- இறுதி பயனர்கள் எவ்வாறு பயனடைவார்கள்
- ஊழியர்கள் நடவடிக்கை அழைப்புகள்: தொடக்க மற்றும் நிறுத்த நடத்தைகள்
- நிலைமாற்றத்திற்கு ஆதரவளிக்க ஆதாரங்களை முன்னிலைப்படுத்து

Fig 6 Translator announces Document Translation (Preview) [93]

**2.1.5 Data extraction:** Data extraction involves identifying and extracting specific pieces of information from historical documents. This can be useful for creating databases or for analysing trends over time. For more information on data extraction techniques for historical documents [19].

**2.1.6 Text normalization:** Text normalization involves standardizing the spelling and punctuation of text to make it easier to analyse. This can be important when dealing with historical documents that may use non-standard spelling or punctuation. For more information on text normalization techniques for historical documents [20].

Overall, the tasks associated with historical documents are varied and can involve a range of different techniques and technologies. However, the ultimate goal is to make these documents accessible and usable for research and scholarship.

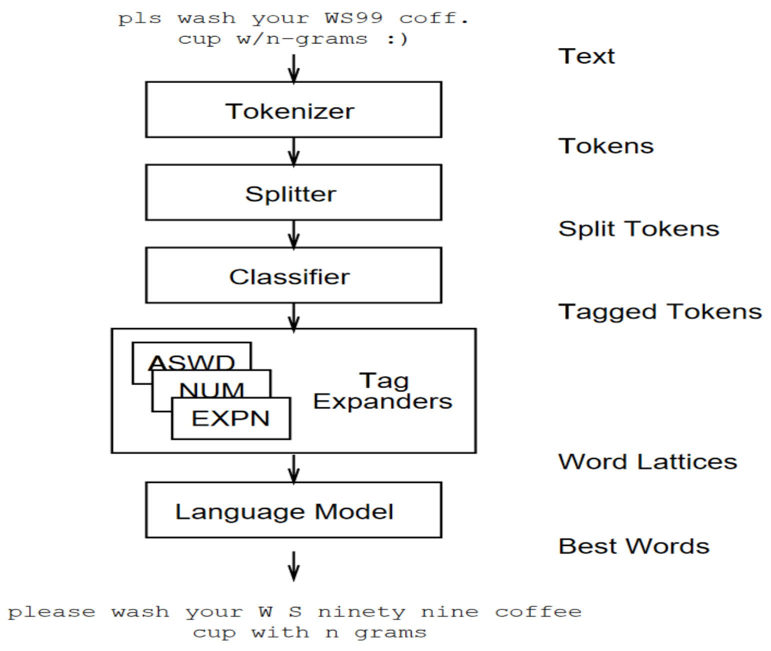


Fig7. Text normalization [94]

## 2.2 Importance of preprocessing methods

Preprocessing of historical documents is essential to making them accessible and usable. Here are some additional points on the importance of preprocessing to existing literature:

**2.2.1 Enhancing legibility:** As historical documents age, their text may become faded, discoloured, or obscured by stains, tears, or other damage. Pre-processing techniques such as image enhancement and restoration can also be used for clarity of text with high accuracy. For example, in a study on the restoration of old texts, researchers used image processing techniques to restore the text of a medieval manuscript that had faded due to water damage [21].

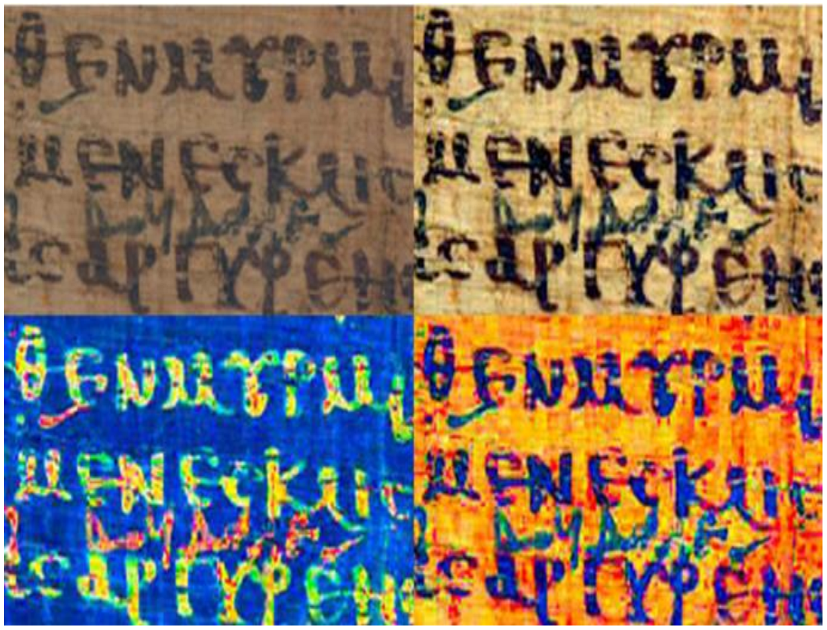


Fig8. Enhancing legibility of historical documents [95]

**2.2.2 Transcription:** Transcription of historical documents involves converting the text into a modern language or script, making it easier to understand and analyse. This can be especially important for documents written in languages that are no longer commonly used or in scripts that are no longer in use. For example, in a study on the transcription of old handwriting, researchers used deep learning algorithms to transcribe and translate the text of a 17th-century Dutch manuscript into modern Dutch [22].

**2.2.3 Data extraction:** Historical documents often contain valuable information such as names, dates, and places, but extracting this information can be challenging due to the structure and language of the text. Preprocessing techniques such as named entity recognition and information extraction can be used to automatically extract important information from the text. For example, in a study on the extraction of personal names from historical documents, researchers used Deep learning algorithms to transcribe extract names from a corpus of 19th-century German documents [23].

**2.2.4 Text normalization:** Historical documents can contain variations in spellings, abbreviations, and punctuation, which can make it challenging to analyse them. Preprocessing techniques such as text normalization and standardization can be used to reduce these variations, making the text easier to compare and analyse. For example, in a study on the normalization of historical texts, researchers used Deep learning algorithms to transcribe normalize and standardize the spelling of words in a corpus of English texts from the 17th and 18th centuries [24].

## 3 Conclusion

This chapter confirms the importance of image pre-processing and its impact on the quality of historical documents. It also highlights the tasks associated with historical documents and stresses the importance of addressing existing challenges and conducting further investigations to determine optimal solutions.

# Chapter 02: Artificial Intelligence And Preprocessing Techniques Overview

# Introduction

This chapter is dedicated to describe artificial intelligence with existing preprocessing techniques. We will also talk about recent works in this field, and then we will show a comparison between these works.

## 1 Artificial Intelligence

### 1.1 Introduction

Artificial intelligence, often abbreviated as AI, is the development of computer systems that can perform tasks that would typically require human intelligence. These tasks may include things like speech recognition, decision-making, language translation, and even creative activities like art and music. The goal of AI is to create machines that can work autonomously, adapt to new situations, and perform tasks that are too dangerous or time-consuming for humans.

AI has a rich history that dates back to the 1950s, but recent advancements in machine learning and deep learning have led to a surge in interest and investment in the field. With the help of large amounts of data and powerful algorithms, it has the potential to revolutionize many industries, from healthcare and transportation to finance and entertainment.

However, as with any new technology, it also brings its own set of challenges and concerns. These may include issues like privacy, bias, and job displacement. As AI continues to evolve and become more integrated into our lives, it will be important to address these challenges and ensure that AI is used in a way that benefits everyone [25][26][27].

### 1.2 Definition

Artificial Intelligence (AI) is a field within computer science dedicated to constructing intelligent machines capable of performing tasks that have traditionally required human intelligence. These tasks encompass areas like visual perception, speech recognition, decision-making, and language translation. The domain of AI is expanding swiftly, involving the creation and used of algorithms and computer programs that empower machines to learn from data, identify patterns, and make predictions or decisions [28].

## 2 History of Artificial Intelligence

Artificial Intelligence (AI) is a field of computer science that aims to create intelligent machines that can perform tasks that typically require human cognition, such as recognising speech, interpreting complex data, and making decisions. The idea of AI has been around for centuries, with roots in Greek mythology and medieval philosophy, but it wasn't until the 20th century that the field truly began to take shape.

The modern history of AI can be traced back to the 1950s, when computer scientists begin to explore the possibility of creating machines that could perform tasks that were previously thought to be exclusively human. In 1956, the Dartmouth Conference marked the birth of AI as a field, with computer scientists gathering to discuss the potential of creating "thinking machines." This led to the development of the first AI programs, such as the Logic Theorist and the General Problem Solver.

In the following decades, AI research saw several waves of optimism and disappointment, with funding and interest in the field ebbing and flowing. In the 1980s and 1990s, the focus shifted to expert systems and rule-based approaches, which were followed by neural networks and machine learning in the 2000s and beyond[29][30].

Today, AI is a fast growing field, with applications in a wide range of industries, including healthcare, finance, transportation, and entertainment. The development of deep learning algorithms, along with the exponential growth in computing power and the vast amounts of data available, has led to major advances in areas such as natural language processing, computer vision, and robotics [28].

## 3 Machine Learning

### 3.1. Definition

Machine learning is a subfield of artificial intelligence that involves designing and developing algorithms and techniques that enable computers to learn from data without being explicitly programmed. In other words, machine learning algorithms can improve their performance through experience, without human intervention [31].

### 3.2. Types of machine learning systems

There are several types of machine learning systems, including [32]:

#### 3.2.1. Supervised learning

Supervised learning involves using labeled data to train a machine learning model. Each training sample in the dataset is associated with a corresponding score or output value. Therefore, the algorithm learns to predict output labels or new and unseen data values.

#### 3.2.2. Unsupervised learning

Unsupervised learning involves the use of unlabeled data to train a machine learning model. The algorithm attempts to learn the basic characteristics or distributions contained in the data, without any labels or output values to guide it. Clustering and dimensionality reduction are common examples of unsupervised learning techniques.

#### 3.2.3. Reinforcement learning

Reinforcement learning involves the use of trial-and-error to learn optimal behavior. The algorithm determines what actions to take in a given situation to maximize a reward (in the form of a number) in order to achieve a specific goal. This type of learning is often used in robotics and game-playing applications.

# 4 Deep Learning

Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers to learn and make predictions from complex and large-scale data. Deep learning algorithms are designed to automatically learn hierarchical representations of data by progressively extracting higher-level features [33].

## 4.1. Neural Networks

Neural networks form the foundation of deep learning. They consist of interconnected nodes, called artificial neurons or units, organized in layers. The input layer receives the raw data, which is then passed through multiple hidden layers, and finally, the output layer produces the desired predictions or classifications.

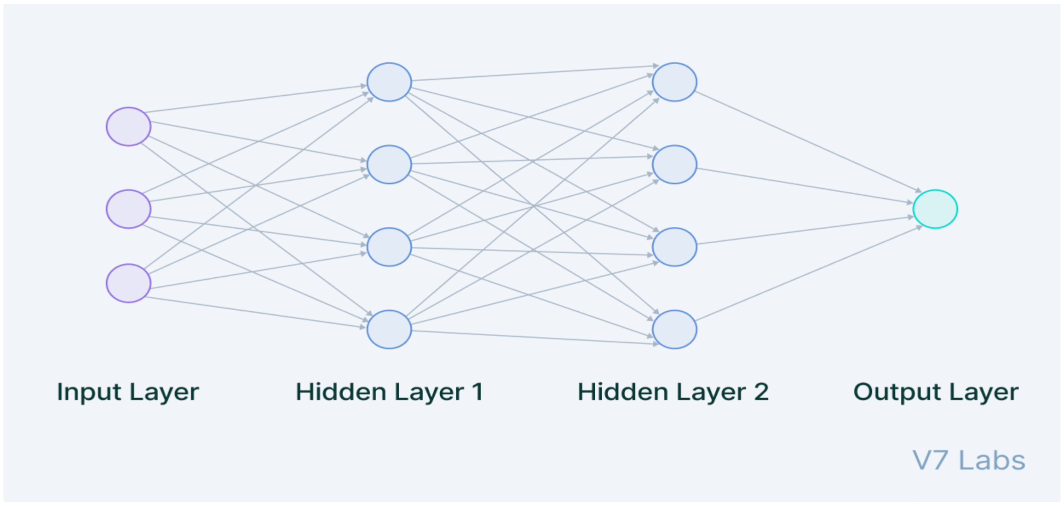


Fig 9. The Architecture of Neural Networks [96]

## 4.2 Deep Neural Networks

Deep neural networks refer to neural networks with many hidden layers, allowing them to learn increasingly abstract and complex representations of the input data. The depth of the network enables it to capture intricate patterns and dependencies in the data, leading to improved performance in tasks such as image and speech recognition, natural language processing, and many others [34] [35].

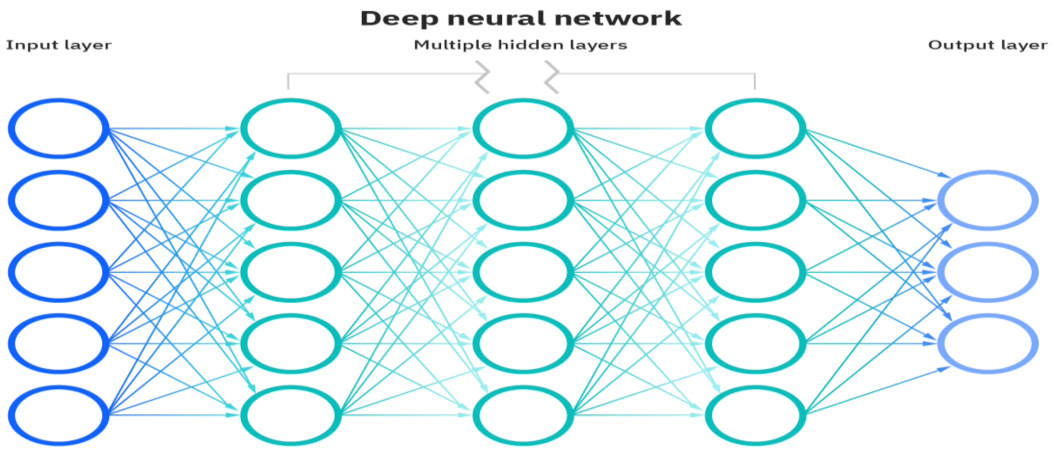


Fig 10. Deep Neural Networks [97]

### 4.2.1 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a specialized type of deep neural networks commonly used for analysing visual data. They are designed to automatically learn and extract spatial hierarchies of features from images through repeated application of convolutional and pooling operations.

### 4.2.2 Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) are another type of deep neural network well suited for processing sequential data, such as time series or natural language. RNNs have recursive connections that allow them to retain memory of previous inputs, and enable them to model temporal and context dependencies in data.

# 5 Preprocessing techniques overview

Preprocessing techniques play a crucial role in character recognition systems, where they are applied to color or gray-level document images containing text and/or graphics. Due to the computational demands of color images, grayscale or binary images are commonly used in such systems. However, these images often present challenges such as non-uniform backgrounds and watermarks, making text extraction difficult without preprocessing. The objective of preprocessing is to obtain a binary image that exclusively contains the text.

To achieve this, a series of steps are employed. First, image enhancement techniques are applied to reduce noise and correct contrast. Then, thresholding is performed to eliminate background elements like scenes, watermarks, and noise. Page segmentation follows, separating graphics from text. Next, character segmentation is employed to isolate individual characters, and morphological processing is used to improve characters affected by thresholding or other preprocessing techniques. These technologies offer various options for integration into character recognition systems, and their application may vary at different stages of an OCR system. Subsequent sections will delve into some of these mentioned technologies.”

## 5.1 Image enhancement techniques

In the field of image processing, image enhancement techniques are used to improve the quality of images to improve human perception. These technologies include a combination of actions, including noise removal, darkening reduction, contrast enhancement, and detail amplification. This section delves into an exploration of several commonly used image enhancement techniques, highlighting their application and effectiveness in improving the visual quality of images.

### 5.1.1 Spatial image filtering operations

Filters play a crucial role in image processing, serving the purpose of either smoothing the image by reducing high frequencies or enhancing it by accentuating low frequencies and detecting edges. Image restoration and enhancement techniques can be performed in either the spatial or frequency domain, with the frequency domain approach often utilizing Fourier transforms as a popular method. However, Fourier transforms can be computationally intensive and may not always be necessary. In contrast, spatial filtering techniques, which involve convolution with small masks like 3x3 filters, offer a faster and simpler alternative. This chapter focuses exclusively on spatial filtering techniques, as they are effective for image processing tasks. Images captured may contain noise, which can degrade their quality and hinder analysis. Additionally, certain regions within an image may require specific emphasis or highlighting. Spatial processing techniques can be broadly classified into two categories: point processing, which involves transforming individual pixels independently, and mask processing, which considers a pixel along with its neighboring pixels within a defined mask to generate an enhanced pixel at specific coordinates (x, y) in the resulting image.

#### 5.1.1.1 Point processing

Point processing is a basic technique in image processing that falls under the category of spatial processing. It involves modifying the values of individual pixels in an image independently to improve desired features. Mathematically, this can be expressed using the following equation (1):

$$\text{Enhanced\_pixel}(x,y) = f(\text{original\_pixel}(x,y)) \qquad (1)$$

In equation (1), enhanced_pixel(x, y) represents the pixels at the (x, y) coordinates in the resulting enhanced image. original_pixel(x, y) represents the corresponding pixel in the original image. The transformation function f ( ) is applied to the original pixel to determine its modified value in the enhanced image. By applying point manipulations, such as contrast adjustment, brightness correction, and color manipulation, to each pixel independently, specific improvements can be achieved in local areas of the image. This flexibility allows for targeted improvements and enables customization of image enhancement techniques based on specific requirements.

#### 5.1.1.1.1 Contrast stretching

Contrast stretching is a type of point processing technique in image processing used to improve the contrast of an image by expanding its dynamic range. The dynamic range refers to the range of pixel intensities in an image, from the darkest to the lightest values. When an image has low contrast, this means that the range of intensities is relatively narrow, making it difficult to distinguish between different parts of the image. Contrast stretching can be used to increase the range of pixel intensities, making the differences between different parts of the image more noticeable.

This technique involves applying a linear transformation to the pixel values in an image, such that the darkest pixel value is mapped to black (0) and the lightest pixel value is mapped to white (255), with all other pixel values scaled proportionally between them. This extends the intensity range across the entire spectrum, resulting in an image with enhanced contrast. The linear transformation is typically performed using a simple formula, such as (2):

$$O(x,y) = (I(x,y) - I\_min) * (255 / (I\_max - I\_min)) \qquad (2)$$

Where O(x, y) is the output pixel value at location (x,y), I(x,y) is the input pixel value at location (x,y), I_min is the minimum pixel value in the image, I_max is the maximum pixel value in the image, and 255 is the maximum possible pixel value in an 8-bit grayscale image.

#### 5.1.1.1.2 Global image thresholding

Global image thresholding is a technique used in image processing to separated objects of interest from the background based on the intensity values of pixels. Typically applied to gray-level or color document scanned images, it involves assigning pixels to either the foreground or background region based on a predetermined threshold value. One commonly used method for determining the threshold value is Otsu's method, which calculates the threshold based on the image's histogram.

The iterative procedure for global thresholding is as follows:

**Step 1:** Choose an initial threshold value T.

**Step 2:** Separate the pixels of the input image into two groups: those with intensity values greater than T, and those with intensity values less than or equal to T.

**Step 3:** Calculate the average intensity values, or means, of each of the two groups.

**Step 4:** Calculate a new threshold value T_new as the average of the two means (4):

T_new = (mean1 + mean2) / 2 (4)

Where mean1 and mean2 are the mean intensity values of the two pixel groups.

**Step 5:** Repeat steps 2-4 until the difference between T and T_new is smaller than a predefined tolerance value.

Where many thresholding methods have been published, for example, [36]. Comparing the performance of more than 20 global threshold algorithms using standardized metrics or shape metrics. The comparison showed that the Otsu separation method gave the best performance [36], [37]. On the other hand, in an evaluation of change detection by Rosin & Ioannidis concluded that Otsu algorithm has very poor performance compared to other universal methods [38], [37]. Trier and Jain's OCR evaluation study examined four universal techniques by which the Otsu method was shown to be superior to the other methods examined during the study [39]. In addition, Fisher compared 15 universal methods and confirmed that Otsu's method is preferred in document image processing [40]. Otsu's method is a widely used technique for converting a gray level image into a binary image and then calculating the optimal level separating these two classes such that their co-diffusion (interlayer contrast) is minimal. Otsu's method searches for the limit that minimizes intra-class variance, defined in equation (5) as a weighted sum of the variances between the two classes [37].

$$\sigma^2{}_{\omega}(t) = \omega\ (t)\sigma^2(t)_{\ (1)} + \omega\ (t)\sigma^2(t)_{\ (2)} \qquad (5)$$

The weights ωi are the probabilities of the two categories separated by a threshold t and 2 σ i is the variance of these categories. Otsu shows that reducing variance within a category is the same as maximizing variance between categories, Equation (6):

$$\sigma_b{}^2(t) = \sigma^2 - \sigma_{\omega}{}^2(t) = \omega_{(1)}\ (t)\omega_{(2)}\ (t)[\mu_{(1)}\ (t) - \mu_{(2)}(t)]^2 \qquad (6)$$

This is expressed in terms of class probabilities ωi and class mean μi which in turn can be updated iteratively. The steps of the algorithm are:

- Calculate the histogram and probabilities for each intensity level
- Initialize ωi(0) and μi(0)
- Exceed all threshold values t = 1 …. To maximum intensity.
- Update ωi(0) and μi(0) - Calculate the maximum ( ) 2 t σ b , which corresponds to the desired threshold

**Step 6:** Set all pixels with intensity values greater than or equal to T to a value of 255 (white), and all pixels with intensity values less than T to a value of 0 (black), resulting in a binary image.

#### 5.1.1.1.3 Histogram processing

Histogram processing plays an important role in image optimization, as well as image compression and segmentation processing. It includes graph analysis and processing, which represents the frequency distribution of different values on the gray level in the image. The histogram provides valuable insights into the pixel intensity distribution, ranging from 0 (representing black) to 255 (representing white). Scanned or captured images often suffer from a limited color range or low contrast, resulting in a lack of detail. By applying graph processing techniques, the image can be optimized to improve the level of detail and facilitate subsequent machine vision tasks, such as segmentation.

#### 5.1.1.1.3.1 Histogram equalisation

Histogram equalization is a widely used global technique in image processing. It’s main objective is to stretch the histogram of an image across between range of pixel values (0 - 255). By doing so, it effectively enhances the contrast of the image, making it more visually appealing for human inspection. Furthermore, histogram equalization can be employed to normalize variations in illumination, thus aiding in image understanding tasks. The process it self is relatively straightforward. For each brightness level 'j' in the original image, the new pixel level value 'k' is calculated using the equation (7):

$$k = \sum_{i=0}^{i} Ni \;/\; T \qquad (7)$$

In this equation, the sum considers the number of pixels in the image, obtained by integrating histograms, with a brightness value equal to or less than “j”. The letter “T” represents the total number of pixels in the image [41]. In addition, graph equalization serves as a basic operation that can be used to generate new images based on specification or modification of the graph.

Therefore, histogram equalization provides a powerful means to enhance image contrast and to manipulate illumination differences in image comprehension tasks. Its simplicity and effectiveness make it a valuable tool in image processing. [41]

### 5.1.1.1.3.2 Histogram specification (Matching)

Histogram matching is an image processing technique used to adjust the color distribution of two images by making use of their image histograms.

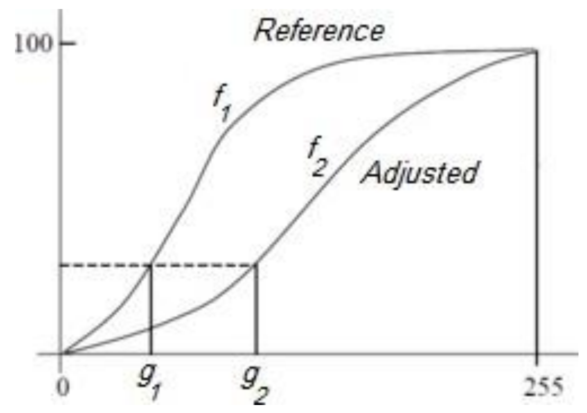


Fig 11 shows the cumulative distribution functions for both the reference and normalized images [98].

Adjusting the histogram may involve aligning the cumulative function f2 of the image to be mapped using the cumulative distribution function (CDF) of the reference image, denoted by f1. To achieve histogram matching, histograms are calculated for both images, followed by the calculation of the CDF of the reference image (f1) and the image to be modified (f2). The goal of graph matching is to match the closest CDF f2 to the reference image CDF f1. For each gray level g1, the corresponding gray level g2 is defined as f1(g1) = f2(g2), as shown in Figure 11. In this process it produces the result of the graph matching function, denoted by M(g1) = p2 [42 ].

### 5.1.1.1.4 Log transformations

Log transforms are a specific type of point processing technique used in image processing to improve the contrast of images that have low dynamic range. This technique involves applying a logarithm function to the pixel values of the input image and later adjusting the values to a new range.

The log transformation formula is expressed as follows (8):

$$O(x,y) = c * \log(1 + I(x,y)) \qquad (8)$$

Here, the pixel value of the output image at the (x, y) position is denoted as O(x, y), and the corresponding input image pixel value at the same position is represented as I(x, y). The constant 'c' is used to scale the resulting image intensity values to the desired range.

Through the use of log-shifting, the dynamic range of the input image is compressed, resulting in improved contrast in lower intensity regions with in the image. This transformation makes it easier to visualize and analyse the fine details and subtle differences in the image.

#### 5.1.1.1.5 Power law transformation

The power law transformation, also referred to as gamma correction, is a point processing technique employed to enhance images by mapping the original pixel values to new values using a power-law function.

The power law transformation function is generally represented as follows (9):

$$s = cr\hat{}\gamma \qquad (9)$$

In this equation denotes the output pixel value, "r" represents the input pixel value, "c" is a constant, and "" corresponds to the gamma value. The gamma value determines the degree of enhancement applied to the image. When $\gamma$ is less than 1, the resulting image appears darker, while values greater than 1 produce a brighter image.

Power law transformation serves various purposes such as correcting for nonlinear elements in the image sensor's density response, adjusting the brightness and contrast of an image, or accentuating specific features within an image. By manipulating the gamma value, different visual effects can be achieved, allowing for customized optimization based on the specific requirements of the image processing task.

### 5.1.2. Local thresholding

Local thresholding techniques are used to segment images of documents with non-uniform backlighting or complex backgrounds, including watermarks commonly found on security documents. In situations where global thresholding methods encounter graphical images displaying multiple peaks, the precise separation of foreground and background becomes very complex. The complexity introduced by the multiple peaks hinders the effectiveness of global thresholding techniques in distinguishing objects from the background. It is important to note that local thresholding methods are designed for specific applications and may not work well across different scenarios. Local threshold results can vary, increase or decrease, depending on the contrast and brightness characteristics of the image.

Several surveys have been conducted to compare different thresholding techniques. For example, Trier and Jain conducted an evaluation to evaluate the performance of 11 adaptive dual modalities widely recognized at the national level [39].

Their evaluation focused on the ability of the OCR unit to accurately recognize handwritten numbers from hydrographic images. According to their findings, the Niblack method [43] showed to be the best performer among the evaluated techniques. However, it is important to bear in mind the authors' conclusion that this observation may not hold when different sets of images are used in combination with various feature extraction methods and classifiers. Therefore, the selection and effectiveness of local threshold methods can depend on the specific characteristics and requirements of the specific application.

## 5.2 Noise Removal

Noise removal is an important step in image processing to enhance the reliability and robustness of OCR systems, especially when images contain various sources of noise, including random variations such as photon noise, thermal noise, on-chip electronic noise, and measurement noise. While some noise sources can be mitigated through sensor or CCD camera improvements, some noise components persist, necessitating the application of noise reduction techniques.

One of the prevalent types of noise in binary images is isolated pixels or salt and pepper noise, which can be effectively removed through a process known as padding. This approach involves filling each isolated pixel "island" with surrounding "sea" pixels [44]. In grayscale images, both intermediate filters and low-pass filters such as median blur and Gaussian blur have proven effective in removing isolated pixel noise.

Low-pass filters, including Gaussian blur and median filters, work by smoothing images by averaging adjacent pixel values. The Gaussian filter is determined by formula (11):

$$G(x, y) = (1/(2\pi\sigma^2)) * e^{(-(x^2 + y^2)/(2\sigma^2))} \qquad (11)$$

Here, x and y represent the distance from the center of the filter, and denote the standard deviation of the Gaussian distribution. Filter average, on the other hand, simply averages the pixel values within the filter window. By moving the filter window over each pixel in the image, these filters can be applied to calculate the filtered value.

Additional noise reduction techniques include wave-based methods that take advantage of a multiscale approach to suppress noise while preserving edges and image detail. These methods are particularly effective at reducing noise within textured areas of an image. Moreover, noise reduction algorithms based on deep learning have shown promising results in effectively reducing noise.

The choice of an appropriate noise reduction technique depends on the specific type of noise present and the characteristics of the image being processed. Different technologies may show different levels of effectiveness across different applications and should be chosen accordingly.

## 5.3 Skew detection/correction

Skew in document images is a common problem as it has a detrimental effect on the accuracy of different document image analysis methods. To address this problem, deviation detection and correction techniques are used. Also, there are various methods for detecting deflection, such as drop profile analysis, Hough transform, connected components, clustering, and linear correlation. In addition, specific applications have introduced new technologies, such as the center-of-gravity method specifically designed for Arabic document images. However, the accuracy of most of these techniques is limited, usually in the 0.1° range. As a result, further research is necessary to improve their performance in dealing effectively with deviance-related challenges. The choice of aberration detection and correction technique should be based on the specific requirements of the application and the characteristics of the images being considered.

## 5.4 Page segmentation

To improve page segmentation, one can consider using more advanced and modern techniques such as deep learning based methods. Convolutional neural networks (CNNs) have been shown to perform well on page segmentation tasks, especially in dealing with complex and irregular document layouts. For example, the use of fully convolutional networks (FCN) for page segmentation has been proposed in recent studies [45]. Another approach is to use generative adversarial networks (GANs) to generate realistic document images, which can then be used to train page segmentation models [46]. These methods can be trained on a variety of document images to make them more powerful and efficient for a variety of document types and formats.

In addition, it may be useful to combine multiple segmentation methods to improve accuracy, such as combining a top-down and bottom-up approach as described in the Kruatrachue and Suthaphan method mentioned in the text. Finally, it is important to consider pre-processing steps, such as skew detection and correction, to ensure the best possible segmentation results.

## 5.5 Image size normalization

After splitting the characters, the resulting orphan characters are ready for the feature extraction stage. In order to achieve data consistency, it is usual to normalize orphaned characters to a predetermined size. This size can be determined by empirical or empirical means, and the requirements of the application and also the feature extraction or classification techniques used must be considered. Then, orphan characters undergo trait extraction to extract relevant characteristics that can be used for subsequent analysis or classification purposes.

# 6 Recent works

### 6.1 Apostolos Antonacopoulos, Basilis Gatos, and al (2016)

This work is proposed a deep learning-based method for page segmentation of historical document images. The proposed method uses a Convolutional Neural Networks (CNNs), have been applied for feature extraction and classification of medieval manuscripts [47].

The dataset used are:

- BnF dataset: A collection of manuscripts from the Bibliography nationale de France (BnF))
- Leuven dataset: A dataset from the University of Leuven containing medieval manuscripts
- Bibale dataset: A dataset from the BIBALE project containing manuscripts from the Gironde Archives departments. And this winner of this competition NNML 83.80%

### 6.2 M. Seuret and al. (2017)

This work [48] is proposed a deep learning-based method for page segmentation of historical document images. The proposed method uses a convolutional neural network (CNN) to predict the regions of interest in the document image, such as text regions, image regions, and background regions. The preprocessing step in this method includes image normalization and binarization, which take up to 10% of the preprocessing time. The proposed method achieves state-of-the-art performance on several benchmark and datasets used are: G.Washington 89%, Parzival 64% and St. Gall 89%.

### 6.3 G. Renton and al. (2017)

This work [49] is proposed a deep learning-based method FCN, a type of neural network specifically designed for pixel-wise predictions, to accurately segment text lines in handwritten documents. The main objective of their work is to automatically extract individual lines of text from handwritten document images, which can be a challenging task due to variations in writing styles, overlapping text, and other complexities inherent to handwritten documents.The dataset used a different dataset competition (ICDAR 2017) that gave 93%

### 6.4 A. Fischer and al. (2018)

This work [50] is proposed a fully convolutional neural network (FCN) for text line segmentation in historical handwritten documents. The proposed method uses an FCN to predict the pixel-wise text line labels in the document image. The preprocessing step in this method includes image normalization and binarization, which take up to 15% of the preprocessing time. The proposed method achieves state-of-the-art performance on several benchmark datasets used are: G.Washington 87%, Parzival 91% and St. Gall 95%.

### 6.5 B. barakat, and al. (2018)

This work [51] is proposed a method leverages capabilities of FCN witch are deep neural networks specifically designed for pixel-wise predictions. In this case, the FCN is trained to predict the pixel-level labels that correspond to the boundaries of text lines in the document image. The dataset used is

Islamic heritage project (IHP) and validation 88%.

### 6.6 R.Alaasam and al. (2019)

This work [52] is proposed a method siamese network architecture, which is a type of neural network

Designed for learning similarity or dissimilarity between input pairs . The siamese network is trained to

Comparison different regions within the manuscript and learn the distinctive features that enable layout analysis. The dataset used Arabic manuscript that gave 93%.

### 6.7 A. Singh and M. Vatsa (2019)

Provides this work [53] a comprehensive review of deep learning-based methods for handwritten text recognition. The review covers the preprocessing steps, including image normalization, binarization, line segmentation, and word segmentation, that are commonly used in these methods. The authors analyse the performance of various deep learning-based methods on several benchmark datasets and highlight the strengths and weaknesses of these methods.

### 6.8 G. Gatos and al. (2020)

This work [54] is proposed a deep learning-based method for handwritten text recognition on medieval manuscripts. The proposed method uses a CNN and a bidirectional LSTM (BLSTM) network to recognize the text in the document image. The preprocessing step in this method includes image normalization, binarization, and line segmentation, which take up to 30% of the preprocessing time. The proposed method achieves state-of-the-art performance on several benchmark datasets of medieval manuscripts.

### 6.9 S. S. Suresh and S. Sundaram (2020)

Provides this work [55] a comprehensive survey of deep learning-based methods for handwritten text recognition. The survey covers the preprocessing steps, including image normalization, binarization, line segmentation, and word segmentation, that are commonly used in these methods. The authors analyse the performance of various deep learning-based methods on several benchmark datasets and highlight the recent advancements and challenges in this field.

### 6.10 L. G. Monwar and al. (2020)

Provides this work [56] a comprehensive survey of methods for handwritten text recognition on historical documents. The survey covers traditional methods as well as deep learning-based methods, and discusses the preprocessing steps, including image enhancement, binarization, and segmentation, that are commonly used in these methods. The authors analyse the performance of various methods on several benchmark datasets and highlight the recent advancements and challenges in this field.

## 7 Comparison of works

A summary of notable techniques based on preprocessing impact on historical document methods is presented in **Table 7** it can be seen that although the performance of different studies varies depending on the technique as well as the dataset used, in general, research is being done to find robust solutions to this problem

| Artical | Methods | Accuracy% |
|---|---|---|
| **6.1 Apostolos Antonacopoulos, Basilis Gatos, and al(2016) [47]** | Convolutional feature extraction and classification of medieval manuscripts. | 83.80% |
| **6.2 M. Seuret and al. (2017)[48]** | convolutional neural network (CNN) to predict the regions of interest in the document image, such as text regions, image regions, and background regions | G.Washington 89%, Parzival 64% St. Gall 89%. |
| **6.3 G. Renton and al. (2017) [49]:** | method FCN, a type of neural network specifically designed for pixel-wise predictions, to accurately segment text lines in handwritten documents | ICDAR2017 93% |
| **6.4 A. Fischer and al. (2018) [50]:** | fully convolutional neural network (FCN) for text line segmentation in historical handwritten documents. | G.Washington 87%, Parzival 91% St. Gall 95%. |
| **6.5 B. barakat, and al. (2018) [51]:** | FCN witch are deep neural networks specifically designed for pixel-wise predictions | 88% |
| **6.6 R.Alaasam and al. (2019)[52]:** | . method siamese network architecture, which is a type of neural network Designed for learning similarity or dissimilarity between input pairs | 93% |
| **6.7 A. Singh and M. Vatsa (2019)[53]:** | comprehensive review of deep learning-based methods for handwritten text recognition. | / |

| | | |
|---|---|---|
| **6.8 G. Gatos and al. (2020)[54]:** | The proposed method uses a CNN and a bidirectional LSTM (BLSTM) network to recognize the text in the document image. | / |
| **6.9 S. S. Suresh and S. Sundaram (2020)[55]:** | Comprehensive survey of deep learning-based methods for handwritten text recognition. | / |
| **6.10 L. G. Monwar and al. (2020)[56]:** | Comprehensive survey of methods for handwritten text recognition on historical documents. | / |

**Table 7** Comparison of works

## 8 Conclusion

In this chapter, we have dealt with artificial intelligence, got acquainted with many of its technologies, also challenged us with some pre-processing methods, and we also talked about recent works.

# Chapter 03:

# Adopted

# Approach

## Introduction

This chapter is dedicated to talking about the classification of historical documents, especially in terms of history (**chronology**), as well as presenting the system applied by the MPS data set. We also show some pre-processing techniques applied in this system.

## 1 Historical documents classification

Classification of historical documents refers to the process of categorising or organising historical documents into different classes or classes based on their content, characteristics, or other relevant features. It also involves the use of computational methods and techniques to analyse, interpret and categorize large volumes of historical documents [57].

Tasks in the classification of historical documents can also vary depending on the goals and specific requirements of the analysis. Some common tasks in the classification of historical documents include:

- Document Classification: This task is to assign historical documents to predefined classes or categories based on their subject matter, genre, authorship, and especially the time period. For example, categorising documents on topics such as politics, religion, or art, or categorizing them by the time period in which they were created.
- Text classification: Text classification focused on categorizing historical documents based on the textual content they contain. It includes analysis of language, vocabulary, and textual styles to tailor documents to specific categories. For example, categorizing documents as letters, memorandums, newspapers, legal documents, or literary works.
- Selected Entity Identification: Specific Entity Identification involves identifying and categorizing named entities such as the names of people, organizations, locations, dates, or other significant entities mentioned in historical documents. It helps extract valuable information and facilitate further analysis.
- Language identification: Language identification is the task of determining the language in which the historical document was written. It is particularly suitable for multilingual groups and helps in organizing and retrieving documents based on language.

The data used to classify historical documents typically consists of digital copies of the documents themselves, along with their associated metadata. The data may include full text, images, or both. In addition, relevant attributes such as document title, author information, year of publication, and other contextual details are often included as part of the data.

The years or time periods associated with historical documents serve as important metadata for classification. They can be used to define time categories, analyse historical trends, or group documents based on specific time periods.

Classification of historical documents also plays an important role in digital humanities, archival research, cultural preservation, and historical analysis. It enables the efficient organization, retrieval, and analysis of large collections of historical documents, providing researchers and historians with valuable insights into the past.

## 1.1 Description of tasks

### Chronology (Dating):

The classification of historical documents is considered in terms of the historical sequence, as it prompts the process of organizing and classifying them based on their chronological order or the time period of creation [58]. It involves mapping specific time periods or historical eras in order to facilitate analysis and understanding of historical events, cultural changes, or societal developments over time.

This type of classification is particularly valuable for studying historical trends, tracing the development of ideas or practices, as well as studying the development of historical events. This allows researchers and historians to identify and explore patterns, connections, and shifts in the historical context.

The classification process may involve different methods, including manual annotations by experts, automated methods using machine learning algorithms, or a combination of both. Historical metadata such as document dates, years of publication, author information, or contextual clues within the documents themselves can be used to infer the time period to which documents belong.

By categorizing historical documents in terms of chronology, researchers can gain insights into the evolution of societies, cultures, ideologies, and technologies across different time periods. This helps organize, retrieve and analyse historical data, and supports a deeper understanding of the past and its impact on the present.

## 2 Global architecture

In this part, we will attempt an overview of our system, the latter consisting of pre-processing methods on historical documents and chronologically classifying them using the MPS dataset, 3 classes were used. After reading it, it is divided into two parts (training and 20% validation), and the image is formatted in terms of equal dimensions.

Then we apply these techniques to them in order to note the result to be obtained, as well as to note the difference between applying pre-processing methods to historical documents as well as not applying them, and in the end they are classified with the described tasks (chronological sequence), as show in Fig 12 below.

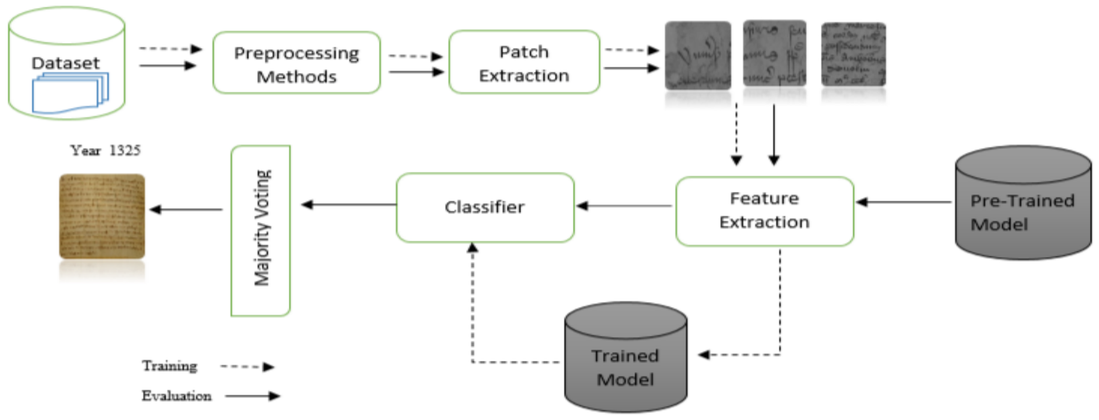


Fig 12 view general of system

## 3 Dataset

The pilot study used a 'Medieval Palaeographic Scale (MPS)' dataset entered [59] to validate the system. This dataset consists of 3,267 images of charters written during the period from 1300 to 1550 AD. Charters have been collected from four prominent cities, 'Arnhem', 'Leiden', 'Leuven' and 'Groningen', which represent different corners of 'medieval Dutch'.

Region. Figure 13 presents a selection of charter images from the four cities included in the MPS dataset.

Fig13.sample images of charters from medieval paleographic scal dataset [59]

## 4 Preprocessing adopted techniques

### 4.1 Binarization and remove noise

Binarization and noise removal are two important image processing techniques that have a significant impact on historical documents.

**Binarization** Binary encoding refers to the process of converting a grayscale or color image into a binary image, in which each pixel is assigned a white or black value based on a certain threshold. In the context of historical documents, binary coding is often used to separate the foreground (text or handwriting) from the background, making it easier to analyse and manipulated the textual content. Binary coding improves documents readability by improving contrast between text and background. **Noise removal**, On the other hand, noise removal involves removing or reducing unwanted noise or noise in an image. In historical documents, noise can arise from various sources such as paper aging, ink fading, wiping artifacts, or physical damage to the document. Noise can significantly reduce the clarity and quality of a document, making it difficult to extract useful information. By applying noise removal techniques, such as filters or image enhancement algorithms, unwanted noise can be reduced, resulting in a cleaner and more visually appealing document.

The combination of binary coding and noise removal techniques is critical in improving the visual quality, clarity, and analysis of historical documents. These pre-processing steps are often performed as part of document digitization and preservation efforts, enabling researchers, historians, and conservators to better study, interpret, and preserve the valuable historical information found in these documents.

## 4.2 Deskew function (remove noise and rotate)

The deskew function, which involves noise removal and rotation, has a significant impact on historical documents.

The deskew function not only addresses noise removal but also corrects the rotation or slant present in the document. Skew refers to the misalignment or tilting of the text or content in the document, which can occur during the scanning process or due to document handling over time. Skew can make the document difficult to read and analyse accurately.

By applying noise removal techniques, such as filters or noise reduction algorithms, the skew function eliminated unwanted defects and disturbances in the document image. This process improves the readability of the text and ensures that the content stands out more prominently, improving the overall quality of the document.

In addition, the keystone correction function corrects the rotation or tilt of the document. This includes determining the angle of rotation and applying the appropriate transformation to align the text correctly. By aligning the text horizontally or vertically, the error correction function makes it easy to read and interpret the document accurately.

The combined effect of noise removal and rotation correction in the skew function greatly improves the overall quality, clarity and readability of historical documents. It promotes the preservation, analysis, and dissemination of historical information, making it accessible to researchers, historians, and the general public. Through noise reduction and rotation correction, the skew function plays an important role in unlocking the valuable content and knowledge contained in historical documents.

## 4.3 Each image in the dataset and perform the intensity stretching

Performing an intensity stretch on each image in the dataset has a significant impact on historical documents.

Density stretching, also known as contrast stretching, is a technique used to improve visual contrast and improve the overall look of images. It adjusts the range of intensity values present in an image to cover the full dynamic range, thus increasing the distinction between different levels of brightness and improving detail within an image.

When applied to historical documents, extreme stretching can help reveal finer details, improve readability, and improve legibility of text and content. Historical documents often suffer from deterioration, such as fading, discoloration, and uneven lighting, which can lead to poor contrast and reduced clarity of content.

By stretching the intensity values, dark areas become darker, and bright areas become brighter, effectively increasing the contrast between foreground and background elements in your document. This allows a clearer visualization of the text, illustrations, and other content in the historical document.

Intensity expansion also helps bring out fine features and subtle details that might otherwise be obscured or difficult to perceive. It enhances the visibility of faded handwriting, intricate patterns, and delicate textures, which are often so prevalent in historical documents.

Moreover, extreme stretching can help restore and preserve historical documents. By improving the overall appearance and readability, it enables scholars, conservators, and historians to better analyse and interpret the content. It facilitates the extraction of valuable information, the identification of the historical context, and contributes to the digitization and publication of historical documents.

In general, applying hyper-stretching to each image in a historical document dataset enhances contrast, improves visibility, reveals fine details, and helps preserve and analyse valuable historical information.

## 4.4 Method top-hat transform

Top-hat conversion is a method that can have a significant impact on historical documents by improving the contrast of text and other dark areas within documents.

Top hat transformation is a morphological process in image processing that extracts small, bright details from a given image while preserving the overall structure. It is particularly useful in applications where the focus is on enhancing fine details or extracting specific features that may be hidden or less noticeable due to differences in lighting or noise.

In the context of historical documents, top-hat conversion can help improve legibility and legibility of text and other important content. Historical documents often suffer from deterioration, including uneven lighting, discoloration, and ink fading, which can reduce contrast and make it difficult to discern fine details.

By applying a top hat transformation, dark areas within a document, such as text and other marks, can be enhanced and made more distinct from the background. This process helps bring out hidden or faded details, such as faded text or subtle handwritten annotations, which are essential to understanding and analyzing historical documents.

The top hat conversion can also be useful for removing unwanted noise or artifacts from document images. It can help reduce the impact of noise or uneven background patterns, resulting in a clearer, clearer representation of the original content.

In general, the top-hat conversion method has a positive effect on historical documents by improving contrast, improving clarity, and revealing hidden details. It helps preserve, analyse, and interpret historical information, making it a valuable tool in the field of document image processing for historical document studies.

### 4.5 Histogram equalization to enhance the contrast of the image

Histogram equalization is a technique used to improve image contrast, which can have a significant impact on historical documents. In historical documents, the contrast between text and background may deteriorate over time due to factors such as aging, fading ink, or uneven lighting during the document capture process.

By applying histogram equalization, the image intensity values are redistributed to cover the full dynamic range. This means that the range of intensity values becomes more evenly distributed, enhancing the differences between the light and dark regions. As a result, text and other details in historical documents become more prominent and easier to read.

Histogram equalization helps reveal hidden details, enhance clarity, and improve the overall visual quality of historical documents. It can compensate for differences in lighting conditions, restore faded or low-contrast text, and make documents easier to analyse and interpret.

By enhancing contrast, histogram equalization contributes to the preservation, digitization, and understanding of historical documents, enabling researchers, historians, and enthusiasts to more effectively study and appreciate these valuable artifacts.

### 4.6 Using the Canny edge detection method

Using the Canny edge method for edge detection has a significant impact on historical documents. Edge detection is a basic image processing technique that aims to identify boundaries between different objects or regions in an image. In the context of historical documents, Canny's edge detection method is particularly valuable for several reasons.

First, historical documents often contain complex patterns, intricate calligraphy, and decorative elements that may have faded or become less distinctive over time. By applying the edge detection method, the edges of these patterns and text elements can be extracted and highlighted, making them cleared and easier to analyse and interpreted.

Second, Canny's edge detection method can help with document segmentation, which involves separating text from the background or differentiate between different regions of interest within a document. This segmentation process is important for many tasks such as text recognition, content analysis, and document understanding. Accurate edge detection through edge detection helps achieve reliable segmentation of documents.

Moreover, the canny edge detection method can be useful in restoring and preserving historical documents. By to set the edges of damaged or deteriorated areas, restoration experts can better understand the extent of the damage and apply appropriate conservation and repair techniques.

Moreover, canny edge detection method can help handwritten text recognition and extraction from historical documents. By detecting the edges of individual characters and words, the Canny Edge edges detection method can contribute to optical character recognition (OCR) systems and other text analysis algorithms, enabling automatic processing and indexing of historical documents.

In short, using the canny edge method on historical documents enhances visual analysis, segmentation, and retrieval, and easy later tasks such as text recognition and content understanding. It plays a critical role in extracting relevant information, maintaining the integrity of the document, and promoting further research and exploration of historical artifacts.

## 4.7 Combined transform [CLAHE (Contrast Limited Adaptive Histogram Equalization) and perform the intensity stretching]

The combined transformation of CLAHE (Adaptive Finite Variation Graph Equation) and density stretching has a significant impact on historical documents. This transformation is intended to enhance contrast and improve the visibility of details in document images. CLAHE is a variant of histogram equalization that adapts to locally improving contrast within small regions of an image. It overcomes the limitations of traditional graph equalization by preventing excessive noise amplification and preserving local contrast. By applying CLAHE, the differences in intensity values within each region of the image are stretched, improving the visibility of fine details and textures.

On the other hand, intensity stretching aims to normalize and scale the intensity range of an image. The minimum and maximum intensity values are extended to cover the full dynamic range, which enhances overall contrast and improves visibility of both dark and bright areas in a document.

When combined, CLAHE and intensity stretching complement each other to achieve a more effective enhancement of historical documents. CLAHE addresses the local contrast variations and enhances the details within small regions, while intensity stretching expands the overall intensity range, enhancing the global contrast of the document.

The impact of this combined transform on historical documents is significant. It can reveal hidden or faint details, enhance the readability of text, and make intricate patterns and illustrations more visible. By improving the contrast and visibility, the combined transform facilitates various tasks such as document analysis, text recognition, content understanding, and preservation.

Furthermore, this transformation can aid in the restoration and digitization of historical documents. By enhancing the contrast and visibility, it helps in the identification and analysis of damaged or faded areas, enabling experts to make informed decisions regarding conservation and repair.

In summary, the combined transform of CLAHE and intensity stretching has a positive impact on historical documents by enhancing contrast, improving visibility, and easy various tasks related to document analysis and preservation. It enables a more detailed examination of historical artifacts and promotes a deeper understanding of their content and context.

## 5 Conclusion

In this chapter, we have provided an overview of the system and preprocessing methods based on the historical document used in this system using the dataset. We also talked in the first part about the classification of historical documents in general and in terms of chronology in particular.

# Chapter 04:

# Results

# Representation

# Introduction

In this chapter, we present the result of the pre-treatment methods used in this system, which we talked about in the third chapter.

We also show a comparison table of the techniques used with recent works.

## 1 Introduction to development tools

### 1.1 Material

The equipment produced is personal PC:

- Processor: Intel® Core i3-3558U @ 1.70GHz   1.70 GHz.
- Memory (RAM): 6.00 GB.
- System type: Windows 10 PRO ,64-bit operating system

### 1.2 History of Python

Python, developed by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the late 1980s[60] , was designed as a successor to the ABC programming language, which was inspired by SETL[61] It's implementation began in December 1989[62].  Van Rossum initially led the project alone as the chief decision-maker until 2018[63], when he retired from his position as Python's "benevolent dictator for life". Following his departure, the active Python core developers elected a five-member Steering Council to lead the project [64] [65].

Python 2.0, with several new features such as cycle-detecting garbage collection, list comprehensions, reference counting, and Unicode support [66], was released on October 16, 2000.

Python 3.0, released on December 3, 2008, backported most of the significant features to Python 2.6.x[67] and 2.7.x, including the 2to3 utility, which automates the translation of Python 2 code to Python 3[68]. Initially scheduled for discontinuation in 2015 but later postponed to 2020 due to concerns about backward compatibility issues, Python 2.7 is no longer being supported[69][70], and no further security patches or other improvements will be released for it[71][72]. Python 3.7 and later are the only versions currently supported.

Python 3.9.2 and 3.8.8 were expedited [73] in 2021 as all versions of Python, including 2.7[74], had security vulnerabilities that could lead to remote code execution [75] and web cache poisoning [76]. Python 3.10.4 and 3.9.12 were also released in 2022 [77] due to numerous security vulnerabilities [78], as were 3.8.13 and 3.7.13[79][80][81].

Python 3.11 is the current stable release as of November 2022, boasting faster program execution and improved error reporting [82].

The Python libraries most used in our approach are:

### 1.2.1 NumPy

NumPy is a foundational package for scientific computing in Python. It is a versatile library that offers a multidimensional array data structure, along with a wide range of operations and functions for efficient numerical computations. With NumPy, you can perform various mathematical, logical, and statistical operations on arrays quickly and easily.

### 1.2.2 Tensorflow

TensorFlow is an open-source library that simplifies numerical computation and deep learning tasks. Its graph-based computation, automatic differentiation, neural network building blocks, GPU acceleration, and deployment capabilities make it a powerful tool for developing and deploying machine learning models. The wide-ranging features and strong community support contribute to TensorFlow's popularity in the field of artificial intelligence and deep learning.

### 1.2.3 Matplotlib

Matplotlib is a Python library used for creating visualizations and plots. It provides a wide range of functions and tools for generating various types of plots, charts, histograms, and other visual representations of data.

To define and use Matplotlib in Python, you need to import the library and its relevant modules.

### 1.2.4 Torch

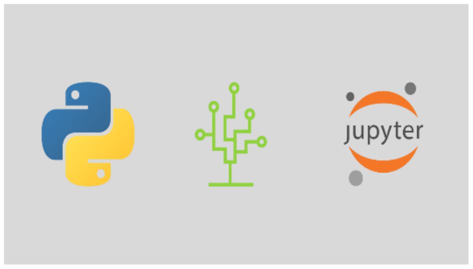
Torch, also known as PyTorch, is an open-source machine learning library widely used for various tasks such as deep learning, natural language processing, and computer vision. It provides a flexible and efficient framework for building and training neural networks.

## 1.2 Anaconda

Anaconda is a popular open-source distribution of the Python and R programming languages, used for data science, machine learning, and scientific computing. It provides a comprehensive and user-friendly platform for managing software dependencies, package installation, and project environments, as well as integrated development environments (IDEs) and other tools for data analysis and visualization[83][84].

## 1.3 Jupyter

Jupyter is a web-based interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports a variety of programming languages, including Python, R, and Julia, and is widely used for data analysis, scientific computing, machine learning, and education [85].

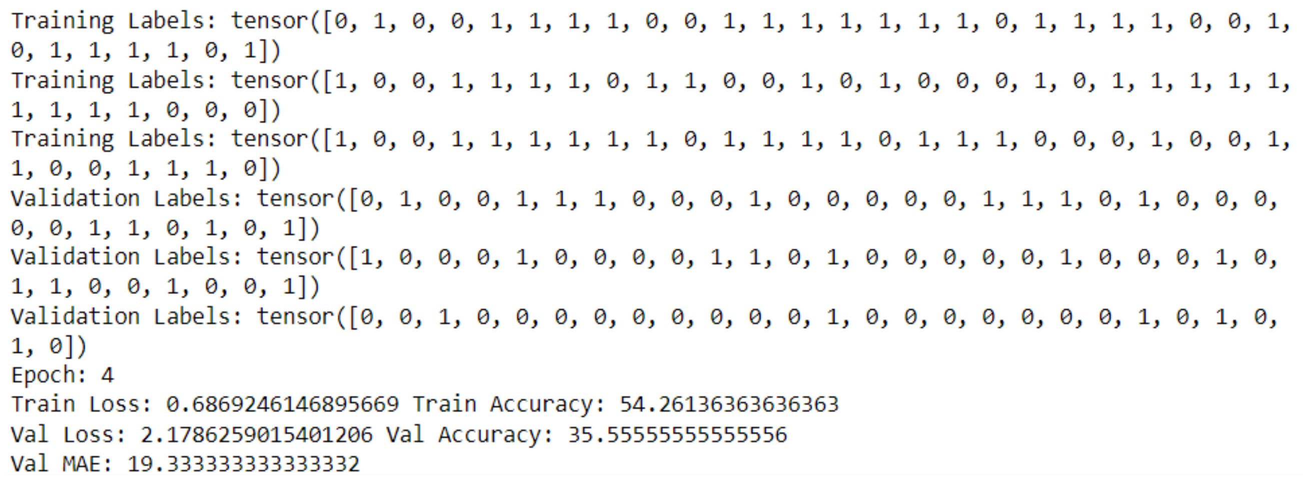


Jupyter notebooks, the primary interface of the Jupyter environment, consist of a series of cells that can contain code, text, equations, and multimedia elements. Users can execute code cells in real-time and see the output immediately, allowing for rapid prototyping and experimentation [86].

Jupyter was originally created as part of the IPython project in 2014, but has since become a standalone open-source project with a large and active community of contributors [87].Top of Form

## 2 Results without Preprocessing

Pre-processing methods have a significant impact on historical documents. Without applying them, we notice that they are not clear and not very readable, as shown in the picture.



```
Training Labels: tensor([0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1,
0, 1, 1, 1, 1, 0, 1])
Training Labels: tensor([1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0])
Training Labels: tensor([1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
1, 0, 0, 1, 1, 1, 0])
Validation Labels: tensor([0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
0, 0, 1, 1, 0, 1, 0, 1])
Validation Labels: tensor([1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
1, 1, 0, 0, 1, 0, 0, 1])
Validation Labels: tensor([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
1, 0])
Epoch: 4
Train Loss: 0.6869246146895669 Train Accuracy: 54.26136363636363
Val Loss: 2.1786259015401206 Val Accuracy: 35.55555555555556
Val MAE: 19.333333333333332
```

Fig14 loss accuracy without preprocessing

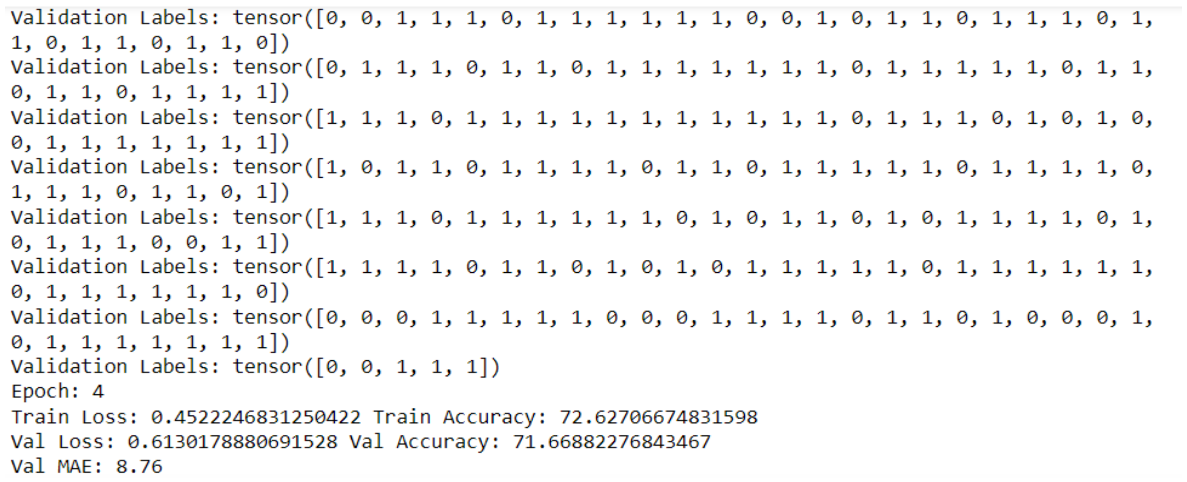
## 3 Results with Preprocessing

### 3.1 Binarization and remove noise

The code combines image preprocessing steps such as binarization, noise removal, and deskewing of an image.

- The **transform** variable defines a set of transformations using the **tt.Compose** function. These transformations include resizing the image to (224, 224) pixels, converting it to a PyTorch tensor, and normalizing the pixel values.
- The **deskew** function takes an image path as input and performs the following steps:
    - Loads the image using OpenCV.
    - Converts the image to grayscale.
    - Binarizes the grayscale image using Otsu's thresholding.
    - Removes noise from the binary image using median blur.
    - Estimates the skew angle of the image using Hough lines detection.
    - Applies a rotation transformation to deskew the image based on the estimated angle.
    - Converts the rotated image back to a PIL image.
- After defining the **deskew** function, the **transform_3** variable is assigned a new set of transformations using **tt.Compose**. These transformations include:
    - Applying the **deskew** function using **tt.Lambda**.

- Resizing the image to (224, 224) pixels.
- Applying random horizontal and vertical flips for data augmentation.
- Applying random rotations up to 30 degrees.
- Converting the image to a PyTorch tensor.
- Normalizing the pixel values using mean and standard deviation values.



```
Validation Labels: tensor([0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 0])
Validation Labels: tensor([0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1])
Validation Labels: tensor([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,
0, 1, 1, 1, 1, 1, 1, 1])
Validation Labels: tensor([1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 1, 1, 0, 1])
Validation Labels: tensor([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
0, 1, 1, 1, 0, 0, 1, 1])
Validation Labels: tensor([1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 0])
Validation Labels: tensor([0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1])
Validation Labels: tensor([0, 0, 1, 1, 1])
Epoch: 4
Train Loss: 0.4522246831250422 Train Accuracy: 72.62706674831598
Val Loss: 0.6130178880691528 Val Accuracy: 71.66882276843467
Val MAE: 8.76
```

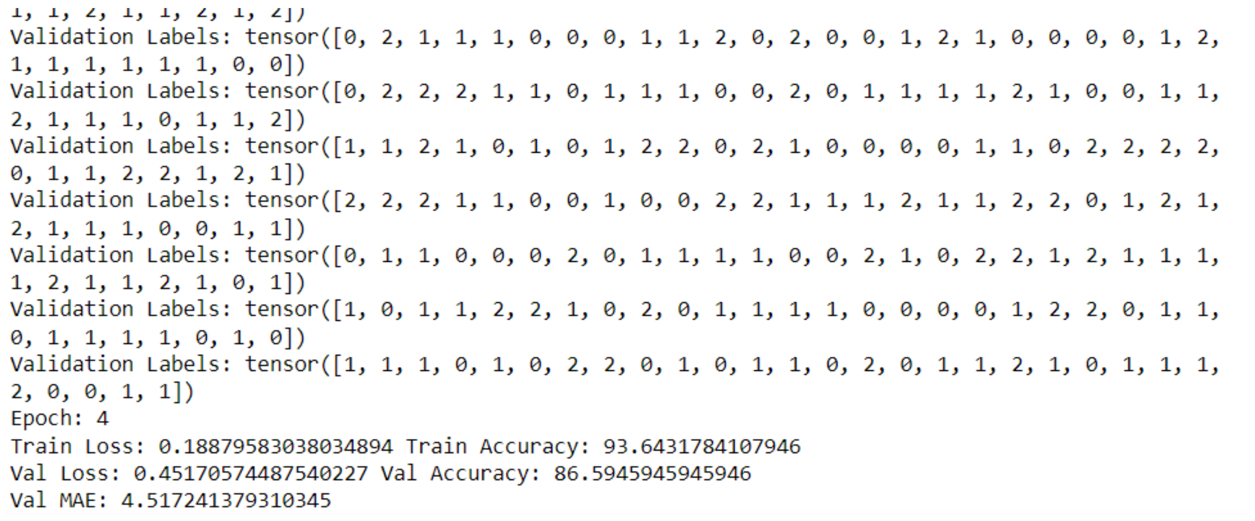
Fig 15 loss Accuracy

## 3.2 Deskew function (remove noise and rotate)

This code defines a **deskew** function that removes noise and rotates an image. Simplified explanation:

- The **transform** variable defines a set of transformations using the **tt.Compose** function. These transformations include resizing the image to (224, 224) pixels, converting it to a PyTorch tensor, and normalizing the pixel values.
- The **deskew** function takes an image path as input and performs the following steps:
  - Loads the image using OpenCV.
  - Removes noise from the image using median blur.
  - Calculates the image's dimensions and center point.
  - Generates a rotation matrix using **cv2.getRotationMatrix2D**.
  - Applies the rotation to the image using **cv2.warpAffine**.
  - Converts the rotated image back to a PIL image.
- After defining the **deskew** function, the **transform_3** variable is assigned a new set of transformations using **tt.Compose**. These transformations include:
  - Applying the **deskew** function using **tt.Lambda**.

- Resizing the image to (224, 224) pixels.
- Applying random horizontal and vertical flips for data augmentation.
- Applying random rotations up to 30 degrees.
- Converting the image to a PyTorch tensor.
- Normalizing the pixel values using mean and standard deviation values.



```
1, 1, 2, 1, 1, 2, 1, 2])
Validation Labels: tensor([0, 2, 1, 1, 1, 0, 0, 0, 1, 1, 2, 0, 2, 0, 0, 1, 2, 1, 0, 0, 0, 0, 1, 2,
1, 1, 1, 1, 1, 1, 0, 0])
Validation Labels: tensor([0, 2, 2, 2, 1, 1, 0, 1, 1, 1, 0, 0, 2, 0, 1, 1, 1, 1, 2, 1, 0, 0, 1, 1,
2, 1, 1, 1, 0, 1, 1, 2])
Validation Labels: tensor([1, 1, 2, 1, 0, 1, 0, 1, 2, 2, 0, 2, 1, 0, 0, 0, 0, 1, 1, 0, 2, 2, 2, 2,
0, 1, 1, 2, 2, 1, 2, 1])
Validation Labels: tensor([2, 2, 2, 1, 1, 0, 0, 1, 0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2, 0, 1, 2, 1,
2, 1, 1, 1, 0, 0, 1, 1])
Validation Labels: tensor([0, 1, 1, 0, 0, 0, 2, 0, 1, 1, 1, 1, 0, 0, 2, 1, 0, 2, 2, 1, 2, 1, 1, 1,
1, 2, 1, 1, 2, 1, 0, 1])
Validation Labels: tensor([1, 0, 1, 1, 2, 2, 1, 0, 2, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 2, 2, 0, 1, 1,
0, 1, 1, 1, 1, 0, 1, 0])
Validation Labels: tensor([1, 1, 1, 0, 1, 0, 2, 2, 0, 1, 0, 1, 1, 0, 2, 0, 1, 1, 2, 1, 0, 1, 1, 1,
2, 0, 0, 1, 1])
Epoch: 4
Train Loss: 0.18879583038034894 Train Accuracy: 93.6431784107946
Val Loss: 0.45170574487540227 Val Accuracy: 86.5945945945946
Val MAE: 4.517241379310345
```

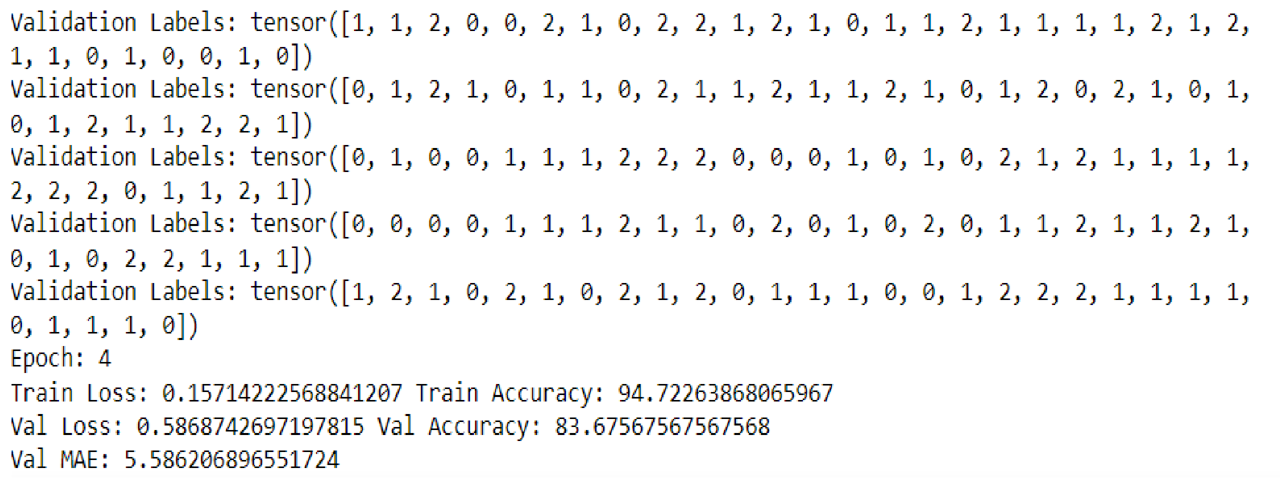
Fig 16 loss Accuracy

### 3.3 Each image in the dataset and perform the intensity stretching

This code performs intensity stretching on each image in the dataset. Brief explanation:

- The **deskew** function takes an image path as input and performs the following steps:
    - Loads the image using OpenCV.
    - Removes noise from the image using median blur.
    - Rotates the image to deskew it.
    - Converts the rotated image to grayscale.
    - Calculates the minimum and maximum intensity values of the grayscale image.
    - Stretches the intensity values of the image to cover the full range.
    - Converts the stretched image back to a PIL image.
- The **transform** variable defines a set of transformations using the **tt.Compose** function. These transformations include resizing the image to (224, 224) pixels, converting it to a PyTorch tensor, and normalizing the pixel values.
- The **transform_3** variable is assigned a new set of transformations using **tt.Compose**. These transformations include:
    - Applying the **deskew** function using **tt.Lambda**.

- Resizing the image to (224, 224) pixels.
- Applying random horizontal and vertical flips for data augmentation.
- Applying random rotations up to 30 degrees.
- Converting the image to a PyTorch tensor.
- Normalizing the pixel values using mean and standard deviation values.



```
Validation Labels: tensor([1, 1, 2, 0, 0, 2, 1, 0, 2, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2,
1, 1, 0, 1, 0, 0, 1, 0])
Validation Labels: tensor([0, 1, 2, 1, 0, 1, 1, 0, 2, 1, 1, 2, 1, 1, 2, 1, 0, 1, 2, 0, 2, 1, 0, 1,
0, 1, 2, 1, 1, 2, 2, 1])
Validation Labels: tensor([0, 1, 0, 0, 1, 1, 1, 2, 2, 2, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 1, 1, 1, 1,
2, 2, 2, 0, 1, 1, 2, 1])
Validation Labels: tensor([0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 2, 0, 1, 0, 2, 0, 1, 1, 2, 1, 1, 2, 1,
0, 1, 0, 2, 2, 1, 1, 1])
Validation Labels: tensor([1, 2, 1, 0, 2, 1, 0, 2, 1, 2, 0, 1, 1, 1, 0, 0, 1, 2, 2, 2, 1, 1, 1, 1,
0, 1, 1, 1, 0])
Epoch: 4
Train Loss: 0.15714222568841207 Train Accuracy: 94.72263868065967
Val Loss: 0.5868742697197815 Val Accuracy: 83.67567567567568
Val MAE: 5.586206896551724
```

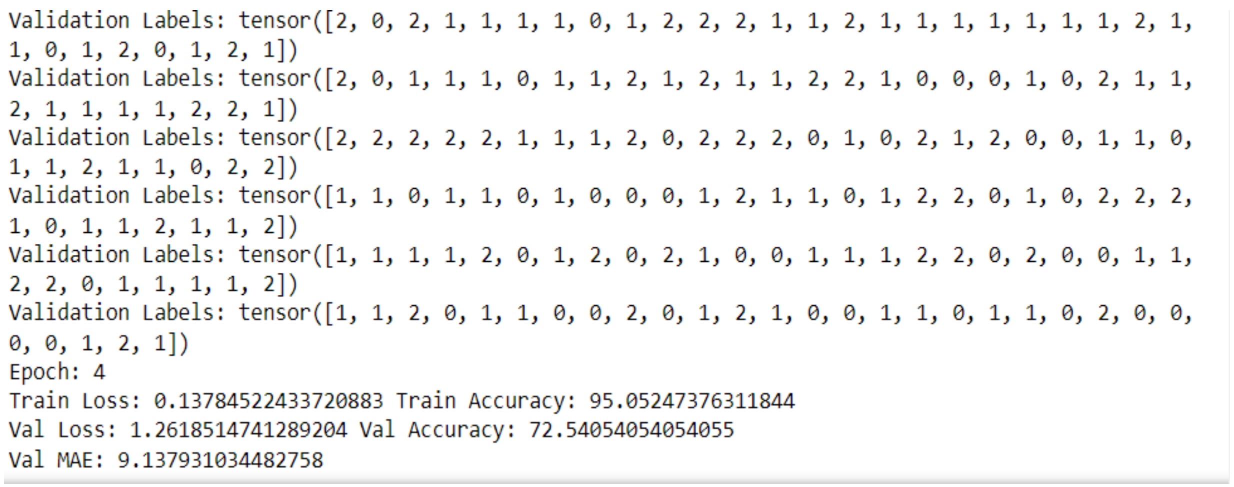
Fig 17 loss Accuracy

## 3.4 Method top-hat transform

This method is used to enhance the contrast of text and other dark regions in an image

- The **deskew** function takes an image path as input and performs the following steps:
  - Loads the image using OpenCV.
  - Converts the image to grayscale.
  - Defines a structuring element for the morphological operation using **cv2.getStructuringElement** .
  - Applies the top-hat transform using **cv2.morphologyEx** with the **cv2.MORPH_TOPHAT** operation.
  - Normalizes the output image using **cv2.normalize**.
  - Stretches the intensity values of the image to cover the full range using **cv2.normalize**.
  - Converts the resulting image back to a PIL image.
- The **transform** variable defines a set of transformations using the **tt.Compose** function. These transformations include resizing the image to (224, 224) pixels, converting it to a PyTorch tensor, and normalizing the pixel values.
- The **transform_3** variable is assigned a new set of transformations using **tt.Compose**. These transformations include:

- Applying the **deskew** function using **tt.Lambda**.
- Resizing the image to (224, 224) pixels.
- Applying random horizontal and vertical flips for data augmentation.
- Applying random rotations up to 30 degrees.
- Converting the image to a PyTorch tensor.
- Normalizing the pixel values using mean and standard deviation values.



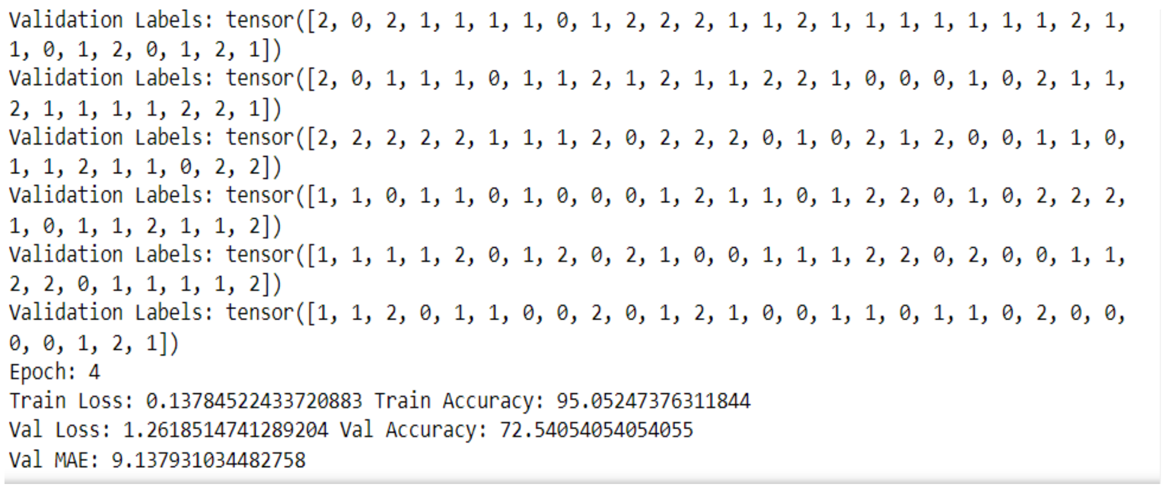
```
Validation Labels: tensor([2, 0, 2, 1, 1, 1, 1, 0, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
1, 0, 1, 2, 0, 1, 2, 1])
Validation Labels: tensor([2, 0, 1, 1, 1, 0, 1, 1, 2, 1, 2, 1, 1, 2, 2, 1, 0, 0, 0, 1, 0, 2, 1, 1,
2, 1, 1, 1, 1, 2, 2, 1])
Validation Labels: tensor([2, 2, 2, 2, 2, 1, 1, 1, 2, 0, 2, 2, 2, 0, 1, 0, 2, 1, 2, 0, 0, 1, 1, 0,
1, 1, 2, 1, 1, 0, 2, 2])
Validation Labels: tensor([1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 2, 1, 1, 0, 1, 2, 2, 0, 1, 0, 2, 2, 2,
1, 0, 1, 1, 2, 1, 1, 2])
Validation Labels: tensor([1, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 0, 0, 1, 1, 1, 2, 2, 0, 2, 0, 0, 1, 1,
2, 2, 0, 1, 1, 1, 1, 2])
Validation Labels: tensor([1, 1, 2, 0, 1, 1, 0, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 1, 0, 2, 0, 0,
0, 0, 1, 2, 1])
Epoch: 4
Train Loss: 0.13784522433720883 Train Accuracy: 95.05247376311844
Val Loss: 1.2618514741289204 Val Accuracy: 72.54054054054055
Val MAE: 9.137931034482758
```

Fig 18 loss Accuracy

## 3.5 Histogram equalization to enhance the contrast of the image

This code uses histogram equalization to enhance the contrast of the image:

In this code, we use the **cv2.equalizeHist** function to perform histogram equalization on the grayscale image. This method is another common technique used to enhance the contrast of an image.



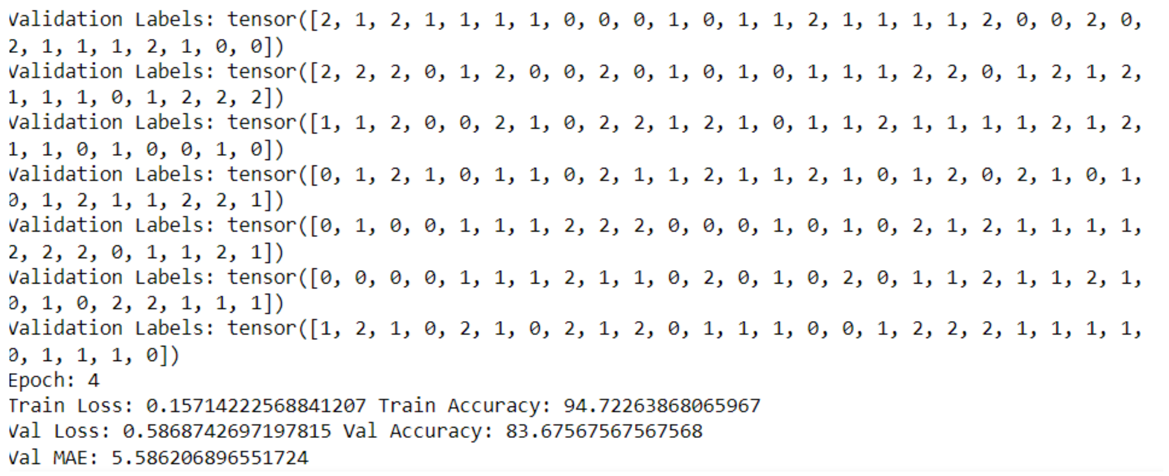
```
Validation Labels: tensor([2, 0, 2, 1, 1, 1, 1, 0, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
1, 0, 1, 2, 0, 1, 2, 1])
Validation Labels: tensor([2, 0, 1, 1, 1, 0, 1, 1, 2, 1, 2, 1, 1, 2, 2, 1, 0, 0, 0, 1, 0, 2, 1, 1,
2, 1, 1, 1, 1, 2, 2, 1])
Validation Labels: tensor([2, 2, 2, 2, 2, 1, 1, 1, 2, 0, 2, 2, 2, 0, 1, 0, 2, 1, 2, 0, 0, 1, 1, 0,
1, 1, 2, 1, 1, 0, 2, 2])
Validation Labels: tensor([1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 2, 1, 1, 0, 1, 2, 2, 0, 1, 0, 2, 2, 2,
1, 0, 1, 1, 2, 1, 1, 2])
Validation Labels: tensor([1, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 0, 0, 1, 1, 1, 2, 2, 0, 2, 0, 0, 1, 1,
2, 2, 0, 1, 1, 1, 1, 2])
Validation Labels: tensor([1, 1, 2, 0, 1, 1, 0, 0, 2, 0, 1, 2, 1, 0, 0, 1, 1, 0, 1, 1, 0, 2, 0, 0,
0, 0, 1, 2, 1])
Epoch: 4
Train Loss: 0.13784522433720883 Train Accuracy: 95.05247376311844
Val Loss: 1.2618514741289204 Val Accuracy: 72.54054054054055
Val MAE: 9.137931034482758
```

Fig 19 loss Accuracy

## 3.6 Using the Canny edge detection method

The Canny edge detection method:

- The Canny edge detection method to the input image, which is a technique for detecting edges in an image by applying a series of mathematical operations to it. The resulting image highlights the edges in white and the rest of the image in black. The output image is then resized to 224x224 and converted to a PyTorch tensor with normalization applied, so it can be used as input to a neural network.
- The **canny_edge_detection** function takes an image path as input and applies the Canny edge detection method to detect the edges in the image. The resulting edges are then converted back to a PIL image.
- The **transform canny** uses the **canny_edge_detection** function in the **tt.Lambda** transform to apply the Canny edge detection to each image in the dataset. It also includes additional transformations such as resizing, random horizontal and vertical flips, random rotation, and normalization to prepare the images for further processing or model training.

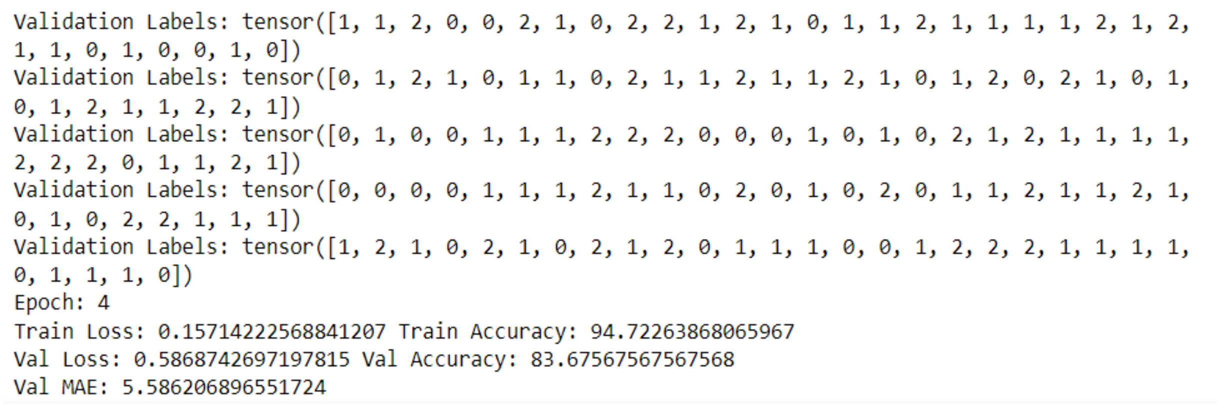


```
Validation Labels: tensor([2, 1, 2, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 2, 1, 1, 1, 1, 2, 0, 0, 2, 0,
2, 1, 1, 1, 2, 1, 0, 0])
Validation Labels: tensor([2, 2, 2, 0, 1, 2, 0, 0, 2, 0, 1, 0, 1, 0, 1, 1, 1, 2, 2, 0, 1, 2, 1, 2,
1, 1, 1, 0, 1, 2, 2, 2])
Validation Labels: tensor([1, 1, 2, 0, 0, 2, 1, 0, 2, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2,
1, 1, 0, 1, 0, 0, 1, 0])
Validation Labels: tensor([0, 1, 2, 1, 0, 1, 1, 0, 2, 1, 1, 2, 1, 1, 2, 1, 0, 1, 2, 0, 2, 1, 0, 1,
0, 1, 2, 1, 1, 2, 2, 1])
Validation Labels: tensor([0, 1, 0, 0, 1, 1, 1, 2, 2, 2, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 1, 1, 1, 1,
2, 2, 2, 0, 1, 1, 2, 1])
Validation Labels: tensor([0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 2, 0, 1, 0, 2, 0, 1, 1, 2, 1, 1, 2, 1,
0, 1, 0, 2, 2, 1, 1, 1])
Validation Labels: tensor([1, 2, 1, 0, 2, 1, 0, 2, 1, 2, 0, 1, 1, 1, 0, 0, 1, 2, 2, 2, 1, 1, 1, 1,
0, 1, 1, 1, 0])
Epoch: 4
Train Loss: 0.15714222568841207 Train Accuracy: 94.72263868065967
Val Loss: 0.5868742697197815 Val Accuracy: 83.67567567567568
Val MAE: 5.586206896551724
```

Fig 20 loss Accuracy

## 3.7 Combined transform [CLAHE (Contrast Limited Adaptive Histogram Equalization) and perform the intensity stretching]

'deskew_and_stretch': function combined the deskew function and the intensity stretching code to perform both operations on the image. The combined transform uses this deskew_and_stretch function in the tt.Lambda transform to apply the combined operation to each image in the dataset.



```
Validation Labels: tensor([1, 1, 2, 0, 0, 2, 1, 0, 2, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2,
1, 1, 0, 1, 0, 0, 1, 0])
Validation Labels: tensor([0, 1, 2, 1, 0, 1, 1, 0, 2, 1, 1, 2, 1, 1, 2, 1, 0, 1, 2, 0, 2, 1, 0, 1,
0, 1, 2, 1, 1, 2, 2, 1])
Validation Labels: tensor([0, 1, 0, 0, 1, 1, 1, 2, 2, 2, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 1, 1, 1, 1,
2, 2, 2, 0, 1, 1, 2, 1])
Validation Labels: tensor([0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 0, 2, 0, 1, 0, 2, 0, 1, 1, 2, 1, 1, 2, 1,
0, 1, 0, 2, 2, 1, 1, 1])
Validation Labels: tensor([1, 2, 1, 0, 2, 1, 0, 2, 1, 2, 0, 1, 1, 1, 0, 0, 1, 2, 2, 2, 1, 1, 1, 1,
0, 1, 1, 1, 0])
Epoch: 4
Train Loss: 0.15714222568841207 Train Accuracy: 94.72263868065967
Val Loss: 0.5868742697197815 Val Accuracy: 83.67567567567568
Val MAE: 5.586206896551724
```

Fig 21 loss Accuracy

## 4 Table of the results

**Table 4** presents a summary of preprocessing techniques on historical documents. This can be seen the different in the effect of the results on them in general through the table shown below.

| **Preprocessing methods used** | **Model** | **Dataset** | **Accuracy %** |
|---|---|---|---|
| 1 binarization and remove noise | CNN | MPS | Train 72.66% Valid 71.66% |
| 2 deskew function (remove noise and rotate) | | | Train 93.64% Valid 86.59% |
| 3 deskew function to each image in the dataset and perform the intensity stretching(this method after patch) | | | Train 94.72% Valid 83.67% |
| 4 method top-hat transform This method is used to enhance the contrast of text and other dark regions in an image | | | Train 95.05% Valid 72.54% |
| 5 histogram equalization to enhance the contrast of the image (after patches) | | | Train 95.05% Valid 72.54% |
| 6 the Canny edge detection method (after patches) | | | Train 94.72% Valid 83.67% |
| 7 combined transform[ CLAHE (Contrast Limited Adaptive Histogram Equalization)and perform the intensity stretching] | | | Train 94.72% Valid 83.67% |

**Table 4** table of the results

## 5 Comparison with the recent work in the same Dataset

**Table 5** provides a summary of the comparison of recent work with the used preprocessing methods. This is done with the use of the same data set, as can be seen in the table shown below.

| **Preprocessing methods used** | **Model** | **Dataset** | **Accuracy %** |
|---|---|---|---|
| 1 Methodology to automatically date the historical manuscripts. | ConvNet | MPS | 91.71% |
| 2 binarization and remove noise | CNN | MPS | Train 72.66%<br>Valid 71.66% |
| 3 deskew function (remove noise and rotate) | | | Train 93.64%<br>Valid 86.59% |
| 4 deskew function to each image in the dataset and perform the intensity stretching(this method after patch) | | | Train 94.72%<br>Valid 83.67% |
| 5 method top-hat transform This method is used to enhance the contrast of text and other dark regions in an image | | | Train 95.05%<br>Valid 72.54% |
| 6 histogram equalization to enhance the contrast of the image (after patches) | | | Train 95.05%<br>Valid 72.54% |
| 7 the Canny edge detection method (after patches) | | | Train 94.72%<br>Valid 83.67% |
| 8 combined transform[ CLAHE (Contrast Limited Adaptive Histogram Equalization)and perform the intensity stretching] | | | Train 94.72%<br>Valid 83.67% |

**Table 5** Comparison with recent work in same dataset

## 6 Comparison with recent works

**Table 6** provides a summary of the comparison of recent work with the preprocessing methods used in this system. This can be seen in the table shown below.

| **Preprocessing methods used** | **Article** | **Accuracy %** |
|---|---|---|
| 1. CNNs, have been applied for feature extraction and classification of medieval manuscripts. | [47] | 83.8% |
| 2. convolutional neural network (CNN) to predict the regions of interest in the document image, such as text regions, image regions, and background regions | [48] | 89% with (G.Washington dataset),<br>64% with ( Parzival dataset)<br>89% with ( St. Gall dataset) |
| 3. method siamese network architecture, which is a type of neural network Designed for learning similarity or dissimilarity between input pairs | [52] | 93% |
| 4. binarization and remove noise | 71.66% | |
| 5.deskew function (remove noise and rotate) | 86.59% | |
| 6.methode deskew function to each image in the dataset and perform the intensity stretching(this method after patch) | 83.67% | |
| 7 method top-hat transform This method is used to enhance the contrast of text and other dark regions in an image | 81.40% | |
| 8 histogram equalization to enhance the contrast of the image (after patches) | 72.54% | |
| 9 the Canny edge detection method (after patches) | 83.67% | |
| 10 combined transform[ CLAHE (Contrast Limited Adaptive Histogram Equalization)and perform the intensity stretching] | 83.67% | |

**Table 6** Comparison with recent works

We note the difference in the results of the techniques used in this system with the recent works, as we find there are differences in different proportions shown in the above table in the impact of these technologies on the system used. Where we also noticed that when applying preprocessing methods, the result is better, unlike not using them, which we talked about earlier.

## 7 Conclusion

In this part, we presented the results of the preprocessing methods used, and we have also compared the results with the recent work in the table.

# General Conclusion

The field of pre-processing methods on historical documents is an area of research that has been explored for several decades and is considered a work of art. There is an improvement in the results, but so far there is no perfect solution in its processing methods which achieve 100% recognition rate than a generic processor. Therefore, research into it remains open. The aim of this work is to achieve a system that works on pre-treatment methods on historical documents.

During this research, we went through some of its stages, including the extraction of features, such as the history of writing. In order to validate our system, a pilot study using the MPS database is performed, this study aimed at the problem of complete clarity of historical documents. It examined the effectiveness and features of the image by using pre-processing methods on historical documents.

The obtained results are encouraging and validate the effectiveness of pre-treatment methods in the characterization of historical documents. It is also possible to improve in these methods by the type of image or the combination of some pre-processing methods that provide good results.

## References:

[1] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in NIPS workshop on deep learning and unsupervised feature learning, no. 2, 2011, p. 5.

[2] S. S. Bukhari, T. M. Breuel, A. Asi, and J. El-Sana, “Layout analysis for arabic historical document images using machine learning,” in Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on. IEEE, 2012, pp. 639–644.

[3] A. Asi, R. Cohen, K. Kedem, J. El-Sana, and I. Dinstein, “A coarse-tofine approach for layout analysis of ancient manuscripts,” in Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. IEEE, 2014, pp. 140–145.

[4] B. Kurar and J. El-Sana, “Binarization free layout analysis for arabic historical documents using fully convolutional networks,” in 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR). IEEE, 2018, pp. 151–155.

[5] S. Eskenazi, P. Gomez-Kramer, and J.-M. Ogier, “A comprehensive ¨ survey of mostly textual document segmentation algorithms since 2008,” Pattern Recognition, vol. 64, pp. 1–14, 2017.

[6]Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2008). Digital Image Processing using MATLAB. Prentice Hall.

[7]Pratt, W. K. (2007). Digital Image Processing. Wiley.

[8]Russ, J. C. (2011). The Image Processing Handbook. CRC Press.

[9]Gonzalez, R.C., Woods, R.E., & Eddins, S.L. (2004). Processing digital images using MATLAB. Pearson education.

[10]Pratt, W.; (2007). digital image processing. John Wiley and Sons.

[11]Sonka, M., Hlavac, V., & Boyle, R. (2014). Image processing, analysis, and machine vision. Cengage Learning.

[12]Schneider, P. (2019). Using historical documents. International Encyclopedia of Human Geography, 2nd ed. doi: 10.1016/B978-0-08-102295-5.10341-9

[13]Lester, J. (2019). Primary sources in historical research. Oxford Research Encyclopedia of American History. doi: 10.1093/acrefore/9780199329175.013.290

[14]The National Archives. (n.d.). What are historical documents? Retrieved from https://www.nationalarchives.gov.uk/education/resources/what-are-historical-documents/

[15]NEDCC. (n.d.). Preservation leaflets. Northeast Document Conservation Center. Retrieved from https://www.nedcc.org/free-resources/preservation-leaflets/overview

[16]Library of Congress. (2019). Digitization. Library of Congress. Retrieved from https://www.loc.gov/preservation/digitization/

[17]HathiTrust Research Center. (n.d.). Transcription. HathiTrust Research Center. Retrieved from https://analytics.hathitrust.org/transcription.html

[18]National Archives. (n.d.). Translating historical documents. National Archives. Retrieved from https://www.archives.gov/research/alic/reference/translation.html

[19]Digital Humanities at Berkeley. (n.d.). Data extraction. Digital Humanities at Berkeley. Retrieved from https://digitalhumanities.berkeley.edu/resources/techniques/data-extraction

[20]Mullen, L. (2013). Normalization of historical text. Literary and Linguistic Computing, 28(1), 29-39. doi: 10.1093/llc/fqs024

[21]Kedem, G., et al. (2019). Towards restoration of ancient texts: A novel approach to color reduction of degraded parchment. Signal Processing: Image Communication, 76, 110-118.

[22]Bergsma, S., et al. (2019). Handwritten text recognition of 17th-century Dutch church records using machine learning. Journal of Cultural Heritage, 39, 247-252.

[23]Schulz, K., et al. (2020). Automatic extraction of personal names from historical German documents. Digital Scholarship in the Humanities, 35(4), 728-746.

[24]Piotrowski, M., et al. (2019). A hybrid machine learning and rule-based approach for spelling normalization of historical texts. Digital Scholarship in the Humanities, 34(2), 350-364.

[25]Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed.). Prentice Hall.

[26]Koller, D., & Friedman, N. (2009). Probabilistic graphical models: principles and techniques. MIT Press.

[27]Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books.

[28]Russell, S. J., Norvig, P., Davis, E., & Genesereth, M. (2010). Artificial intelligence: a modern approach. Pearson Education.

[29]Nilsson, N. J. (1998). Artificial intelligence: A new synthesis. Morgan Kaufmann.

[30]McCorduck, P. (2004). Machines who think: A personal inquiry into the history and prospects of artificial intelligence (2nd ed.). AK Peters.

[31]Mitchell, T. (1997). Machine learning. McGraw Hill.

[32]F. David, "Types of Machine Learning Algorithms You Should Know | by David Fumo | Towards Data Science," Jun. 15, 2017. https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861 (accessed May 16, 2022).

[33]LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[34]Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.

[35]Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[36]Sahoo, P.; Soltani, S. & Wong, A. (1988). A Survey of Thresholding Techniques, Computer Vision Graphics Image Processing, Vol. 41, pp. 233-260.

[37]Otsu, N. (1979). A Threshold Selection Method From Grey-level Histograms, IEEE Transactions On systems, Man and Cybernetics, SMC-9, pp. 62-66.

[38]Rosin, P. & Ioannidis, E. (2003). Evaluation of Global Image Thresholding for Change Detection, Pattern Recognition Letters, Vol. 24, pp. 2345-2356.

[39]Trier, O. & Jain, A. (1995). Goal-Directed Evaluation of Binarization Methods, IEEE Trans.On Pattern Recognition and Machine Intelligence, Vol. 17, No. 12, pp. 1191-1201.

[40]Fischer, S., (2000). Digital Image Processing: Skewing and Thresholding, Master of Science thesis, University of New South Wales, Sydney, Australia.

[41]Russ, J. (2007). *The Image Processing Handbook,* Fifth Edition, CRC Press, Boca Raton, FL, USA.

[42]Horn, B.K.P. & R.J. Woodham (1979). Destriping LANDSAT MSS Images using Histogram Modification, *Computer Graphics and Image Processing*, Vol. 10, No. 1, May 1979, pp. 69–83.

[43]Niblack, W. (1986). An Introduction to Digital Image Processing, Prentice Hall, ISBN-10 0134806743, ISBN-13 : 978-0134806747, Englewood Cliffs, NJ. USA, pp. 115-116.

[44]O'Gorman, L.; Sammon, M. & Seul, M. (2008). Practical Algorithms For Image Analysis, Cambridge University Press, ISBN 978-0=521-88411-2, New York, NY, USA.

[45]Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic Data for Text Localisation in Natural Images. arXiv preprint arXiv:1604.06646.

[46]Tensmeyer, C., Luo, Y., Zhang, Y., & Li, X. (2020). Towards Robust Page Segmentation of Historical Documents via Adversarial Training. arXiv preprint arXiv:2005.07668.

[47] Apostolos Antonacopoulos, Basilis Gatos, and al ,Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2016

[48] Seuret, M., and al. "Convolutional Neural Networks for Page Segmentation of Historical Document Images." Document Analysis and Recognition (ICDAR), 2017 14th International Conference on. IEEE, 2017.

[49] G.Renton and al. " Handwritten Text Line Segmentation Using Fully Convolutional Neural Networks". Document Analysis and Recognition (ICDAR), 2017 International Conference on. IEEE, 2017.

[50] Fischer, A., and al. "A Fully Convolutional Neural Network for Text Line Segmentation in Historical Handwritten Documents." Document Analysis and Recognition (ICDAR), 2018 15th International Conference on. IEEE, 2018.

[51] B.barakat and al" Text Line Segmentation for Challenging Handwritten Document Images using Fully Convolutional Network " document analysis and recognition 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)2018

[52] R.Alaasam and al , "Layout Analysis on Challenging Historical Arabic Manuscripts using Siamese Network", international conference icdar 2019

[53] Singh, A., and al, M. "Handwritten Text Recognition Using Deep Learning Techniques: A Review." IEEE Transactions on Human-Machine Systems, vol. 49, no. 4, pp. 382-392, 2019.

[54] Gatos, G., and al. "A Deep Learning Framework for Handwritten Text Recognition on Medieval Manuscripts." Document Analysis and Recognition (ICDAR), 2020 16th International Conference on. IEEE, 2020.

[55] Suresh, S. S., and Sundaram, S. "Handwritten Text Recognition using Deep Learning: A Survey." Expert Systems with Applications, vol. 141, article ID 112990, 2020.

[56] Monwar, L. G., and al. "Handwritten Text Recognition on Historical Documents: A Survey." Pattern Recognition Letters, vol. 136, pp. 52-60, 2020.

[57]McGillivray, B. (2019). Deep Learning and the Humanities: Methods and Applications. Journal of Cultural Analytics, 2(1), 1-22.

[58]Riddell, A., Wiggins, G.A, & Stairmand, M. (2018). Computational techniques for instrumental analysis of medieval Latin texts. Journal of Cultural Analytics, 3(1), 1-21.

[59]Sheng He, Petros Sammara, Jan Burgers, and Lambert Schomaker. Towards style-based dating of historical documents. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 265–270. IEEE, 2014.

[60] Venners, Bill (13 January 2003). "The Making of Python". Artima Developer. Artima. Archived from the original on 1 September 2016. Retrieved 22 March 2007.

[61] van Rossum, Guido (29 August 2000). "SETL (was: Lukewarm about range literals)". Python-Dev (Mailing list). Archived from the original on 14 July 2018. Retrieved 13 March 2011.

[62] van Rossum, Guido (20 January 2009). "A Brief Timeline of Python". The History of Python. Archived from the original on 5 June 2020. Retrieved 20 January 2009.

[63] Fairchild, Carlie (12 July 2018). "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life". Linux Journal. Archived from the original on 13 July 2018. Retrieved 13 July 2018.

[64]"PEP 8100". Python Software Foundation. Archived from the original on 4 June 2020. Retrieved 4 May 2019.

[65] "PEP 13 – Python Language Governance". Python.org. Archived from the original on 27 May 2021. Retrieved 25 August 2021.

[66]Kuchling, A. M.; Zadka, Moshe (16 October 2000). "What's New in Python 2.0". Python Software Foundation. Archived from the original on 23 October 2012. Retrieved 11 February 2012.

[67]van Rossum, Guido (5 April 2006). "PEP 3000 – Python 3000". Python Enhancement Proposals. Python Software Foundation. Archived from the original on 3 March 2016. Retrieved 27 June 2009.

[68]"2to3 – Automated Python 2 to 3 code translation". docs.python.org. Archived from the original on 4 June 2020. Retrieved 2 February 2021.

[69]"PEP 373 – Python 2.7 Release Schedule". python.org. Archived from the original on 19 May 2020. Retrieved 9 January 2017.

[70]"PEP 466 – Network Security Enhancements for Python 2.7.x". python.org. Archived from the original on 4 June 2020. Retrieved 9 January 2017.

[71] "Sunsetting Python 2". Python.org. Archived from the original on 12 January 2020. Retrieved 22 September 2019.

[72]"PEP 373 – Python 2.7 Release Schedule". Python.org. Archived from the original on 13 January 2020. Retrieved 22 September 2019.

[73] Langa, Łukasz (19 February 2021). "Python Insider: Python 3.9.2 and 3.8.8 are now available". Python Insider. Archived from the original on 25 February 2021. Retrieved 26 February 2021.

[74]"Red Hat Customer Portal – Access to 24x7 support and knowledge". access.redhat.com. Archived from the original on 6 March 2021. Retrieved 26 February 2021.

[75]"CVE – CVE-2021-3177". cve.mitre.org. Archived from the original on 27 February 2021. Retrieved 26 February 2021.

[76] "CVE – CVE-2021-23336". cve.mitre.org. Archived from the original on 24 February 2021. Retrieved 26 February 2021.

[77] Langa, Łukasz (24 March 2022). "Python Insider: Python 3.10.4 and 3.9.12 are now available out of schedule". Python Insider. Retrieved 19 April 2022.

[78] Langa, Łukasz (16 March 2022). "Python Insider: Python 3.10.3, 3.9.11, 3.8.13, and 3.7.13 are now available with security content". Python Insider. Retrieved 19 April 2022.

[79] Langa, Łukasz (17 May 2022). "Python Insider: Python 3.9.13 is now available". Python Insider. Retrieved 21 May 2022.

[80] "Python Insider: Python releases 3.10.7, 3.9.14, 3.8.14, and 3.7.14 are now available". pythoninsider.blogspot.com. 7 September 2022. Retrieved 16 September 2022.

[81] "CVE - CVE-2020-10735". cve.mitre.org. Retrieved 16 September 2022.

[82] corbet (24 October 2022). "Python 3.11 released [LWN.net]". lwn.net. Retrieved 15 November 2022.

[83] Anaconda Documentation: https://docs.anaconda.com/

[84]Anaconda on Github: https://github.com/anaconda

[85]Jupyter documentation: https://jupyter.readthedocs.io/en/latest/

[86]Official Jupyter website: https://jupyter.org/

[87]Project Jupyter on GitHub: https://github.com/jupyter/jupyter

[88] https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcR3gO7lu1FuCSs26xM5GlJdGXH482bX5__ZepBsXbWeUWpyBtdw

[89] https://media.istockphoto.com/id/170614926/fr/photo/la-constitution.jpg?s=1024x1024&w=is&k=20&c=JJX7w20AjqnrcqbFOq1RhhAgXqUasj1s-iLRGIOrS4I=

[90] https://phs-app-media.s3.amazonaws.com/s3fs-public/u42/IMG_0040.jpg

[91] https://images.squarespace-cdn.com/content/v1/555510e6e4b0ecb85ccf4059/cf48d258-321f-43c9-9ecf-34dca5dbb04c/RDS+-+Digital+preservation+of+historical+records+-+Our+process.jpg?format=1500w

[92] https://www.cogitotech.com/wp-content/uploads/2023/03/Character-Recognition.webp

[93] https://winbuzzer.com/wp-content/uploads/2021/02/Document-Translation-Azure-Translator-Microsoft-696x367.jpg.webp

[94] https://d3i71xaburhd42.cloudfront.net/d6a371a80b3afaa2c09b62a0d48da3049bd68119/250px/18-Figure2-1.png

[95] https://pbs.twimg.com/media/FdGUDtTXkAItKEY?format=jpg&name=900x900

[96] https://assets-global.website-files.com/5d7b77b063a9066d83e1209c/60ee08e5f4978a3485339ed5_layers.png

[97] “Que signifie Réseau de neurones artificiels (RNA)? - Definition IT de Whatis.fr.” https://www.lemagit.fr/definition/Reseau-de-neurones-artificiels-RNA (accessed Jun. 08, 2022).

[98] https://www.researchgate.net/profile/Yasser-Alginahi/publication/221909023/figure/fig1/AS:393975442755585@1470942473400/Cumulative-distributive-functions-for-reference-and-adjusted-images-Histogram_Q320.jpg