



University of Kerbala
College of Computer Science & Information Technology
Computer Science Department

Ancient Textual Restoration Using Deep Neural Networks

A Thesis

Submitted to the Council of the College of Computer Science &
Information Technology / University of Kerbala in Partial Fulfillment of
the Requirements for the Master Degree in Computer Science

Written by

Ali Abbas Ali Abo Aloaub

Supervised by

Prof. Dr. Baheeja Khudair Shukur and Prof. Dr. Asia Mahdi Naser

2024 A.D.

1445 A.H.

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

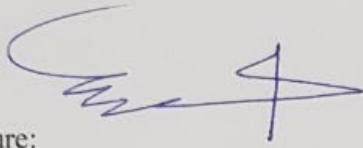
بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

صدق الله العلي العظيم

المجادلة آية (١١)

Supervisor Certification

I certify that the thesis entitled (**Ancient Textual Restoration Using Deep Neural Networks**) was prepared under my supervision at the department of Computer Science/College of Computer Science & Information Technology/ University of Kerbala as partial fulfillment of the requirements of the degree of Master in Computer Science.



Signature:

Supervisor Name: Prof. Dr. Baheeja Khudair Shukur

Date: / /2024



Signature:

Supervisor Name: Prof. Dr. Asia Mahdi Naser

Date: / /2024

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled “Ancient Textual Restoration Using Deep Neural Networks” for debate by the examination committee.



Signature:


Assist. Prof. Dr. Muhannad Kamil Abdulhameed


Head of Computer Science Department


Date: / /2024


Certification of the Examination Committee

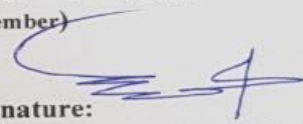
We hereby certify that we have studied the dissertation entitled (**Ancient Textual Restoration Using Deep Neural Networks**) presented by the student (**Ali Abbas Ali Abo Aloaub**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (**Excellent**) standing as a thesis for the Master degree in Computer Science.


Signature:
Name: Abdul-Wahab Sami Ibrahim
Title: Assist. Prof. Dr
Date: / / 2024
(Chairman)

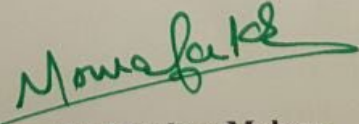

Signature:
Name: Elham Muhammed Thabit
Title: Assist. Prof. Dr
Date: / / 2024
(Member)


Signature:
Name: Asia Mahdi Naser
Title: Prof. Dr
Date: / / 2024
(Member and Supervisor)


Signature:
Name: Ayad Hamed Mousa
Title: Assist. Prof. Dr
Date: / / 2024
(Member)


Signature:
Name: Baheeja Khudair Shukur
Title: Prof. Dr
Date: / / 2024
(Member and Supervisor)

Approved by the Dean of the College of Computer Science & Information Technology, University of Kerbala.


Signature:
Assist. Prof. Dr. Mowafak khadom Mohsen
Date: / / 2024
(Dean of College of Computer Science & Information Technology)

Dedication

I'd like to dedicate this work:

To The Prophet Mohammed and his Ahl al-Bayt, and Imam Al-Mahdi;

Peace & blessings be upon them.

To my parents, brothers, supervisor and friends.

To everyone who supported me.

Acknowledgement

At the first, I would like to thank Allah the one above all of us and the omnipresent God, for answering my prayers and giving me the strength and courage to do this work.

I want to express my profound gratitude to my supervisors for their boundless dedication, guidance, and encouragement during this work. their efforts were invaluable, and I am blessed to be advised by them.

Special thanks go to all the staff members of the College of computer science and information technology for their faithful efforts to give us the utmost scientific topics and endless support in all directions.

Finally, I would like to state that no amount of words could adequately convey how grateful I am to my parents and thank my brother, sisters, and my friends for encouraging and supporting me.

Ali Abbas Ali

Abstract

Ancient texts are important because they connect us with ancient civilizations, through which we gain cultural, religious and scientific knowledge. Ancient texts, whether on papyrus, parchment, or other substrates, are often fragmented, degraded, or partially erased due to the passage of time. Restoration of these texts presents a significant challenge to historians and scholars, requiring meticulous manual effort and expertise.

Ancient text restoration is a specialized branch of the text restoration that focuses on recovering and preserving textual content from historical or ancient documents.

Traditional restoration methods rely heavily on manual intervention by experts, which is time-consuming and often subjective. In recent years, the application of machine learning (ML) and artificial intelligence (AI) techniques has shown promise in automating and enhancing the restoration process.

Deep learning techniques have shown remarkable success in various domains, including image processing and natural language processing. In this thesis, different models were proposed for the restoration of ancient texts by using deep neural networks.

Two datasets used for training and testing the models the first dataset being “Codex Sinaiticus” a manuscript dating back to the fourth century, it is a significant artifact as it provides the earliest extant complete copy of the New Testament in the Christian Bible. The handwritten material is written in the Greek language.

The second dataset being “Argonautica 3” which refers to an epic poem written by the ancient Greek poet Apollonius of Rhodes in the 3rd century BCE which is written in the Greek language too.

The dataset has been preprocessed by encoding dataset. New lines, numbers, symbols and special characters have been removed. After that the result text has been tokenized, generate missing character, class label obtained, augmentation performed to support dataset, and normalization process performed.

Three prediction models were used as proposed models for retrieving missing ancient texts, Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN), and Generative Adversarial Networks (GAN) and the results were testing accuracy 86%, 92% and 98.3% according to the first dataset and 94%, 88% 98.7% according to the second dataset respectively.

Comparing the performance of each model, GAN gave the best accuracy results, and thus it proved its effectiveness in the field of restoring missing text. The results of the proposed system were also compared with other restoration techniques, where the results showed that the proposed technique had higher accuracy results than others.

Overall, this work contributes to the interdisciplinary intersection of deep learning and digital humanities, offering a promising solution for the restoration and preservation of ancient textual artifacts.

Declaration Associated with this Thesis

[1] A. A. A. Alkhazraji, B. Khudair and A. M. N. Alzubaidi, "Ancient Textual Restoration Using Deep Neural Networks: A Literature Review," in 2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT), Al-Muthana, Iraq, 2023.

[2] A. A. A. Alkhazraji, B. Khudair and A. M. N. Alzubaidi, "Ancient Textual Restoration Using Deep Neural Networks," in Fifth International Scientific Conference of Alkafeel University (ISCKU 2024), Al-Najaf Al-Ashraf, Iraq, 2024.

Table of Contents

Dedication	i
Acknowledgement	ii
Abstract.....	iii
Declaration Associated with this Thesis	v
Table of Contents.....	vi
List of Tables	ix
List of Figures	x
List of Abbreviations	xi
CHAPTER ONE: GENERAL INTRODUCTION.....	1
1.1 Review	1
1.2 Introduction.....	1
1.3 Problem Statement	2
1.4 Aim of the research.....	3
1.5 Challenges and limitations	3
1.6 Thesis layout	4
CHAPTER TWO: THEORETICAL BACKGROUND	5
2.1 Reveal.....	5
2.2 Introduction	5
2.3 Ancient languages	6
2.3.1 Ancient language background	7
2.3.2 Importance of ancient language restoration	8
2.4 Text processing	8
2.5 Deep Learning.....	11
2.5.1 Activation Function.....	12
2.5.1.1 Rectified Linear Activation (ReLU)	13
2.5.1.2 Sigmoid	14
2.5.1.3 SoftMax.....	14
2.5.2 Loss Function.....	15
2.5.3 Optimization Algorithms	15
2.5.3.1 Adam.....	16
2.5.4 Back Propagation Technique in Neural Networks.....	17
2.5.5 Dimensionality Reduction.....	20

2.5.6 CNN with 2D Architecture	21
2.5.6.1 Generative Adversarial Network (GAN)	21
2.5.6.2 Long Short-Term Memory (LSTM)	22
2.5.6.3 Recurrent Neural Networks (RNNs).....	24
2.6 Evaluation Measures	26
2.7 Literature Review.....	28
CHAPTER THREE: PROPOSED METHODOLOGY	36
3.1 Review	36
3.2 The proposed system.....	36
3.2.1 Dataset Cleaning	38
3.2.1.1 UTF-8 Encoding	38
3.2.1.2 Remove new lines, numbers, and special characters	39
3.2.2 Tokenization	39
3.2.3 Missing Random Character Generation	42
3.2.4 Tokens Augmentation.....	42
3.2.5 Text encoding	42
3.2.6 Reshape	43
3.2.7 Normalization	43
3.2.8 Long Short-Term Memory (LSTM)	44
3.2.9 Recurrent Neural Network (RNN).....	47
3.2.10 Generative Adversarial Network (GAN)	48
CHAPTER FOUR: RESULTS AND DISCUSSION	54
4.1 Review	54
4.2 Hardware requirement.....	54
4.3 Dataset	54
4.3.1 Codex Sinaiticus	54
4.3.2 Argonautica, 3	55
4.4 Case study	55
4.4.1 Dataset preprocessing.....	55
4.4.1.1 UTF-8 Encoding	55
4.4.1.2 Removing numbers and special characters	56
4.4.1.3 Remove newline symbol.....	56

4.4.1.4 Tokenization.....	57
4.4.1.5 Compute language characters	57
4.4.1.6 Generate missing character	57
4.4.1.7 Compute target class characters	58
4.4.1.8 Augmentation process	59
4.4.1.9 Text encoding	59
4.4.1.10 Reshpe	59
4.4.1.11 Normalization.....	59
4.4.2 Prediction models.....	59
4.4.2.1 LSTM model.....	59
4.4.2.2 RNN model	63
4.4.2.3 GAN model.....	67
4.5 Discussion	72
CHAPTER FIVE: CONCLUSION AND FUTURE WORKS	73
5.1 Review	73
5.2 Conclusions.....	73
5.3 Suggestions for future works	74
REFERENCES.....	75

List of Tables

<i>Table 2-1: Studies summarization.....</i>	<i>34</i>
<i>Table 4-1: The precision, recall, f1 score, and support measures for the LSTM model.....</i>	<i>62</i>
<i>Table 4-2: The precision, recall, F1-score, and support measure for the RNN model.....</i>	<i>66</i>
<i>Table 4-3: The precision, recall, F1-score, and support measure for GAN model</i>	<i>70</i>
<i>Table 4-4: Studies comparison</i>	<i>71</i>

List of Figures

<i>Figure 2-1: ReLU Activation Function.</i>	13
<i>Figure 2-2: Logistic (Sigmoid) Activation Function.</i>	14
<i>Figure 2-3: Figuring out the relationship between the gradient and the loss function</i>	18
<i>Figure 2-4: GAN network</i>	22
<i>Figure 2-5: LSTM network</i>	24
<i>Figure 2-6: RNN network</i>	26
<i>Figure 2-7: Confusion matrix</i>	27
<i>Figure 3-1: The proposed system for ancient text restoration.</i>	37
<i>Figure 4-1: Removing '\n' from the dataset</i>	56
<i>Figure 4-2: A sample tokenization process.</i>	57
<i>Figure 4-3: The character of language indicates 41 characters.</i>	57
<i>Figure 4-4: Sample of missing values and their classes</i>	58
<i>Figure 4-5: Target class character list appointed to 26 classes.</i>	58
<i>Figure 4-6: Summary of LSTM model layers.</i>	60
<i>Figure 4-7: Training process for LSTM model.</i>	60
<i>Figure 4-8: LSTM training and validation accuracy.</i>	61
<i>Figure 4-9: Training and validation loss.</i>	61
<i>Figure 4-10: LSTM model confusion matrix.</i>	63
<i>Figure 4-11: The summary of RNN layers.</i>	63
<i>Figure 4-12: The training process of the RNN model</i>	64
<i>Figure 4-13: The training and validation accuracy for the RNN model</i>	65
<i>Figure 4-14: The training and validation loss for RNN model.</i>	65
<i>Figure 4-15: The confusion matrix for the RNN model</i>	67
<i>Figure 4-16: The summary of GAN layers.</i>	67
<i>Figure 4-17: The training process of the GAN model</i>	68
<i>Figure 4-18: The training and validation accuracy of the GAN model.</i>	69
<i>Figure 4-19: The training and validation loss of GAN model.</i>	69
<i>Figure 4-20: The confusion matrix for the GAN model.</i>	71

List of Abbreviations

Abbreviation	Description
BRNN	Bidirectional Recurrent Neural Network
BPTT	Backpropagation Through Time
CM	Confusion Matrix
CNN	Convolution Neural Network
CRF	Conditional Random Field
GAN	Generative Adversarial Networks
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
NLP	Natural Language Processing
ReLU	Rectified Linear Activation
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent
UTF-8	Unicode Transformation Format 8-bit

CHAPTER ONE
GENERAL
INTRODUCTION

1.1 Review

This chapter presents an introduction about ancient text restoration, problem statement, aim of the research, challenges and limitations, and thesis layout.

1.2 Introduction

Ancient texts are written records or documents that date back to ancient civilizations and societies. They serve as valuable windows into the past, offering insights into the history, culture, literature, religion, politics, science, and daily life of ancient peoples [1].

Study and interpretation of ancient texts, known as philology, play a crucial role in understanding the development and evolution of human societies. Scholars and researchers in fields such as history, archaeology, anthropology, linguistics, and religious studies rely heavily on these texts to reconstruct narratives of ancient civilizations and to piece together stories of their ancestors [2].

Those documents are typically written on materials such as papyrus, parchment, clay tablets, bamboo slips, or other ancient media, and may be thousands of years old. The restoration process involves overcoming the challenges posed by the passage of time, physical damage, decay, and fading of the writing, which often makes the text difficult to decipher [3].

The importance of ancient text restoration lies in its ability to unlock the knowledge and wisdom of past civilizations. These ancient texts offer valuable insights into the history, culture, language, beliefs, and practices of societies that have long since disappeared. By restoring these

texts, researchers, historians, archaeologists, and linguists can shed light on ancient traditions, literature, scientific discoveries, religious practices, and much more [4].

The restoration of ancient texts is a multidisciplinary endeavor that brings together experts from various fields. It requires a combination of skills, including paleography (the study of ancient writing systems), epigraphy (the study of inscriptions), philology (the study of language and linguistic history), and material science. Additionally, technological advancements in imaging, spectroscopy, and other analytical tools play a significant role in examining and deciphering ancient texts [5].

Artificial intelligence and machine learning play crucial roles in the restoration of ancient texts by offering innovative tools and techniques for digitizing, analyzing, and reconstructing fragmented or damaged texts [6].

1.3 Problem Statement

The problem of predicting the missing part of text is a common task in Natural Language Processing (NLP) and is often referred to as ‘text completion’ or ‘text generation.’ It involves predicting the most likely continuation or missing words in a given sentence or text fragment. This task is a form of language modeling, where the model learns statistical patterns and relationships between words to make accurate predictions about the missing part [7].

The restoration of ancient texts is of immense importance for several reasons, as it offers numerous benefits for scholars, historians, linguists, and society as a whole such as Preserving Cultural Heritage, Understanding History, Advancing Scholarship, Revealing Lost Knowledge, Resolving Historical Debates, Fostering Cultural Exchange,

Correcting Biases and Misconceptions, Enhancing Language Studies, Enriching Literature and Arts, and Strengthening Cultural Identity.

Those texts, “inscriptions”, are often damaged over the centuries, and illegible parts of the text must be restored.

1.4 Aim of the research

The main aim of this study is to develop ancient text restoration model.

To achieve the main aim, the following procedures should be outlined:

1. How to identify the main component of the proposed model.
2. To develop the proposed model using deep neural networks.
3. To evaluate the proposed model in terms of accuracy.

1.5 Challenges and Limitations

Ancient text restoration is a complex and challenging task due to various factors related to the nature of texts and conditions in which they have survived over time. Some of key challenges of ancient text restoration include:

1. Deterioration and damage: ancient texts are often physically degraded due to the passage of time, exposure to environmental elements, and other natural or human-made factors. This deterioration can lead to missing or illegible portions, making the restoration process difficult.
2. Fragmentation: Many ancient texts are fragmented, with pieces scattered across different locations or collections. Reassembling

these fragments to reconstruct the original text can be a laborious and time-consuming task.

3. Ancient scripts and languages: ancient texts may be written in scripts and languages that are no longer in common use, making decipherment and translation challenging. Some scripts may not have a clear relationship with modern languages, adding complexity to the restoration process.
4. Limited availability of source materials: Access to ancient texts and artifacts may be restricted due to their fragility, location, or ownership, making it difficult for researchers to study and restore them.
5. Data limitations: Incomplete or limited data on the original text and historical context can hinder the accuracy and completeness of the restoration process.

1.6 Thesis Layout

The remnant of the thesis is ordered as follows:

Chapter Two: It introduces an introduction about ancient text, text preprocessing, CNN models for text prediction, and model evaluation.

Chapter Three: It presents the proposed system, discusses the practical stages of the suggested model, and explains the details of each step in these stages.

Chapter Four: It contains the results of the proposed system.

Chapter Five: It introduces conclusions and suggestions for future works.

CHAPTER TWO
THEORETICAL
BACKGROUND

2.1 Review

This chapter will present an introduction, ancient languages, text preprocessing, deep learning, prediction models such as GAN, LSTM and RNNs, optimization algorithms, and literature review.

2.2 Introduction

The discipline of the ancient text restoration is a captivating and essential area of academic inquiry that focuses on the safeguarding and retrieval of historical documents and manuscripts from past epochs. Ancient books frequently function as portals to the past, offering unique perspectives on the history, culture, language, and knowledge of civilizations that before. The restoration of ancient writings necessitates a meticulous approach that integrates interdisciplinary knowledge and cutting-edge technologies [8].

The aforementioned writings encompass several forms of written records throughout antiquity, including inscriptions on stone tablets, papyrus scrolls, parchment manuscripts, and other mediums employed for the purpose of documenting knowledge. Over the course of time, the degradation of these materials occurs as a result of various circumstances, including environmental conditions, natural disasters, and intentional acts of destruction. Consequently, the textual content may have a deterioration in legibility, sustain damage, or ultimately face full loss. [9]

The field of ancient text restoration involves a variety of tasks, such as the interpretation of eroded or partially impaired writing, the reconstruction of absent sections of text, and the conservation or safeguarding of delicate materials. A collaborative effort is undertaken by scholars, archaeologists, linguists, and conservators to decipher the

enigmatic content enshrined within these antiquated manuscripts. A diverse range of methodologies is utilized, including multispectral photography, chemical analysis, and consideration of historical context, in order to reconstruct the intricate tapestry of historical events and cultural phenomena [10].

The diligent endeavors of scholars specializing in the restoration of ancient texts afford the opportunity to access a vast reservoir of knowledge that would otherwise remain obscured, thus enhancing the comprehension of historical events and the intellectual and aesthetic accomplishments of those who came before. The restoration of these texts has a dual purpose: illuminating historical events and enhancing the continuous progression of human understanding while fostering a deeper appreciation for the rich and varied fabric of human history [11].

2.3 Ancient languages

Languages serve as the foundation upon which human civilization is built, containing not just proposed methods of communication but also it is a very essence of culture, thought, and historical development. They are living reservoirs of accumulated knowledge and expertise that has been accumulated through countless generations [12].

A rich tapestry of languages has come and gone over the course of the millennia, and each of these languages has left a distinct mark on the very fabric of human existence. Research destination will be a place where ancient languages have left an indelible mark on the course of human history [13].

2.3.1 Ancient languages background

Ancient languages are like echoes of long-forgotten conversations, whispering to the world across the centuries and beckoning to uncover their secrets. They are the keys to understanding the intricate tapestry of human civilization, each offering a unique window into beliefs, customs, and intellectual accomplishments of their respective cultures.

Take, for example, Sanskrit, an ancient Indo-European language celebrated for its role as the sacred language of Hinduism and the treasure trove of knowledge contained in texts like the Vedas, Upanishads, and epics such as the Mahabharata [14]. Its enduring significance lies in its profound influence on Indian culture and philosophy.

Similarly, Latin, the language of the Romans, was not only the administrative and literary language of a powerful empire but also the foundation upon which much of Western thought and scholarship was built. It shaped disciplines such as law, medicine, and science, contributing to the richness of scientific nomenclature and legal terminology [15].

The hieroglyphic script of ancient Egypt, with its enigmatic symbols, is a testament to the fascination with human history, culture, and spirituality. Deciphered thanks to the Rosetta Stone, it unveiled the stories of pharaohs, religious beliefs, and monumental achievements carved into the stone walls of ancient temples. These examples represent just a fraction of linguistic treasures that connect us to our ancestors, illuminate our collective history, and shape our modern world, demonstrating the inexhaustible depths of knowledge and cultural significance held within these ancient tongues [16].

2.3.2 Importance of ancient language restoration

The restoration of ancient texts is of the utmost importance because it acts as a connection to the cultural and historical history. It also helps to preserve priceless insights into values, practices, and information that were held by previous civilizations [17].

These newly restored manuscripts provide a one-of-a-kind view into annals of history. They reveal historical tales, linguistic intricacies, and contextual clues that not only deepen the comprehension of the past but also advance the study of linguistics, archaeology, and multidisciplinary studies. By cracking codes of these ancient languages, they make the researchers able to access the vast store of information that they contain [18].

This information spans fields of science, medicine, and mathematics, in addition to the artistic and literary traditions of antiquity. In addition, they frequently hold relevance in realms of law, diplomacy, and sociology, providing a comprehensive understanding of the relationships that existed amongst societies of the past. By restoring ancient texts, we are preserving threads of the cultural tapestry, so re-establishing the connection to the origins, and ensuring that the voices of the ancestors will continue to reverberate and inspire people in the current world [19].

2.4 Text processing

The preparation of text data is an essential stage in the field of natural language processing (NLP) and data analysis. It encompasses a sequence of procedures aimed at converting unprocessed textual data into a refined and organized format that is well-suited for a variety of NLP

applications. The aforementioned procedure is crucial in enhancing the quality of data and enabling more precise and efficient analysis [20]. The fundamental stages involved in the preparation of text data encompass:

1. **Data collection:** The initial phase entails the acquisition of unprocessed textual data from many sources, encompassing web scraping, document retrieval, or social media application programming interfaces (APIs). Thorough data gathering is crucial and necessitates careful consideration of the analysis's specific requirements [21].
2. **Text cleaning** is a process that aims to remove undesirable components from a given text, such as HTML tags, special characters (e.g., punctuation marks), metadata, or non-textual material. The correction of spelling problems is undertaken in order to enhance the quality of data [22].
3. **Tokenization** refers to the procedure of dividing a given text into discrete units known as tokens, which often correspond to individual words or other meaningful elements. This stage facilitates the examination of text at a fine-grained level, hence simplifying the use of subsequent techniques such as stemming or lemmatization [23].
4. The process of converting all text to lowercase is necessary in order to maintain consistency. This process serves to eliminate the distinction between the same term with different letter cases, hence minimizing redundancy in the data.
5. **Stopword removal:** Stopwords refer to frequently occurring words, such as "the," "is," "in," and "and," that contribute minimal semantic value to the text. By eliminating these phrases, the data's

dimensionality is reduced, allowing for a greater emphasis on keywords that provide more meaningful information [24].

6. Stemming and lemmatization are two techniques used in natural language processing. Stemming involves reducing words to their root form by deleting prefixes or suffixes, such as transforming "running" to "run". On the other hand, lemmatization involves mapping words to their base form using a lexicon, for example, transforming "better" to "good". These strategies aim to establish a standardization of word variants, providing a uniform treatment of related words [25].
7. Exclusion of numerical characters: Numeric characters are often omitted unless the analysis pertains to numerical data. The exclusion of numerical values simplifies the text and aids in directing attention towards the textual content.
8. Handling missing data: The management of missing or null values in textual data is of utmost importance in order to mitigate errors and uphold the integrity of the data. Depending on the characteristics of the data, the process may entail either imputing missing values or eliminating incomplete records [26].
9. The process of encoding and decoding involves the proper utilization of encoding schemes, such as UTF-8, to enable accurate interpretation of textual data, particularly in scenarios involving diverse character sets and languages. The inclusion of this phase is crucial in effectively managing text that is written in multiple languages.
10. Normalization involves several tasks, such as the conversion of abbreviations into their expanded forms (e.g., transforming "don't" into "do not") and the management of terminology specific to a certain domain. This modification enhances the coherence and

comprehensibility of the text [27]. Equation (2.1) represents the z-score normalization.

$$z = \frac{x - \mu}{\sigma} \quad (2.1)$$

Where x is the value

μ : the mean

σ : the standard deviation.

11.The removal of irrelevant information: Textual data frequently encompasses data that is not germane to the analysis. The exclusion of extraneous content, such as metadata or boilerplate text, guarantees that the analysis remains centered on the pertinent textual information.

12.Text length filtering: The application of filtering techniques to exclude texts that are too short or long is crucial in ensuring data quality by mitigating the risk of inadequate information or the introduction of noise.

13.Customized data-unique cleaning: This stage involves the execution of cleaning tasks that are tailored to the unique data, such as the elimination of jargon specific to a certain company or words relevant to a particular industry. The utilization of custom cleaning techniques is vital for conducting domain-specific studies [28].

2.5 Deep Learning

The extraction of meaningful and relevant features is crucial in machine learning. Traditional machine learning cannot understand or process raw natural input, making AI problem-solving problematic.

These issues were addressed by deep Learning approach which combining AI with machine learning [29],[30].

Deep learning uses many processing layers to build computer models that can recognize subtle structures in big datasets by learning data representations at different abstraction levels. How a computer may change the parameters needed to generate each layer from the preceding layer is a concern [31].

With large increases in data, deep learning algorithms are the most used. CNN is popular in deep learning where a layer-based neural network is used. Each layer does convolution, loss, aggregating, computation, etc. [32]. Previous layer output becomes next layer input. CNNs have been used in computer vision for a long time, but the 2012 ImageNet competition revolutionized graphics processing unit (GPU) use and data augmentation [33].

Deep learning had several uses. In [34], authors proposed a deep learning age and gender classification model. Deep learning was used to classify face expressions in [35]. The researchers created a deep learning algorithm to detect cranial anomalies while driving. Deep learning was used to classify a driver's eyes open or closed to prevent accidents [36]. In [37], deep learning identified head movement to diagnose illness.

2.5.1 Activation Function

The activation functions are a fundamental component used with artificial neural networks (ANNs) to transform input signals to output signals. Later strata receive this output signal as input. In an ANN, count the input products and their weights, then add an activation function to calculate the layer output and send it to the next layer [38].

Most activation functions are linear or nonlinear. When using linear activation, output matches input. It can only adjust to linear changes and solve basic issues with some mistake. The complex and nonlinear equations of backpropagation cannot be solved by the input because its derivative value must remain constant [39].

The nonlinear activation functions with multi-degree curves. Multiple-layer neural networks must learn, represent, and comprehend input-output data and complex issues. Non-linear input-output mappings require an activation function. Due to backpropagation, the nonlinear activation function reduces flaws [38]. Essential activation functions for concealed layers are below.

2.5.1.1 Rectified Linear Activation (ReLU)

The ReLU is a type of neural network that is often used, especially in deep learning models. It works by putting the threshold to 0. Simply put, it gives back 0 when x is zero or negative and the same number when x is not zero or negative. The function that the equation (2.2) stands for. The plot of the ReLU activation function [40] is shown in Figure (2.1).

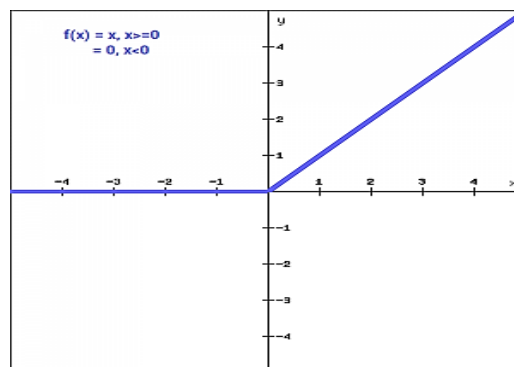


Figure (2.1) ReLU Activation Function [41]

$$ReLU(x) = \max(0, x) \quad (2.2), [40]$$

2.5.1.2 Sigmoid

Since the sigmoid is a nonlinear function, it is very most popular, especially in binary classification. As seen in Figure (2.2), the sigmoid function modifies values in the interval 0 to 1; it can be expressed as equation (2.3) [42]:

$$f(x) = 1/(1 + e^{-x}) \quad (2.3), [42]$$

The sigmoid function is a smooth S-shaped function that is continuously differentiable [38]. The function's derivative is given by equation (2.4), and Figure (2.2) depicts the sigmoid activation function.

$$f'(x) = 1 - \text{sigmoid}(x) \quad (2.4), [42]$$

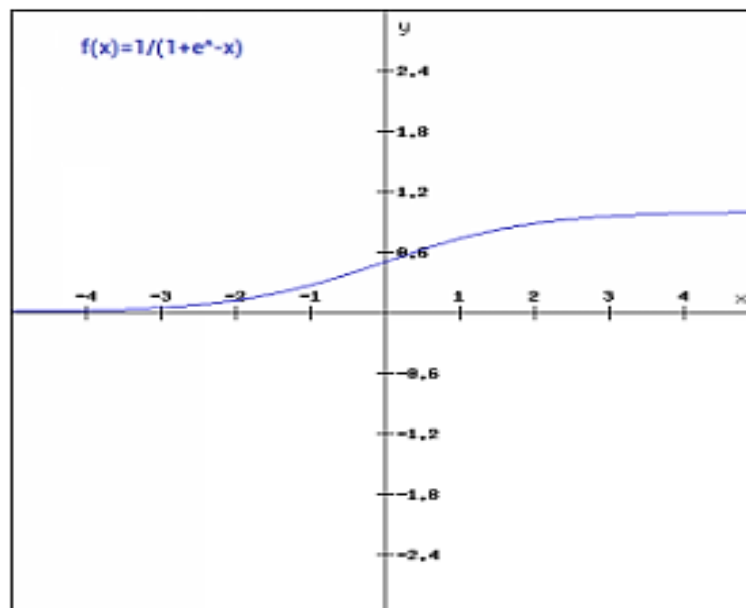


Figure (2.2) Logistic (Sigmoid) Activation Function [38]

2.5.1.3 SoftMax

The SoftMax function is one type of activation function. It is often used in neural computing at the last layer to calculate the multiple probability distributions of multi-classes with more than two classes by

using a list of real values. The result of the Softmax function is a number between 0 and 1 where the sum of all the probabilities is 1, and the class with the highest value is the goal class. Softmax is shown in equation (2.5), which is [39].

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (2.5), [39]$$

2.5.2 Loss Function

Earlier models of neural networks measured error by comparing the actual output to the predicted output. Numerous formulae, known as Loss Functions [43], for calculating neural network error have emerged.

Each loss function will produce a distinct error value for the same prediction if the network is trained with multiple loss functions. Loss functions can be categorized as classification loss functions, regression loss functions, and embedding loss functions [44].

Classification issues are addressed with classification loss functions. Regression problems employ regression loss functions. While the embedding loss functions are used to measure the similarity between two inputs, the embedding loss functions are utilized for tasks that require measuring the similarity between two inputs. Training a neural network with widely-used loss functions [45].

2.5.3 Optimization Algorithms

Choosing an algorithm to optimize a neural network is a crucial step. Batch or deterministic gradient techniques, which manage all training examples in a large batch at once, and stochastic or online

methods, which only deal with a single instance at a time, are the most common types of optimization techniques in machine learning [44].

Optimization algorithms include adaptive learning algorithms and constant learning rate algorithms, such as stochastic gradient descent (SGD). In the first category, there is a manual selection of a learning rate. In this type of algorithm, determining the learning rate is a difficult task. Selecting a low learning rate slows down the learning process and lengthens the duration of training. If you select a relatively high learning rate, which slows down the convergence process, the loss value may vacillate around the minimum value. The learning rate for the algorithms in the second group, however, does not need to be manually set. Instead, they employ a heuristic method that automatically adjusts the learning rate [46]. Several algorithms that fall into these two categories have been created, with the most significant one is below:

2.5.3.1 Adam

This adaptive learning rate optimization technique (Adam) evaluates the rate at which every individual learns a particular parameter (or set of parameters). Adam utilized first- and second-moment estimations in order to establish Adam's learning parameters. A random variable raised to the power of n represents the current expectation. The moment can be mathematically demonstrated in equation (2.6), [47]:

$$m_n = E[X^n] \quad (2.6), [47]$$

Where: m is the moment, and X is a random variable.

The following equations (2.7) ,(2.8) are used to estimate, the first and second moments Adam [47].

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (2.7), [47]$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (2.8), [47]$$

Where m_t is the previous first moment and v_t is the previous second moment. In the first step, both of these values are set to 0.

β_1 and β_2 are two new parameters that have been added to the algorithm. They are set to 0.9 and 0.999 as their default values.

After finding out the values of the first and second moments, the network weights are changed using the following equation (2.9).

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (2.9), [47]$$

Where W is network weights, η is Step size, $\epsilon = 10^{-8}$

2.5.4 Back Propagation Technique in Neural Networks

Back-propagation (BP) is the most important neural network preparation process. The error rate of the previous epoch is used to fine-tune neural network weights. Weight adjustments reduce mistakes and make the model more general, improving precision [45]. The BP algorithm, which uses the weights update approach, is used to educate multi-layer networks to recognize patterns. Using the optimization function, the BP method alters initial weights to lessen the disparity between intended and predicted output [48]. Most optimization algorithms assess the partial loss function's weight derivative, the gradient. Gradient measurements determine the opposing weight

adjustment. This loop continues until the model fails [44]. Figure 2.3 shows how to calculate gradient and how loss function and network weights relate.

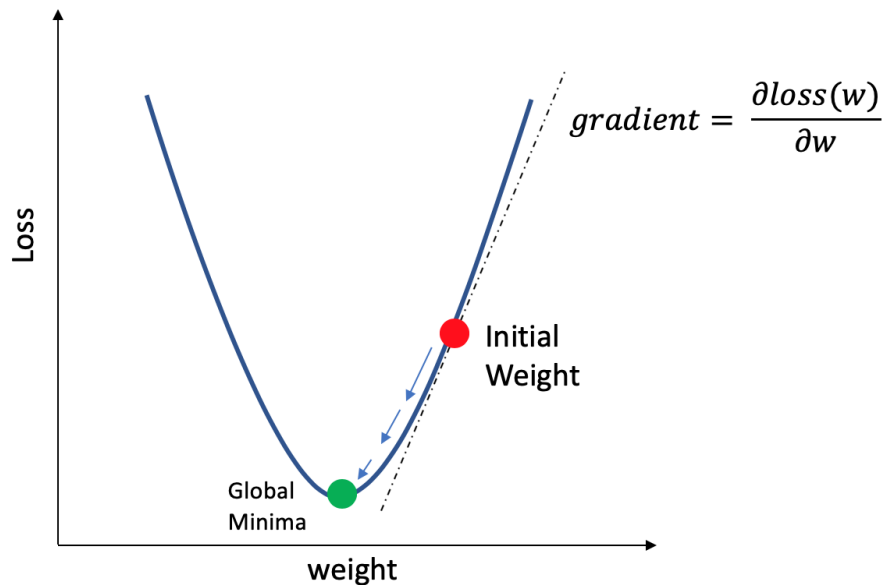


Figure (2.3) Figuring out the relationship between gradient and loss function [76]

The BP algorithm includes the following forward and reverse steps:

- a. The procedure is depicted in Figure (2.3) of the perceptron.
 1. Patterns of input are transmitted into the network.
 2. Compute predictions of output.
 - Compute the net, which is the sum of the product and input weights, using the formula: (2.1).
 - Using the activation function, calculate $f(\text{net})$, the expected output denoted as 1 out or 0 for the net in equation (2.2).
 3. Compare desired (target) and predicted (out) outputs, error differences between the desired and predicted output, for each

neural output must be calculated using a loss function such as cross-entropy, which is the preferred one to be used with the output that is distributed according to the probability of each class, thus cross-entropy loss function is an appropriate loss function for softmax or sigmoid classifiers that return the probability of each class. The expression appears as the formula (2.10) [49] below.

$$H(\text{out}) = -\sum_i(\text{target}) \log(\text{out}) \quad (2.10), [49]$$

- b. The procedure of backpropagation process in NN.
1. Recalculated errors must be propagated in the opposite direction.
 2. Gradient descent, one of the most prevalent optimization techniques used with the back-propagation method, is employed to bring the projected output closer to the intended output. The total error is derived primarily from $E(\text{total})$ in order to make adjustments. The error with respect to a particular weight is also referred to as gradient descent with respect to w_1 , for example using the chain rule as shown in the equation below: (2.11). [49], [50].

$$\frac{\partial E(\text{total})}{\partial w_1} = \frac{\partial E(\text{total})}{\partial \text{out}(o1)} * \frac{\partial \text{out}(o1)}{\partial \text{net}(o1)} * \frac{\partial \text{net}(o1)}{\partial w(i)} \quad (2.11), [49]$$

3. Weight difference value (ΔW) is derived by multiplying the resultant gradient descent with regard to a particular weight value by the learning rate (η), as shown in the following equation (2.12) [50].

$$\Delta W = \eta \frac{\partial E(\text{total})}{\partial w(i)} \quad (2.12), [50]$$

4. The present weight will be modified by subtracting the weight difference, as stated in the following equation (2.13) [50].

$$W(i)_{\text{new}} = W(i) - \Delta W \quad (2.13), [50]$$

By substituting (2.12) with (2.13), the new weight can be obtained by applying the following equation (2.14) [49].

$$W(i)_{\text{new}} = W(i) - \eta (\partial E(\text{total})) / (\partial w(i)) \quad (2.14), [49]$$

In a multi-layer or deep network, the weight-adjustment process returns to the concealed layer to update the weights before returning to the input layer from the output layer until the error difference falls below the upper error bound (E_{max}). This section is called the backward phase because a gradient learning process begins at the output node and continues going backward [51].

It is essential to note that the gradient vanishing is the greatest back-propagation concern. The vanishing gradient problem is the difficulty encountered when training a neural network as a result of a small weight update received from the preceding layer, resulting in a small weight shift that, in the worst-case scenario, can completely cease the training process. Due to the variety of activation functions, some can exacerbate the issue while others can remedy it. Due to its ability to maintain only positive values while keeping negative values close to zero, ReLU is currently the most important or default activation function [52].

2.5.5 Dimensionality Reduction

Numerous disciplines, including numerical analysis, data processing, and machine learning, confront the issue of curse dimensions. The prevalent theme of curse dimensional problems is that as the dimensionality increases, the quantity of available data decreases

precipitously. Due to the tremendous capacity of its algorithms to manage multiple dimensions, deep learning has circumvented this problem [53].

2.5.6 CNN with 2D Architecture

This topic concentrates on the architecture required to manage a particular type of data. Convolutional Neural Networks (CNN) are a biologically inspired form of feed-forward networks in which the connections between neurons attempt to record input data distortion or shift pattern invariances. The vast majority of CNN architectures presumed networks would operate with two-dimensional input data (typically images). Each layer of a conventional CNN converts one activation volume to another [54].

2.5.6.1 Generative Adversarial Network (GAN)

Ian Goodfellow and colleagues established a deep learning model Generative Adversarial Network (GAN) in 2014. GANs generate new data instances in various domains, including pictures, writing, music, and others. Their ability to produce realistic, high-quality content has made GANs popular [55].

The GAN model has two parts:

1. The generator is a neural network that uses random noise or input data to generate data that closely resembles valid data. The generator generates authentic-looking images from random noise [56].
2. The discriminator is a neural network that analyzes data to distinguish between legitimate (e.g., actual photos) and synthetic (made) data. It mostly classifies binary data [57].

Due to their adversarial training, GANs are named such. Operations work as follows:

- The generator and discriminator are trained simultaneously, but in an adversarial manner.
- The generator aims to generate data with high realism, making it indistinguishable from authentic data, leaving the discriminator unable to distinguish between the two.
- The discriminator aims to improve its ability to separate authentic data from synthetic data.
- Training improves the generator's ability to generate data that closely resembles genuine samples and the discriminator's ability to identify real from created instances. The feedback loop this creates improves data quality continuously.

GAN training aims to reach an equilibrium state where the generator generates data that cannot be distinguished from genuine data and the discriminator makes random guesses [58].

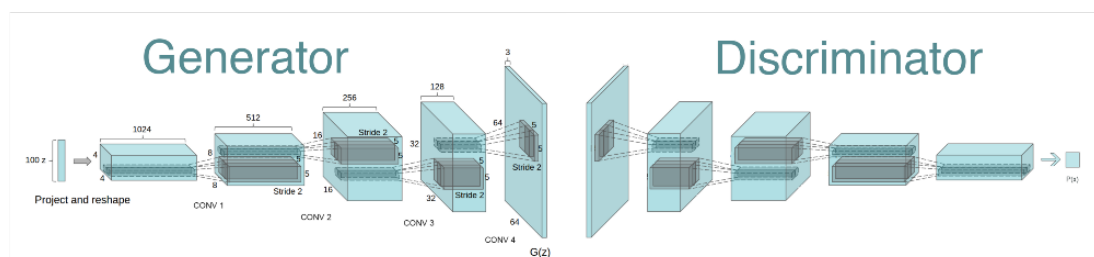


Figure (2.4) GAN Network

2.5.6.2 Long Short-Term Memory (LSTM)

LSTM solve the vanishing gradient problem, a common difficulty in standard RNNs. LSTM models excel in sequential data and time-series

analysis, making them popular in natural language processing, speech recognition, and other fields [59].

Essential LSTM traits and principles include:

1. Memory cells: LSTM models can capture and retain information over long sequences. This makes LSTMs ideal for long-term context comprehension and preservation.
2. LSTM models use gates to regulate information flow, with three types [60].
 - The forget gate mechanism discards information from the cell state.
 - The input gate selects new information for storage.
 - The output gate controls information selection for prediction [61].
3. The cell state is the internal state of the LSTM unit, located horizontally at the top. Data can be transmitted over long sequences.
4. The hidden state is the output of an LSTM cell used for prediction. The technique preferentially propagates cell state information.
5. The LSTM model uses Backpropagation Through Time (BPTT), a modified version of the backpropagation technique. This method lets the neural network learn and adjust its parameters over time [62].

LSTM models can manage and capture long-term dependencies in datasets better than RNNs. This is especially important in language modeling, because preceding words that came far earlier in the sequence might affect the semantic interpretation of a word in a phrase. Traditional

RNNs struggle to store information over long periods because to the vanishing gradient problem [63].

LSTM models are important in machine translation, text production, and speech recognition. These methods are also used in financial time series forecasting, video analysis, and other fields. More recently, LSTM counterparts like GRUs offer a comparable advantage while reducing computational complexity.

LSTMs are powerful, but Transformer-based models, a recent deep learning breakthrough, have received attention in NLP. This is due to their parallel processing and greater performance across NLP applications. However, LSTM models remain a key concept in neural networks and are still used in sequential input applications [64].

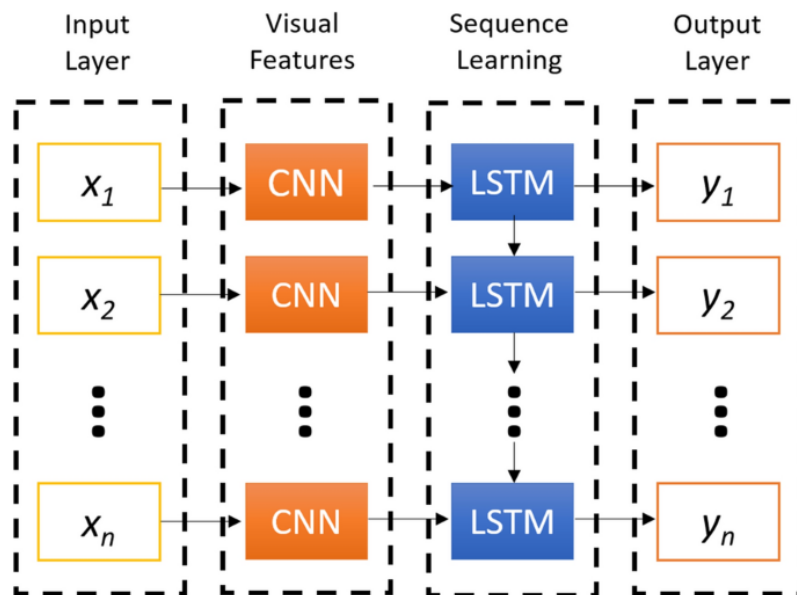


Figure (2.5) LSTM network

2.5.6.3 Recurrent Neural Networks (RNNs)

RNNs are a class of artificial neural networks designed for handling sequences and time-series data. Unlike feedforward neural networks, where information flows in one direction from input to output, RNNs have

connections that loop back on themselves, allowing them to maintain a form of memory or context [65].

Key features and concepts of RNNs include:

1. Time-dependent processing: RNNs can process sequences of data, such as time-series data, natural language, and sensor data, by taking into account the order and context of the elements in the sequence.
2. Hidden state: RNNs have a hidden state that serves as an internal memory. The hidden state at a given time step is influenced by the input at that time step and the previous hidden state. This allows RNNs to maintain context and remember information from previous time steps [66].
3. Recurrent connection: The recurrent connection in RNNs allows information to be passed from one time step to the next. This is achieved by creating a loop in the network, with the output at one time step serving as input to the same network at the next time step [67].
4. Vanishing gradient problem: RNNs are prone to the vanishing gradient problem, which occurs when gradients become too small during training, making it difficult for the network to learn long-range dependencies. This can lead to issues in retaining information over long sequences [68].

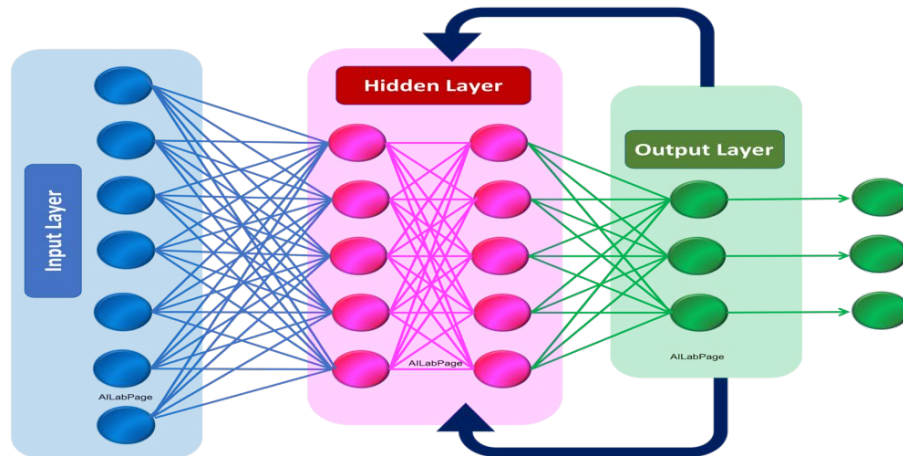


Figure (2.6) RNN network

2.6 Evaluation Measures

When constructing a machine learning model, the evaluation of performance and efficiency is crucial. For the machine learning model to be trustworthy, an assessment method must be selected that is proportional to the model's work. Frequently, while assessing machine learning models, many scales are employed to guarantee accurate evaluation. In machine learning, there are three primary types of evaluation measures: those used to assess classification, Regression and clustering tasks. Classification tasks may be evaluated using a variety of metrics, including accuracy, confusion matrix, recall, precision, and F1-score [69].

1. Accuracy measure

As demonstrated in equation (2.15), accuracy is the ratio of the average number of correct predictions to the total number of input samples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (2.15)$$

2. Confusion Matrix (CM)

CM is one of the most significant tools for providing a full explanation of the classification model's performance. The confusion matrix for a binary classification model is depicted in Figure (2.7).

		Predicted Values	
		Positive (1)	Negative (0)
Actual Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure (2.7) Confusion matrix [81]

Each prediction will fall into one of four categories:

- True positive (TP): a correctly predicted positive outcome.
- False positive (FP): a positive prediction that is inaccurate.
- True negative (TN): the correct forecast of a negative outcome.
- False negative (FN): a negative forecast that is incorrect.

The matrix accuracy can be obtained by averaging the main diagonal values using the equation (2.16) [71].

$$\mathbf{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \quad (2.16)$$

3. Recall

The Recall is the ratio of correct positives to the sum of correct positives and incorrect negatives; it is used to quantify the classifier's capacity to recognize all positive cases; and it is one of the most essential metrics utilized with models including unbalanced datasets. This metric was calculated using the following formula: (2.17).

$$\mathbf{Recall} = \frac{TP}{TP+FN} \quad (2.17)$$

4. Precision

Precision is defined as the capacity of a classifier to not label a negative instance as positive and is described by the ratio of right positives to the total of right and incorrect positives. a measure of precision that can be determined using the equation shown below (2.18) [69].

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad (2.18)$$

5. F1-score

The F-score is represented as the arithmetic mean of recall and precision. The F1 seeks to strike a balance between recall and precision and is used to quantify a test's accuracy, which determines how many instances it correctly classifies, and robustness by preventing the model from ignoring a substantial number of cases. F1 Score ranges between 0 and 1, with the bigger value indicating superior performance. The following equation (2.19) reflects the F-score measure [70]; it is the F-score measure.

$$\text{F}_1 \text{ score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.19)$$

2.7 Literature Review

Numerous researches and strenuous efforts have been made over the past years to try to recover and understand the ancient texts of the importance of these texts. Several types of research that preceded us will be mentioned in this field:

- Y. Assael et al. (2019) introduce PYTHIA, an ancient text restoration model. This model recovers missing characters from damaged text inputs using deep neural networks. The system is precisely designed

to manage long-term contextual information and accommodate missing or corrupted letter and word representations. A sophisticated pipeline was created to turn PHI, the largest digitized collection of ancient Greek inscriptions, into machine learning-friendly language for its training. This converted text is PHI-ML. According to PHI-ML, PYTHIA's predictions have a 30.1% character mistake rate, whereas human epigraphists have 57.3%. PYTHIA's Top-20 hypotheses included the ground-truth sequence 73.5% of the time. This finding proves the influence of this assistive technology on digital epigraphy, making it the standard for ancient text restoration. [72].

- H. Wang et al. (2019) uses a bidirectional LSTM + CRF model to segment Archaic Chinese phrases. The suggested method uses a linear statistical model in the bidirectional LSTM neural network to annotate sentences. This model also used stochastic gradient descent (SGD) to reduce overfitting and the viterbi algorithm to find the best sentence order. This experiment analyzes historical writings like the History of the Han Dynasty, the History of the later Han Dynasty, the Three Kingdoms, and the Book of Jin to test the proposed method. In the open test, precision, recall, and F1 are 0.77, 0.75, and 0.76. These values are 0.90, 0.88, and 0.76 in the closed test [73].
- Q. Zhao (2021) used machine-learning techniques to detect Chinese artifacts. The study classified important Chinese cities by temple, contemporary city, harbor, battle, and South China. Decision tree algorithms were utilized for recognition and gradient boosting for perception. The study found that algorithms detected ancient Chinese artifacts with 98% accuracy and prediction. The models help locate archaeological sites [74].

- Lengauer et al. (2022) introduced a methodology that enables the automated identification of the underlying generation rule associated with a repetitive surface pattern. A workflow is proposed for the reconstruction of missing or damaged parts of the surface painting, based on the generation rule and preserved patterns from the same pattern class. The authors assess the effectiveness of their methodology by implementing it on a range of pottery artifacts originating from ancient Peruvian and Greek civilizations. Their findings demonstrate that their automated approach is capable of successfully addressing diverse problematic scenarios [75].
- L. Wartschinski et al. (2022) presented an LSTM network to classify vulnerable code token sequences. This network highlights source code portions prone to vulnerabilities and provides confidence ratings for its predictions. The efficiency of Vudenc was evaluated using 1,009 vulnerability-fixing changes from various GitHub repositories. SQL injection, XSS, Command injection, XSRF, Remote code execution, Path disclosure, and Open redirect were covered in these commits. The commits were used for training. During the experiment, Vudenc achieved a recall rate of 78%-87%, a precision rate of 82%-96%, and an F1 score of 80%-90%. Vudenc's code, vulnerability datasets, and word2vec model training Python corpus are all available for replication. [76].
- Yuan et al. (2022) added the Houma Alliance Book's ancient handwritten characters database. They also presented a multi-modal fusion method for recognizing archaic handwritten letters. The database contains 297 classes and 3,547 samples of Houma Alliance old handwritten characters from the core book collection and human imitative writing. In addition, the decision-level classifier fusion

technique combines three popular deep neural network designs to recognize ancient handwritten characters. Their new database has been tested. The research community bases the new database on initial experimental results. These data demonstrate the efficacy and productivity of their methods [77].

- L. Jian et al. (2022) present the notion of a hybrid network model within the domain of deep learning. The authors proceed to develop a hybrid network model specifically designed for assessing the readability of English text. This model combines a convolutional neural network, a bidirectional long short-term memory network, and an attention mechanism network. By employing machine learning techniques to automate feature extraction, the authors successfully enhance the efficiency and performance of text readability measurement, eliminating the need for manual extraction methods [78].
- Williams et al. (2023) addressed the difficulty of mastering ground-truth cuneiform tablet transcribing subtasks. Their findings show that a RetinaNet object detector can obtain 0.78 localization mAP and a ResNet classifier can reach 0.89 top-5 sign classification accuracy. The end-to-end pipeline has 0.80 top-5 classification accuracy. In the classification module, DeepScribe clusters cuneiform signs by morphology. The researchers compare the artificial grouping method to printed sign listings and assess its potential benefits. Individual training of these components is sufficient to construct a system that can analyze Achaemenid-era cuneiform tablet pictures and offer researchers transliteration suggestions. The researchers evaluate the model's ability to identify and categorize signs, which can inform the creation of a linguistically competent transliteration system. They also

examine if the approach can be applied to other cuneiform writing periods [79].

- Papavassileiou et al. (2023) Present a generative neural language model for Mycenaean Greek, the earliest known form of Greek and related with the Linear B script. They propose using a Bidirectional Recurrent Neural Network (BRNN) to assess Mycenaean documents' statistical trends and comparing it to the n-gram approach. The approach supports defective Mycenaean texts, particularly partially incomplete words on clay tablets with variable degrees of degradation. Their strategy is experimentally verified using ground-truth data. They next apply their strategy to real-world instances and compare the outcomes to expert judgments. They also suggest a way to update their dataset, which improves their results [80].
- Locaputo et al. (2023) explain their study on developing a deep learning model to restore ancient Latin inscriptions. The first step was to find a database with many old Greek-Latin inscriptions from reputable corpora. Their technique comprises creating a data pre-processing pipeline for denoising, normalisation, and annotation schema unification. They found four promising deep-learning methods to fill Latin text gaps using various methods. The concept is based on the cutting-edge model for restoring ancient Greek inscriptions. The next idea uses pre-trained language models. The third concept uses recent computer vision and diffusion model advances. Finally, an additional proposal addresses the first and third techniques' drawbacks, particularly the need for epigraphists' gap dimensions conjectures [81].
- Wenjun et al. (2023) two-branch character restoration networks were introduced. The Example Attention Generative Adversarial Network

(EA-GAN) model uses reference examples and a generative adversarial network framework. Reference the features of the sample character to accurately restore a damaged character, even if the damage is substantial. To extract the damaged and example characters' unique properties, the EA-GAN model uses two branches. The impaired character is then restored using contextual information and the reference character's distinctive features at various magnification settings during up-sampling. To solve mismatch between example and damaged character features and a limited convolution receptive field, an Example Attention block is used to facilitate restoration. The research uses qualitative and quantitative analysis on two datasets: MSACCS, which the researchers created, and real scene photos. EA-GAN uses the Example Attention block's additional example to achieve accurate text structure, unlike other inpainting networks. The peak PSNR increased 9.82% and the structural similarity (SSIM) increased 1.82%. The Visual Geometry Group (VGG) network and AlexNet calculated a 35.04% and 16.36% drop in learnt perceptual image patch similarity (LPIPS). Their method outperformed inpainting methods. The digital conservation of ancient Chinese literary works is made easier by its ability to restore in the presence of untrained people [82]. Table (2.1) summarizes this research

Table (2.1) Studies summarization

Authors	Year	Methodology	Dataset	Performance
Y. Assael et al [72]	2019	PYTHIA	Greek inscriptions	73.5% error rate
H. Wang et al [73]	2019	LSTM+CRF	Ancient Chinese sentence	76% F-score
Q. Zhao [74]	2021	Decision tree algorithm for recognition and gradient boosting for perception aspects	Ancient artifacts found throughout China	98% accuracy
Lengauer et al [75]	2022	Content-based pattern recognition	Pottery from the Josefina Ramos de Cox museum	5% and 15% construction error
L. Wartschinski et al. [76]	2022	LSTM	Datasets for the vulnerabilities, and the Python corpus	80%-90% F1-score
Yuan et al. [77]	2022	DCF-LAR	Houma Alliance Book	84.82% accuracy
L. Jian et al. [78]	2022	CNN proposed model	English handwriting texts	89.1% accuracy
Williams et al. [79]	2023	ResNet	5,000 annotated tablet images	89% accuracy

Papavassileiou et al. [80]	2023	BRNN	Mycenaean Greek	76.68% accuracy
Locaputo et al. [81]	2023	LSTM	Epigraphik-Datenbank Clauss/Slaby (EDCS)	-
Wenjun et al. [82]	2023	EA-GAN	Self-built dataset MSACCS	9.82% PSNR

CHAPTER THREE

PROPOSED

METHODOLOGY

3.1 Review

The proposed system practical stages have been discussed in this chapter. The dataset has been preprocessed by UTF-8 encoding, removing new lines, numbers, and special characters. After that, the text has been tokenized, and the generation of missing character were done randomly from each token then each token was labeled to complete the data set.

Some of the tokens are rarely repeated for that augmentation is necessary. Text encoding is essential to facilitate computations. Then reshaping stage involves restructuring the data into the appropriate shape that the neural network expects. The dataset has been normalized. Different models have been trained and tested the dataset to predict the results such as LSTM, RNN, and GAN.

3.2 The proposed system

The suggested proposed system is used for solving the ancient text restoration. This proposed system is shown in Figure (3.1).

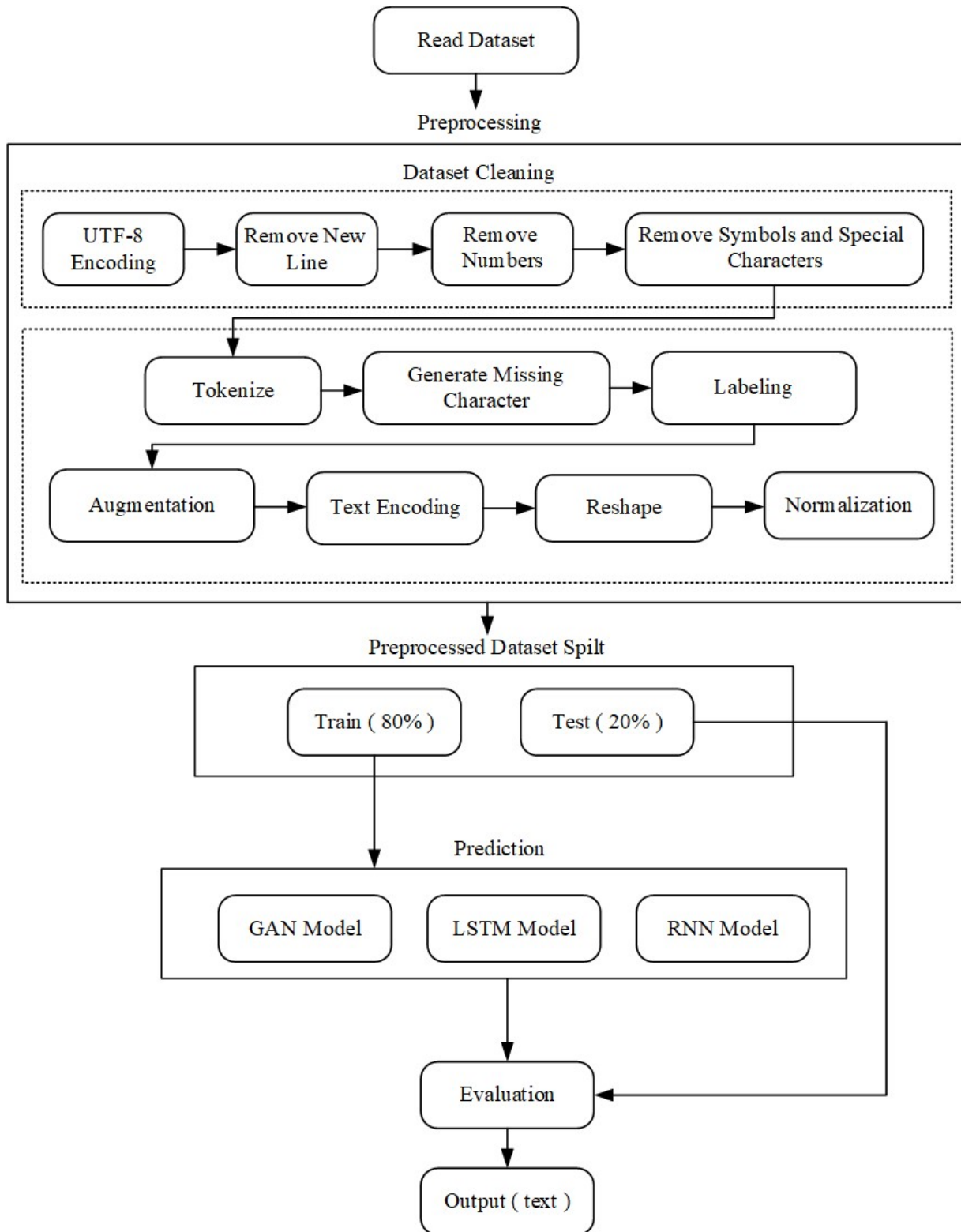


Figure (3.1) The proposed system for ancient text restoration

3.2.1 Dataset Cleaning

The process of dataset cleaning, also known as data preparation, is a crucial step in which errors, inconsistencies, and missing values within a dataset are identified and corrected. This procedure is necessary to assure the accuracy, dependability, and appropriateness of the dataset for analysis or machine learning purposes. The process encompasses various tasks, including managing missing data, addressing outliers, validating data format and values, eliminating duplicates, standardizing and normalizing data, encoding categorical variables, undertaking feature engineering, and conducting assessments of data quality. The iterative and collaborative nature of this process is crucial in order to guarantee the high quality of the dataset and the reliability and significance of the subsequent data analysis or machine learning models.

3.2.1.1 UTF-8 Encoding

The Unicode Transformation Format 8-bit (UTF-8) encoding standard is employed for the purpose of representing text in various writing systems that are prevalent worldwide. The utilized encoding scheme exhibits variable length, whereby a character is represented by a sequence of one to four bytes. It is worth mentioning that frequently encountered characters, such as ASCII, are typically encoded utilizing a solitary byte. UTF-8 has emerged as the main character encoding for the World Wide Web and has solidified its position as the dominating standard for text encoding in the realm of computer science.

3.2.1.2 Remove new lines, numbers, and special characters

In this stage empty lines in the dataset are removed as a process to rearrange it, to decrease the processing time of the dataset. In order to exclude numerical values and special characters from a given text, an effective approach is to utilize a method referred to as "regular expression substitution." Regular expressions, sometimes known as regex, are formal mathematical representations that define a set of strings. In the given context, it is feasible to formulate a regular expression pattern that represents all characters that are not alphabetic or white spaces, including uppercase and lowercase letters from A to Z, as well as spaces and tabs. Consequently, every occurrence of this specific pattern is replaced with a null string, thereby effectively removing all numeric values and special characters from the provided text. The technology adopted in this process selectively preserves only the alphabetic characters and spaces in the resulting text. This approach is commonly utilized for the purposes of text modification and analysis.

3.2.2 Tokenization

Tokenization is a fundamental text preparation technique utilized in the domain of NLP. Its purpose is to divide a given text into smaller components known as tokens. These tokens have the ability to include words, phrases, or symbols, with each one representing significant aspects of text and playing a fundamental role in many NLP tasks such as text analysis, sentiment analysis, machine translation, and text production. During the initial phase, is often provided, which consists of one or more text documents in diverse forms. Tokenization is a process that takes place at various discrete levels, with the most prevalent being the phrase or word

level. At the lexical level, the text is methodically segmented into discrete units known as words, wherein each word is considered as an independent token. Word-level tokenization is of significant importance in NLP applications such as part-of-speech tagging, named entity recognition, and machine translation. This process serves as a fundamental step in facilitating the execution of these tasks. Algorithm (3.1) the steps of the tokenization process.

Algorithm (3.1) Tokenization process

Algorithm: tokenization process

Input: text (read dataset)

Output: tokens (a list of word tokens)

Begin

- Initialize an empty list to store tokens

tokens = []

- Initialize an empty string to store the current token

current_token = ""

- Iterate through each character in the text

for character in text:

- Check if the character is a letter or digit

if character.isalnum():

- If the character is a letter or digit, add it to the current token

current_token = character

else:

- If the character is not a letter or digit

if current_token:

- If the current token is not empty, add it to the list of tokens

tokens.append(current_token)

- Reset the current token to an empty string

current_token = ""

- Check if the character is not a space

if character != " ":

- If the character is not a space, add it as a separate token

tokens.append(character)

- Check if there's any remaining token in current_token

if current_token:

- If there's a remaining token, add it to the list of tokens

tokens.append(current_token)

End

3.2.3 Missing Random Character Generation

The labeling process has been done by taking each token and choosing a random location within the length of that token. This location will be replaced with a missing character and the character with this location will be the target class for this token.

3.2.4 Tokens Augmentation

The tokens in whole the dataset will be computed in order to know each token how many times repeated in the dataset the token with less twenty times repetitions will be duplicated many times until the repetition number is equal to thirty because most of the words repeated from twenty to forty times for ensure balancing dataset.

3.2.5 Text encoding

The notion of text encoding within the realm of deep learning refers to the process of converting unstructured textual data into a numerical format that is appropriate for use as input in neural networks and other machine learning models. The utilization of encoding is crucial due to the necessity of deep learning models to process numerical input. Text encoding strategies frequently utilized in the field of NLP involve several techniques, including the utilization of word embeddings. These embeddings enable the representation of words as vectors in high-dimensional spaces, allowing for the capture of semantic relationships. Furthermore, one-hot encoding is a widely employed method utilized for the purpose of representing words as binary vectors. These encodings enhance the potential of deep learning models to acquire and extract

patterns, relationships, and semantic meaning from textual data, enabling them to analyze and understand human language.

3.2.6 Reshape

In a deep neural network, the reshape stage is a transformation step where the shape of the data is changed without altering its content. It's commonly used to prepare the data for subsequent layers in the network. This operation is particularly useful when transitioning between different layer types or when the input shape of one layer doesn't match the output shape of the previous layer.

For instance, if you have a convolutional layer that outputs a 3D tensor (e.g., width \times height \times channels), but the next layer, say a fully connected layer, requires a 1D tensor as input, you would use a reshape operation to flatten the 3D tensor into a 1D tensor.

3.2.7 Normalization

Normalization is a crucial data preprocessing technique utilized in the field of deep learning and machine learning. The objective of this process is to adjust and normalize the input features, so guaranteeing that they are confined within a consistent range. The range is sometimes delineated as including values within the interval of 0 to 1, or alternatively, as having a mean of 0 and a standard deviation of 1. The process of normalization is of utmost importance in the training of models, since it ensures that each input variable makes an equitable contribution. Its purpose is to mitigate the dominance of larger-scale features in the learning process and enhance the speed of convergence during training. The utilization of regularization techniques is of utmost importance in the

domain of deep neural networks, as it plays a vital role in improving the robustness of the training process and enhancing the model's ability to effectively generalize to new data examples. Two commonly employed normalization techniques are the Z-score scaling equation (2.1).

Various strategies are utilized to modify the data in order to ensure that it conforms to predetermined ranges, while also preserving its relative relationships.

3.2.8 Long Short-Term Memory (LSTM)

Using a LSTM neural network to predict missing characters in text involves several key steps. It begins by preparing a dataset that includes text sequences with intentionally omitted characters and their corresponding complete versions. These characters are then encoded into numerical representations, often using one-hot encoding. Input-output pairs are created, with the input being the sequence with the missing character and the output being the same sequence with the missing character filled in.

The model architecture consists of an encoder and a decoder. During training, the model is trained using a loss function, such as cross-entropy, which measures the dissimilarity between the predicted character probabilities and the actual characters in the output sequence. Backpropagation through time (BPTT) and optimization techniques like the Adam optimizer are used to update the model's weights.

For inference, the text sequence is input into a text sequence with a missing character to the trained model. The encoder processes the input, and the decoder generates a probability distribution over characters for the

missing position. You sample from this distribution to predict the missing character probabilistically and use it to fill in the gap.

Hyperparameter tuning involves experimenting with different settings to optimize the model's performance. Once the model performs well on test data, it can be deployed for real-world applications where predicting missing characters in text is required. This approach leverages the sequence-to-sequence capabilities of LSTM networks for tasks like text correction and language modeling. Algorithm (3.2) shows the LSTM steps.

Algorithm (3.2) The steps of LSTM

<p>Algorithm: LSTM model Input: coded text data Output: trained weight</p>
<p>Begin</p> <ul style="list-style-type: none"> - Initialize the previous hidden state and cell state <ul style="list-style-type: none"> $h_{prev} = 0$ $c_{prev} = 0$ - Define the parameters of the LSTM cell, including the input weights, recurrent weights and biases <ul style="list-style-type: none"> $W_{ii}, W_{hi}, b_{ii}, b_{hi} = 0$ $W_{if}, W_{hf}, b_{if}, b_{hf} = 0$ $W_{ig}, W_{hg}, b_{ig}, b_{hg} = 0$ $W_{io}, W_{ho}, b_{io}, b_{ho} = 0$ -Input gate <ul style="list-style-type: none"> function sigmoid(x): return $1 / (1 + \exp(-x))$ $i_t = \text{sigmoid}(W_{ii} * x_t + W_{hi} * h_{prev} + b_{ii} + b_{hi})$ -Forget gate <ul style="list-style-type: none"> $f_t = \text{sigmoid}(W_{if} * x_t + W_{hf} * h_{prev} + b_{if} + b_{hf})$ -Input modulation gate <ul style="list-style-type: none"> $g_t = \tanh(W_{ig} * x_t + W_{hg} * h_{prev} + b_{ig} + b_{hg})$ -Cell state update <ul style="list-style-type: none"> $c_t = f_t * c_{prev} + i_t * g_t$ -Output gate <ul style="list-style-type: none"> $o_t = \text{sigmoid}(W_{io} * x_t + W_{ho} * h_{prev} + b_{io} + b_{ho})$ -Hidden state update <ul style="list-style-type: none"> $h_t = o_t * \tanh(c_t)$ <p>End</p>

3.2.9 Recurrent Neural Network (RNN)

Using a RNN to predict a missing character in text is a sequence-to-sequence prediction task. The basic idea is to train an RNN on a dataset of text sequences with missing characters and their corresponding correct sequences and then use the trained model to predict missing characters in new sequences. Here's a simplified outline of how you can approach this task:

➤ Model Architecture:

- Design an RNN-based architecture for sequence-to-sequence prediction. You can use a simple RNN, a more advanced LSTM, or a Gated Recurrent Unit (GRU) depending on your requirements.
- The model should have an encoder-decoder structure. The encoder processes the input sequence with the missing character, and the decoder generates the missing character.
- Use BPTT or any suitable optimization algorithm (e.g., Adam) to update the model's weights.

Algorithm (3.3) The steps of RNN

Algorithm: Recurrent Neural Network (RNN) Input: sequence of input vectors Output: sequence of hidden states
Begin - Initial hidden state $h_{prev} = 0$ - Define RNN parameters Weights and bias $W_{xh}, W_{hh}, b_h = 0$ - Iterate through the input sequence for t in range(sequence_length): - Input vector at time t $x_t = input_sequence[t]$ - Hidden state update $h_t = \tanh(W_{xh} * x_t + W_{hh} * h_{prev} + b_h)$ - Store the hidden state for later use or output $output_sequence[t] = h_t$ - Update the previous hidden state for the next iteration $h_{prev} = h_t$ - The final output_sequence contains the hidden states for each time step End

3.2.10 Generative Adversarial Network (GAN)

Using a GAN for predicting a missing character in text is an unconventional approach. GANs are typically used for generating data, such as images, rather than predicting missing elements within existing data. However, it's possible to adapt GANs for such a task, although it

may not be the most efficient or straightforward approach. Here's a simplified outline of how to utilizing GAN for this purpose:

1- Data Preparation:

Prepare a dataset of text sequences with missing characters and their corresponding complete versions.

2- GAN Architecture:

Design a GAN with generator and discriminator. The generator takes text sequences with missing characters as input and tries to generate missing characters, while the discriminator aims to distinguish between real and generated missing characters.

3- Hyperparameter Tuning:

Experiment with various hyperparameters, such as the architecture of the GAN, learning rates, and training epochs, to optimize the model's performance.

Hyperparameter tuning in the context of GANs involves identifying the optimal combination of hyperparameter values that maximize the performance of the GAN model. GANs are a class of deep learning models including a generator and a discriminator, trained in an adversarial manner. Hyperparameters refer to the predetermined configuration settings of a model that are not learned during the training process but need to be established prior to training commencement. Optimizing these hyperparameters is essential for attaining peak performance and producing synthetic data of superior quality.

The following are often adjusted hyperparameters in GANs:

1. Learning rate: The learning rate governs the magnitude of the increments made throughout the optimization procedure. This hyperparameter is crucial and has a substantial effect on the training

of both generator and discriminator. The initial values of the learning rate are Adam (0.0002, 0.9) represents the Adam optimizer with a learning rate of 0.0002 and a first-moment exponential decay rate (beta1) of 0.9.

2. The number of layers and neurons in generator and discriminator architecture is crucial. The intricacy of the model can impact its capacity to produce authentic samples. The generator has five layers while the discriminator has four layers.
3. Activation functions, such as ReLU, Leaky ReLU, and tanh, have a significant impact on the learning process and the quality of generated samples in both generator and discriminator. In the generator the alpha parameter value = 0.2 and in discriminator =0.3.
4. Batch size: The quantity of samples utilized in each iteration of training can impact the stability of the training process. Reducing the number of batches used in the learning process can result in increased noise, whilst using bigger batch sizes may necessitate more memory. The training batch size is equal to 64.
5. Training duration: The duration of training is determined by the number of training epochs and the stopping criteria, which are important hyperparameters. Insufficient training epochs can cause underfitting, whereas excessive training epochs can result in overfitting. 100 epochs have been used for training.
6. Input noise: The generator's diversity of generated samples is heavily influenced by the input noise vector. Modulating the magnitude and dispersion of the input noise can significantly influence the caliber and diversity of the generated outputs. The input vector for the generator (12) while the discriminator (12,512).

7. Weight initialization: The technique employed to set the initial values of the weights in the neural network might impact the rate at which the model reaches convergence. Typical approaches involve random initialization and Xavier/Glorot initialization. Random initial weight has been used.
8. Selection of optimizer: The selection of an optimizer, such as Adam or SGD, can have an impact on speed and stability of the training process. Various optimizers possess distinct hyperparameters that may require adjustment. Such as Adam values.

Algorithm (3.4) The steps of GAN

Algorithm: Generative Adversarial Network (GAN)

Input: coded text data

Output: trained weight

Begin

- Generator Network
 - function build_generator:()
- Define generator architecture (e.g., a neural network with transposed convolutions)
 - generator =
 - return generator
- Discriminator Network
 - function build_discriminator:()
- Define discriminator architecture (e.g., a neural network with convolutions)
 - discriminator = 0
 - return discriminator
- Compile the Discriminator
 - discriminator = build_discriminator()
 - discriminator.compile(loss='binary_crossentropy',
 - optimizer='adam', metrics=['accuracy'])
- Freeze the Discriminator during GAN training
 - discriminator.trainable = False
- Combined GAN Model (Generator + Discriminator)
 - function build_gan(generator, discriminator):
- Input for the generator
 - gan_input = 0
 - generated_data = generator(gan_input)
 - gan_output = discriminator(generated_data)
 - gan = Model(gan_input, gan_output)
 - gan.compile(loss='binary_crossentropy', optimizer='adam')

return gan

```
- Training Loop
  function train_gan(generator, discriminator, gan, real_data,
    epochs, batch_size):
    for epoch in range(epochs):
- Generate fake data using the generator
      generated_data =
        generator.predict(random_noise(batch_size))
- Label real and fake data
      real_labels = 1(batch_size, 1)
      fake_labels = 0(batch_size, 1)
- Train the discriminator on real data
      d_loss_real = discriminator.train_on_batch(real_data,
        real_labels)
- Train the discriminator on fake data
      d_loss_fake = discriminator.train_on_batch(generated_data,
        fake_labels)
- Calculate total discriminator loss
      d_loss = 0.5 * add(d_loss_real, d_loss_fake)
- Generate new random noise
      noise = random_noise(batch_size)
- Train the generator to fool the discriminator
      g_loss = gan.train_on_batch(noise, ones((batch_size, 1)))
- Print progress
      print("Epoch {epoch+1}, D Loss: {d_loss[0]}, G Loss:
        {g_loss}")
```

End

CHAPTER FOUR

RESULTS AND

DISCUSSION

4.1 Review

The results of the proposed system have been presented in this chapter. The Codex Sinaiticus dataset has been preprocessed and provided for the proposed system to train and test the model.

4.2 Hardware requirement

Python programming language was used to implement the system while the environment was Windows 11, 8 GB RAM, Corei5 10th 2.50 GHz generation processor, and Nvidia G-force GTX 1650 Ti.

4.3 Dataset

Two datasets have been used for training and testing the suggested models. These datasets are:

4.3.1 Codex Sinaiticus

The Codex Sinaiticus, a manuscript dating back to the mid-fourth century, is a significant artifact as it houses the first known complete copy of the New Testament of the Christian Bible. The handwritten material is written in the Greek language. The New Testament is written in the native vernacular language of the time, known as koine, whereas the Old Testament is presented in the translation referred to as the Septuagint, which was embraced by early Greek-speaking Christians. The Codex contains extensive annotations made by a number of early correctors, which significantly affect the text of both Septuagint and New Testament.

The Codex Sinaiticus holds great importance in the realm of reconstructing the original text of the Christian Bible, as well as in the fields of Bible history and Western book-making [83].

4.3.2 Argonautica, 3

This dataset focuses on the semantics of about twenty-five Latin and Ancient Greek words (nouns, verbs, and adjectives) that belong to the semantic field SEA. Four writings make up the corpus:

1. De bello Gallico (Caesar) and Aeneid 1-6 (Vergil) in Latin;
2. Histories 1-2 (Herodotus) in Greek and Argonautica (Apollonius Rhodius) in Greek. After manually annotating the texts using the annotation program INCEPTION, the data was extracted [84].

4.4 Case study

The dataset trained and tested on the proposed model is as follows:

4.4.1 Dataset preprocessing

The dataset was preprocessed using the following steps:

4.4.1.1 UTF-8 Encoding

UTF-8 encoding is a variable-width character encoding capable of encoding all valid character code points in Unicode using one to four one-byte (8-bit) code units representing text in various writing systems that are prevalent worldwide.

4.4.1.2 Removing numbers and special characters

Removing all numeric values and special characters from the provided text.

4.4.1.3 Remove newline symbol

The new line symbol refers to the end of the line and down to the new line in the text removing this symbol will make the text dataset in one paragraph. Figure (4.1) shows the dataset after removing ‘\n’ symbol.

αβιμελεχ ουκ εγνω τις εποιησεν το πραγμα τουτο ουδε συ μοι απηγγει
λας ουδε εγω ηκουσα αλλ η ση μερον και ελαβε αβρααμ προβατα και
μοσχους και εδωκεν τω αβιμε λεχ και διεθεντο αμφοτεροι διαθη κην και
εστησε αβρααμ επτα αμνα δας των προβατων μο τω φρεατι του ορ κου και
επεκαλε σατο εκει το ονο μα του κυ θς αιω νιος παρωκησε δε αβρααμ
εν τη γη των φυλιςτιειμ ημερας πολλας και εγενετο μετα τα ρηματα
ταυτα επειρασενεπειραζεν ο θς τον αβρααμ και ειπεν προς αυτον αβρααμ
αβρααμ ο δε ειπεν ιδου εγω και ειπεν λαβε το υιον σου τον αγαπητον
ον ηγαπη σατον ισαακ και πορευθητι εις τη γην την υψηλην και
ανενεγκε αυ τον εκει εις ολοκαρ πωσιν εφ εν τω ορειωνων αν σοι εγω
δε και το παι δαρion διελευσο μεθα εως ωδε και προσκυνησαντες
αναστρεψωμεν προς υμας ελαβεν δε αβρααμ τα ξυλα της ολοκαρπωσε ως
και επεθηκεν ισαακ τω υιω αυ του ελαβεν δε με τα χειρα και το πυρ
και την μαχαιραν και επορευθησαν οι δυο αμα ειπεν δε ισαακ προς αβρααμ
τον πατε ρα αυτου ειπας πατερ ο δε ειπεν τι εστιν τε'

Figure (4.1) Removing '\n' from the dataset

4.4.1.4 Tokenization

The dataset is then tokenized according to the space character. Figure (2.4) shows the tokenization process.

```
'αβιμελεχ', 'ουκ', 'εγνω', 'τις', 'εποιησεν', 'το', 'πραγμα',
'τουτο', 'ουδε', 'συ', 'μοι', 'απηγγει', 'λας', 'ουδε', 'εγω',
'ηκουσα', 'αλλ', 'η', 'ση', 'μερον', 'και', 'ελαβε', 'αβρααμ',
'προβατα', 'και', 'μοσχους', 'και', 'εδωκεν', 'τω', 'αβιμε', 'λεχ',
'και', 'διεθεντο', 'αμφοτεροι', 'διαθη', 'κην', 'και', 'εστησε',
'αβρααμ', 'επτα', 'αμνα', 'δασ', 'των', 'προβατων', 'μο', 'τω',
'φρεατι', 'του', 'ορ', 'κου', 'και', 'επεκαλε', 'σατο', 'εκει',
'το', 'ονο', 'μα', 'του', 'κυ', 'θς', 'αιω', 'νιος', 'παρωκησε',
'δε', 'αβρααμ', 'εν', 'τη', 'γη', 'των', 'φυλιςτιειμ', 'ημερας',
'πολλας', 'και', 'εγενετο', 'μετα', 'τα', 'ρηματα', 'ταυτα',
'επειρασενεπειραζεν', 'ο', 'θς', 'τον', 'αβρααμ', 'και', 'ει',
'πεν', 'προς', 'αυτον', 'αβρααμ', 'αβρααμ', 'ο', 'δε', 'ειπεν',
'ιδου', 'εγω', 'και', 'ειπεν', 'λαβε', 'το', 'υιον'
```

Figure (4.2) A sample tokenization process

4.4.1.5 Compute language characters

This process is necessary to know the set of target classes for the prediction process.

```
41
', 'α', 'β', 'γ', 'δ', 'ε', 'ζ', 'η', 'θ', 'ι', 'κ', 'λ', 'μ', 'ν', 'ξ', 'ο', 'π', 'ρ', 'τ', 'υ', 'φ', 'χ', 'ψ', 'ω', 'ι',
'υ', 'φ', 'ζ', 'θ', 'α', 'η', 'ο', 'ο', 'η', 'ι', 'α', 'η', 'ι', 'υ', 'ω'
```

Figure (4.3) The character of language indicates 41 characters

4.4.1.6 Generate missing character

In this process a character from each token in the dataset has been removed and replaced with '\$' character and the removed character

represent the target class for the prediction process. Figure (4.4) generates missing values and their classes.

```
x_data: ['$βιμελεχ', 'ου$', 'ε$νω', 'τ$ς', 'εποι$εν', 'πρα$μα',
'tο$το', 'ου$ε', 'μο$', '$πηγγει', 'λ$ς', 'ουδ$', 'ε$ω',
'ηκου$$', '$λλ', 'μ$ρον', 'κ$ι', 'ελ$βε', 'α$ρααμ', 'προ$ατα',
'κα$', '$ο$χο$υς', '$αι', 'ε$ωκεν', '$βιμε', 'λ$χ', '$ιεθεντο',
'$μφοτεροι', 'δι$θη', 'κ$ν', 'ε$τ$ε', 'αβραα$', 'επ$τ$', 'α$να',
'δ$ς', 'τ$ω$', 'προβ$των', 'φρε$τι', 'τ$υ', '$ου', 'επεκ$λε',
'c$το', 'εκε$', 'ο$ο', 'το$', '$ιω', 'ν$ο$ς', 'π$ρωκησε',
'αβ$ααμ', '$ων', 'φυλι$τι$ιμ', 'ημερ$ς', 'πολλ$ς', 'ε$ενετο',
'μετ$', 'ρ$ματα', 'τα$τα', 'τ$ν', 'π$ν', 'πρ$ς', '$υτον',
'ε$πεν', 'ι$ου', '$γω', 'ειπε$', 'λ$βε', 'υ$ον', 'c$υ', '$ον',
'α$α', 'π$τον', 'ηγ$πη', 'c$ctον', 'ι$αα$', 'πορ$υθητι', '$ις',
'$ην', 'τ$η$', 'υψη$ην', '$νενεγκε', 'ε$ει', 'ε$ς', 'ολοκ$ρ',
'πω$ν', 'ορ$ων', 'co$', 'εγ$', 'π$ι', 'δ$ριον', '$ιελευco',
'μεθ$', '$ως', 'ω$ε', '$ρο$ς', 'υμ$ς', 'ελ$βεν', 'ξυλ$', '$ης',
'ολοκ$ρωσε', 'επεθ$κεν']

y_data: ['α', 'κ', 'γ', 'ι', 'η', 'γ', 'υ', 'δ', 'ι', 'α', 'α',
'ε', 'γ', 'α', 'α', 'ε', 'α', 'α', 'β', 'β', 'ι', 'μ', 'κ', 'δ',
'α', 'ε', 'δ', 'α', 'α', 'η', 'η', 'μ', 'α', 'μ', 'α', 'ν', 'α',
'α', 'ο', 'κ', 'α', 'α', 'ι', 'ν', 'υ', 'α', 'ι', 'α', 'ρ', 'τ',
'ε', 'α', 'α', 'γ', 'α', 'η', 'υ', 'ο', 'ε', 'ο', 'α', 'ι', 'δ',
'ε', 'ν', 'α', 'ι', 'ο', 'τ', 'γ', 'η', 'α', 'α', 'κ', 'ε', 'ε',
'γ', 'ν', 'λ', 'α', 'κ', 'ι', 'α', 'ι', 'ε', 'ι', 'ω', 'α', 'α',
'δ', 'α', 'ε', 'δ', 'π', 'α', 'α', 'α', 'τ', 'α', 'η']
```

Figure (4.4) Sample of missing values and their classes

4.4.1.7 Compute target class characters

In this process, the target class characters are computed to know the class target number. Figure (4.5) shows the target class list. The list consists of 26 characters.

```
26
['α', 'β', 'γ', 'δ', 'ε', 'ζ', 'η', 'θ', 'ι', 'κ', 'λ', 'μ', 'ν',
'ξ', 'ο', 'π', 'ρ', 'τ', 'υ', 'φ', 'χ', 'ψ', 'ω', 'ϊ', 'ϋ', 'ς']
```

Figure (4.5) Target class character list appointed to 26 classes

4.4.1.8 Augmentation process

To train the model on fair data need to ensure data balancing there for every token repeated for less than 30 will be copied with his class number to 30. After this process, the dataset increased from 29365 to 47582 instances.

4.4.1.8 Text encoding

Convert dataset to numerical format.

4.4.1.9 Reshape

Reshaping each element to maximum length.

4.4.1.10 Normalization

Normalization process used to move the encoded character to 0-1 scope in order to move them to prediction models.

4.4.2 Prediction models

The dataset has been trained and tested three prediction models LSTM, RNN, and GAN, and results are listed as follows:

4.4.2.1 LSTM model

Five layers of LSTM has been used for training and testing the dataset. Figure (4.6) explains the summary of LSTM model layers.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 8, 128)	66560
dropout (Dropout)	(None, 8, 128)	0
lstm_1 (LSTM)	(None, 64)	49408
dropout_1 (Dropout)	(None, 64)	0
dense (Dense)	(None, 31)	2015

=====
Total params: 117,983
Trainable params: 117,983
Non-trainable params: 0

Figure (4.6) Summary of LSTM model layers

Figure (4.7) shows the training process of LSTM model.

```
Epoch 1/100
595/595 [=====] - 27s 27ms/step - loss: 2.4757 - accuracy: 0.2449 - val_loss: 2.1923 - val_accuracy: 0.3181 - lr: 0.0010
Epoch 2/100
595/595 [=====] - 15s 25ms/step - loss: 2.0795 - accuracy: 0.3736 - val_loss: 1.9362 - val_accuracy: 0.4369 - lr: 0.0010
Epoch 3/100
595/595 [=====] - 14s 24ms/step - loss: 1.8760 - accuracy: 0.4446 - val_loss: 1.7431 - val_accuracy: 0.4817 - lr: 0.0010
Epoch 4/100
595/595 [=====] - 14s 24ms/step - loss: 1.7426 - accuracy: 0.4789 - val_loss: 1.5831 - val_accuracy: 0.5130 - lr: 0.0010
Epoch 5/100
595/595 [=====] - 14s 24ms/step - loss: 1.6185 - accuracy: 0.5083 - val_loss: 1.4704 - val_accuracy: 0.5364 - lr: 0.0010
Epoch 6/100
595/595 [=====] - 14s 24ms/step - loss: 1.5070 - accuracy: 0.5356 - val_loss: 1.3357 - val_accuracy: 0.5792 - lr: 0.0010
Epoch 7/100
595/595 [=====] - 14s 24ms/step - loss: 1.4139 - accuracy: 0.5595 - val_loss: 1.2475 - val_accuracy: 0.6140 - lr: 0.0010
Epoch 8/100
595/595 [=====] - 15s 25ms/step - loss: 1.3194 - accuracy: 0.5892 - val_loss: 1.1825 - val_accuracy: 0.6193 - lr: 0.0010
Epoch 9/100
595/595 [=====] - 14s 24ms/step - loss: 1.2281 - accuracy: 0.6181 - val_loss: 1.1158 - val_accuracy: 0.6449 - lr: 0.0010
Epoch 10/100
595/595 [=====] - 15s 25ms/step - loss: 1.1535 - accuracy: 0.6391 - val_loss: 0.9424 - val_accuracy: 0.7195 - lr: 0.0010
Epoch 11/100
595/595 [=====] - 14s 24ms/step - loss: 1.0904 - accuracy: 0.6552 - val_loss: 0.8992 - val_accuracy: 0.7206 - lr: 0.0010
Epoch 12/100
595/595 [=====] - 14s 24ms/step - loss: 1.0148 - accuracy: 0.6808 - val_loss: 0.8187 - val_accuracy: 0.7467 - lr: 0.0010
Epoch 13/100
...
Restoring model weights from the end of the best epoch: 20.
595/595 [=====] - 15s 25ms/step - loss: 0.5375 - accuracy: 0.8324 - val_loss: 0.3921 - val_accuracy: 0.8923 - lr: 0.0010
```

Figure (4.7) Training process for LSTM model

Figure (4.8) shows training and validation accuracy while Figure (4.9) explains training and validation loss.

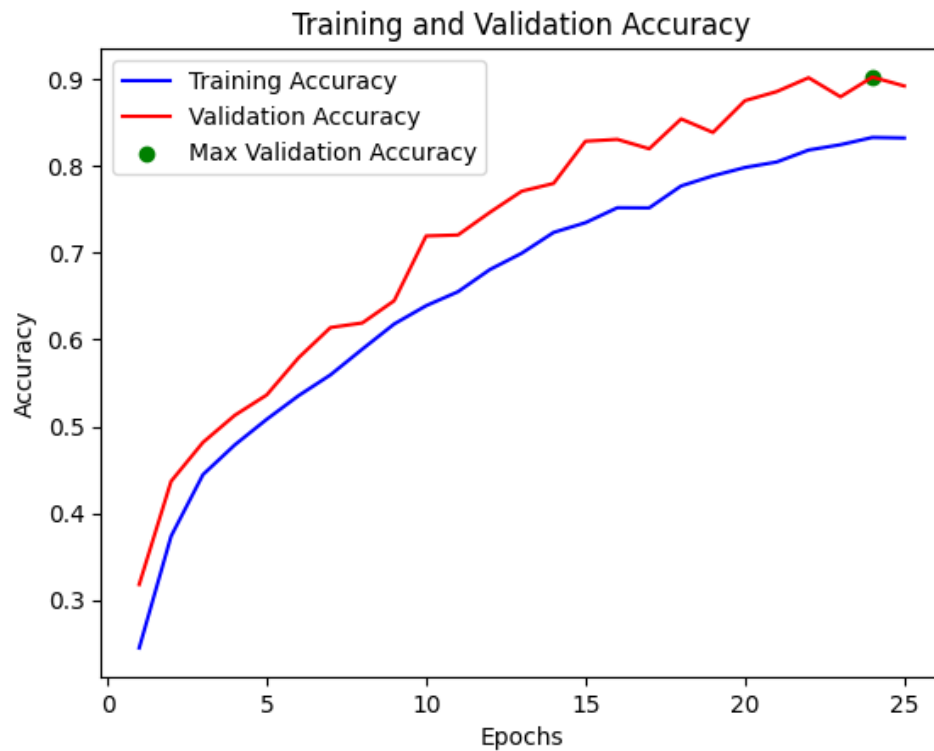


Figure (4.8) LSTM training and validation accuracy

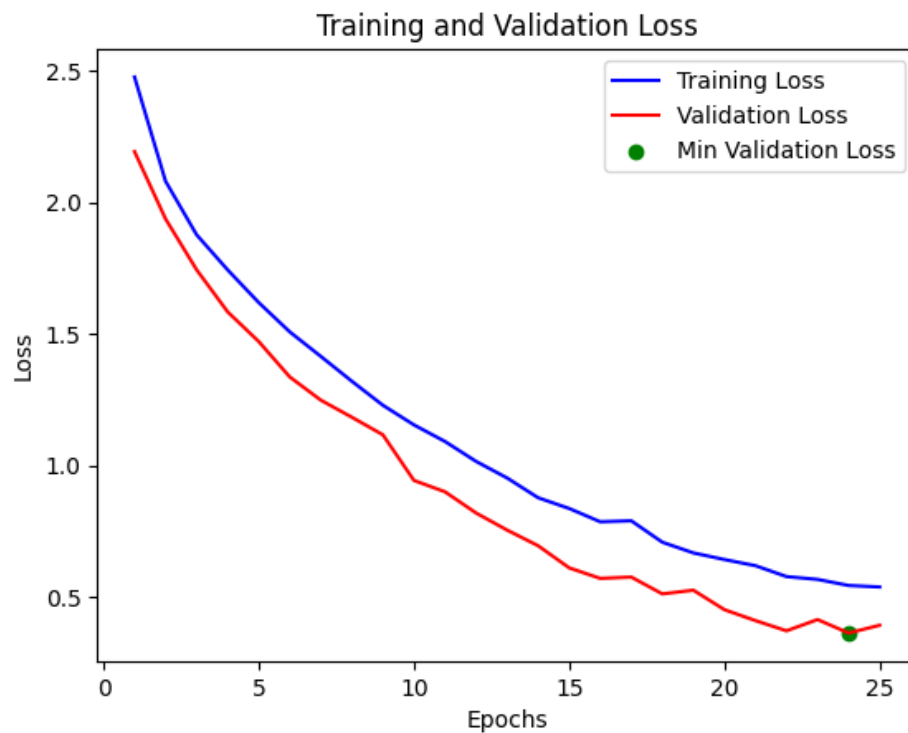


Figure (4.9) Training and validation loss

Table (4.1) lists the precision, recall, f1 score, and support measures.

Table (4.1) The precision, recall, f1 score, and support measures for the LSTM model

classes	precision	recall	f1-score	support
2	0.90	0.88	0.89	284
3	1.00	0.25	0.40	8
4	0.86	0.94	0.90	34
5	0.82	0.82	0.82	38
6	0.84	0.75	0.79	151
8	0.81	0.82	0.82	79
9	1.00	0.33	0.50	15
10	0.83	0.87	0.85	148
11	0.98	0.99	0.99	188
12	0.75	0.94	0.83	32
13	0.90	0.87	0.88	53
14	0.85	0.92	0.88	143
16	0.82	0.87	0.84	251
17	0.77	0.65	0.70	93
18	0.73	0.92	0.81	26
19	0.84	0.95	0.89	296
20	0.98	0.81	0.89	200
21	1.00	1.00	1.00	4
22	0.46	0.86	0.60	7
23	0.00	0.00	0.00	6
24	0.84	0.94	0.89	83
25	0.93	0.50	0.65	28
26	1.00	0.40	0.57	5
30	0.86	0.87	0.86	208
accuracy			0.86	2380
macro avg	0.82	0.76	0.76	2380
weighted avg	0.86	0.86	0.86	2380

Figure (4.10) shows the confusion matrix of the LSTM model.

Confusion Matrix

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	249	0	0	0	4	9	0	4	0	3	0	4	2	0	2	4	0	0	0	0	0	1	0	2
1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0
2	0	0	32	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	31	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
4	2	0	0	2	113	1	0	0	0	0	5	2	12	0	0	2	1	0	6	0	0	0	0	5
5	0	0	0	0	0	65	0	0	0	0	0	0	4	3	0	6	0	0	0	0	0	0	0	1
6	0	0	1	0	0	0	5	0	0	0	0	0	0	2	0	5	0	0	1	0	1	0	0	0
7	2	0	0	5	0	3	0	129	0	0	0	0	1	3	0	1	0	0	0	0	0	0	0	4
8	0	0	2	0	0	0	0	0	186	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	2	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	46	0	0	0	0	2	0	0	0	0	0	0	0	5
11	0	0	0	0	1	0	0	5	0	0	0	132	5	0	0	0	0	0	0	0	0	0	0	0
12	8	0	0	0	2	0	0	3	0	0	0	1	218	0	4	3	2	0	0	0	7	0	0	3
13	2	0	2	0	4	0	0	0	0	7	0	0	3	60	0	8	0	0	0	0	3	0	0	4
14	1	0	0	0	0	0	0	1	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	11	2	0	0	0	3	0	0	280	0	0	0	0	0	0	0	0
16	7	0	0	0	4	2	0	0	0	0	0	0	8	0	3	9	163	0	0	0	0	0	0	4
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	6	0	0	0	0	0
19	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	78	0	0	2
21	1	0	0	0	0	0	0	0	0	0	0	5	0	0	8	0	0	0	0	0	0	14	0	0
22	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0
23	3	0	0	0	4	0	0	0	0	0	0	10	2	3	0	0	1	0	0	0	4	0	0	181

Figure (4.10) LSTM model confusion matrix

4.4.2.2 RNN model

Another model was also trained and tested using the previous dataset. Figure (4.11) shows the summary of RNN layers.

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 8, 128)	16640
dropout_2 (Dropout)	(None, 8, 128)	0
simple_rnn_1 (SimpleRNN)	(None, 64)	12352
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 31)	2015

=====
Total params: 31,007
Trainable params: 31,007
Non-trainable params: 0

Figure (4.11) The summary of RNN layers

Figure (4.12) shows the training process of the RNN model.

```

Epoch 1/100
595/595 [=====] - 13s 14ms/step - loss: 2.2180 - accuracy: 0.3518 - val_loss: 1.8297 - val_accuracy: 0.4565 - lr: 0.0010
Epoch 2/100
595/595 [=====] - 8s 13ms/step - loss: 1.7788 - accuracy: 0.4746 - val_loss: 1.5213 - val_accuracy: 0.5348 - lr: 0.0010
Epoch 3/100
595/595 [=====] - 7s 12ms/step - loss: 1.5310 - accuracy: 0.5409 - val_loss: 1.2503 - val_accuracy: 0.6176 - lr: 0.0010
Epoch 4/100
595/595 [=====] - 6s 10ms/step - loss: 1.3490 - accuracy: 0.5916 - val_loss: 1.1199 - val_accuracy: 0.6759 - lr: 0.0010
Epoch 5/100
595/595 [=====] - 8s 14ms/step - loss: 1.1888 - accuracy: 0.6371 - val_loss: 0.8931 - val_accuracy: 0.7299 - lr: 0.0010
Epoch 6/100
595/595 [=====] - 6s 11ms/step - loss: 1.0569 - accuracy: 0.6738 - val_loss: 0.7300 - val_accuracy: 0.7845 - lr: 0.0010
Epoch 7/100
595/595 [=====] - 6s 10ms/step - loss: 0.9236 - accuracy: 0.7160 - val_loss: 0.6235 - val_accuracy: 0.8321 - lr: 0.0010
Epoch 8/100
595/595 [=====] - 6s 10ms/step - loss: 0.8706 - accuracy: 0.7327 - val_loss: 0.5362 - val_accuracy: 0.8502 - lr: 0.0010
Epoch 9/100
595/595 [=====] - 6s 10ms/step - loss: 0.7963 - accuracy: 0.7544 - val_loss: 0.5668 - val_accuracy: 0.8279 - lr: 0.0010
Epoch 10/100
595/595 [=====] - 6s 10ms/step - loss: 0.7281 - accuracy: 0.7750 - val_loss: 0.4788 - val_accuracy: 0.8634 - lr: 0.0010
Epoch 11/100
595/595 [=====] - 6s 11ms/step - loss: 0.6727 - accuracy: 0.7941 - val_loss: 0.4045 - val_accuracy: 0.8868 - lr: 0.0010
Epoch 12/100
595/595 [=====] - 6s 11ms/step - loss: 0.6514 - accuracy: 0.8006 - val_loss: 0.3730 - val_accuracy: 0.8973 - lr: 0.0010
Epoch 13/100
...
Epoch 29/100
594/595 [=====>.] - ETA: 0s - loss: 0.3770 - accuracy: 0.8943Restoring model weights from the end of the best epoch: 19.
595/595 [=====] - 6s 10ms/step - loss: 0.3768 - accuracy: 0.8943 - val_loss: 0.2004 - val_accuracy: 0.9524 - lr: 0.0010

```

Figure (4.12) The training process of the RNN model

Figure (4.13) shows training and validation accuracy for the RNN model while Figure (4.14) explains training and validation loss.

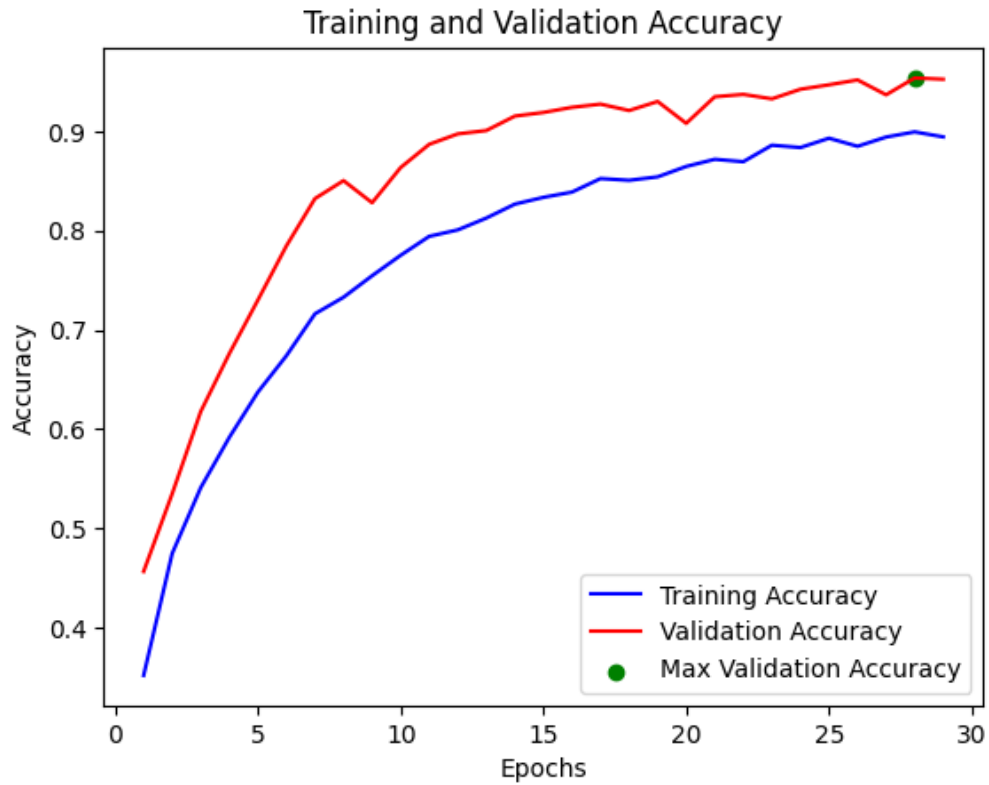


Figure (4.13) The training and validation accuracy for the RNN model

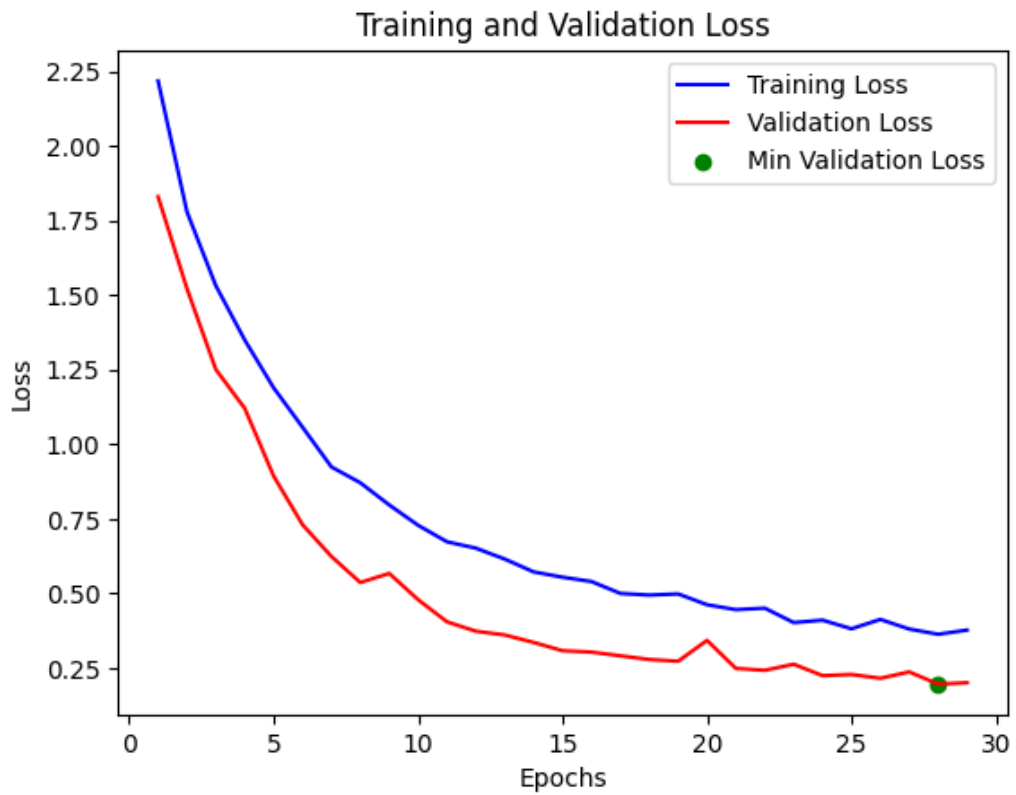


Figure (4.14) The training and validation loss for RNN model

Table (4.2) lists precision, recall, F1-score, and support measures for the RNN model.

Table (4.2) The precision, recall, F1-score, and support measure for the RNN model

classes	precision	recall	f1-score	support
2	0.88	0.94	0.91	284
3	1.00	1.00	1.00	8
4	0.86	0.94	0.90	34
5	0.75	0.95	0.84	38
6	0.98	0.75	0.85	151
8	0.88	0.95	0.91	79
9	1.00	0.47	0.64	15
10	0.89	0.95	0.92	148
11	0.99	0.99	0.99	188
12	1.00	1.00	1.00	32
13	0.85	0.94	0.89	53
14	0.96	0.91	0.94	143
16	0.94	0.94	0.94	251
17	0.97	0.83	0.90	93
18	0.88	0.88	0.88	26
19	0.92	0.96	0.94	296
20	0.99	0.92	0.95	200
21	1.00	1.00	1.00	4
22	0.88	1.00	0.93	7
23	1.00	0.67	0.80	6
24	0.89	0.96	0.92	83
25	0.77	0.82	0.79	28
26	1.00	1.00	1.00	5
30	0.93	0.95	0.94	208
accuracy			0.92	2380
macro avg	0.93	0.90	0.91	2380
weighted avg	0.93	0.92	0.92	2380

Figure (4.15) shows the confusion matrix for the RNN model.

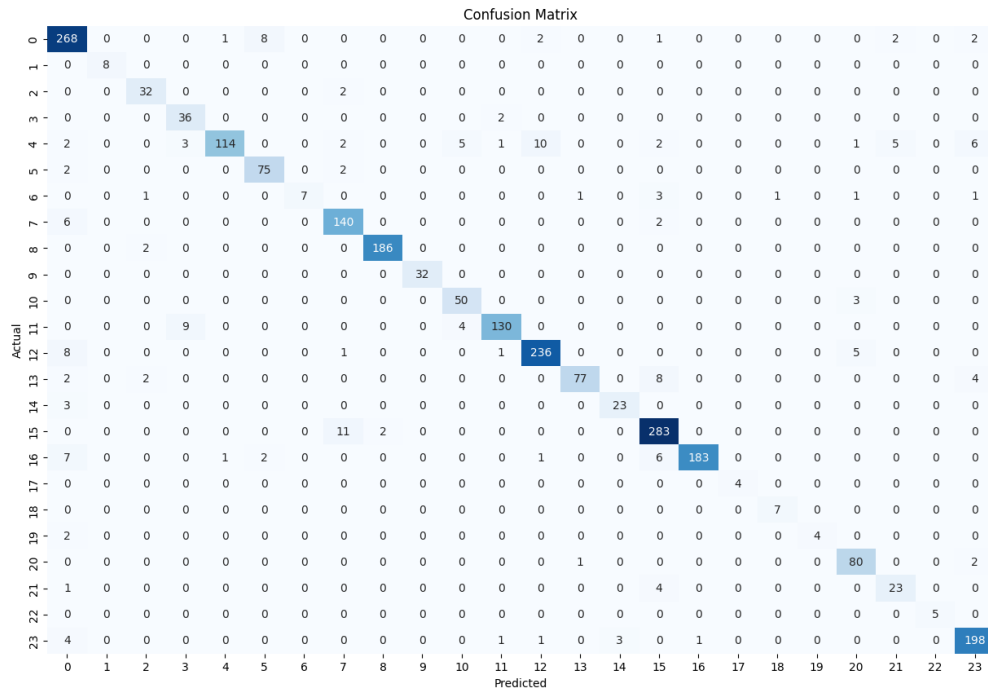


Figure (4.15) The confusion matrix for the RNN model

4.4.2.3 GAN model

The GAN model was also trained and tested using the previous dataset. Figure (4.16) shows the summary of GAN layers.

Layer (type)	Output Shape	Param #
dense_20 (Dense)	(None, 8, 128)	256
leaky_re_lu_8 (LeakyReLU)	(None, 8, 128)	0
batch_normalization_2 (Batch Normalization)	(None, 8, 128)	512
dense_21 (Dense)	(None, 8, 256)	33024
leaky_re_lu_9 (LeakyReLU)	(None, 8, 256)	0
batch_normalization_3 (Batch Normalization)	(None, 8, 256)	1024
dense_22 (Dense)	(None, 8, 512)	131584
=====		
Total params: 166,400		
Trainable params: 165,632		
Non-trainable params: 768		

Figure (4.16) The summary of GAN layers

Figure (4.17) shows training process of the GAN model.

```

595/595 [=====] - 28s 42ms/step - loss: 1.5246 - accuracy: 0.5512 - val_loss: 1.0077 - val_accuracy: 0.7030 - lr: 2.0000e-04
Epoch 2/100
595/595 [=====] - 25s 42ms/step - loss: 0.8256 - accuracy: 0.7530 - val_loss: 0.6570 - val_accuracy: 0.7994 - lr: 2.0000e-04
Epoch 3/100
595/595 [=====] - 23s 38ms/step - loss: 0.5508 - accuracy: 0.8384 - val_loss: 0.4926 - val_accuracy: 0.8429 - lr: 2.0000e-04
Epoch 4/100
595/595 [=====] - 24s 41ms/step - loss: 0.4239 - accuracy: 0.8819 - val_loss: 0.3854 - val_accuracy: 0.8925 - lr: 2.0000e-04
Epoch 5/100
595/595 [=====] - 23s 39ms/step - loss: 0.3590 - accuracy: 0.9017 - val_loss: 0.3628 - val_accuracy: 0.9037 - lr: 2.0000e-04
Epoch 6/100
595/595 [=====] - 24s 40ms/step - loss: 0.3161 - accuracy: 0.9161 - val_loss: 0.3630 - val_accuracy: 0.8917 - lr: 2.0000e-04
Epoch 7/100
595/595 [=====] - 25s 42ms/step - loss: 0.2836 - accuracy: 0.9242 - val_loss: 0.3109 - val_accuracy: 0.9200 - lr: 2.0000e-04
Epoch 8/100
595/595 [=====] - 29s 48ms/step - loss: 0.2722 - accuracy: 0.9297 - val_loss: 0.3267 - val_accuracy: 0.9047 - lr: 2.0000e-04
Epoch 9/100
595/595 [=====] - 23s 39ms/step - loss: 0.2468 - accuracy: 0.9387 - val_loss: 0.2404 - val_accuracy: 0.9354 - lr: 2.0000e-04
Epoch 10/100
595/595 [=====] - 22s 36ms/step - loss: 0.2346 - accuracy: 0.9413 - val_loss: 0.2533 - val_accuracy: 0.9341 - lr: 2.0000e-04
Epoch 11/100
595/595 [=====] - 25s 42ms/step - loss: 0.2225 - accuracy: 0.9469 - val_loss: 0.2198 - val_accuracy: 0.9512 - lr: 2.0000e-04
Epoch 12/100
595/595 [=====] - 23s 38ms/step - loss: 0.2095 - accuracy: 0.9512 - val_loss: 0.2885 - val_accuracy: 0.9426 - lr: 2.0000e-04
Epoch 13/100
595/595 [=====] - 20s 34ms/step - loss: 0.2007 - accuracy: 0.9525 - val_loss: 0.2485 - val_accuracy: 0.9383 - lr: 2.0000e-04
...
Epoch 99/100
595/595 [=====] - 23s 39ms/step - loss: 0.0914 - accuracy: 0.9810 - val_loss: 0.0989 - val_accuracy: 0.9807 - lr: 1.0000e-04
Epoch 100/100
595/595 [=====] - 23s 38ms/step - loss: 0.0922 - accuracy: 0.9805 - val_loss: 0.1025 - val_accuracy: 0.9823 - lr: 1.0000e-04

```

Figure (4.17) The training process of the GAN model

Figure (4.18) shows training and validation accuracy of the GAN model and Figure (4.19) shows training and validation loss.

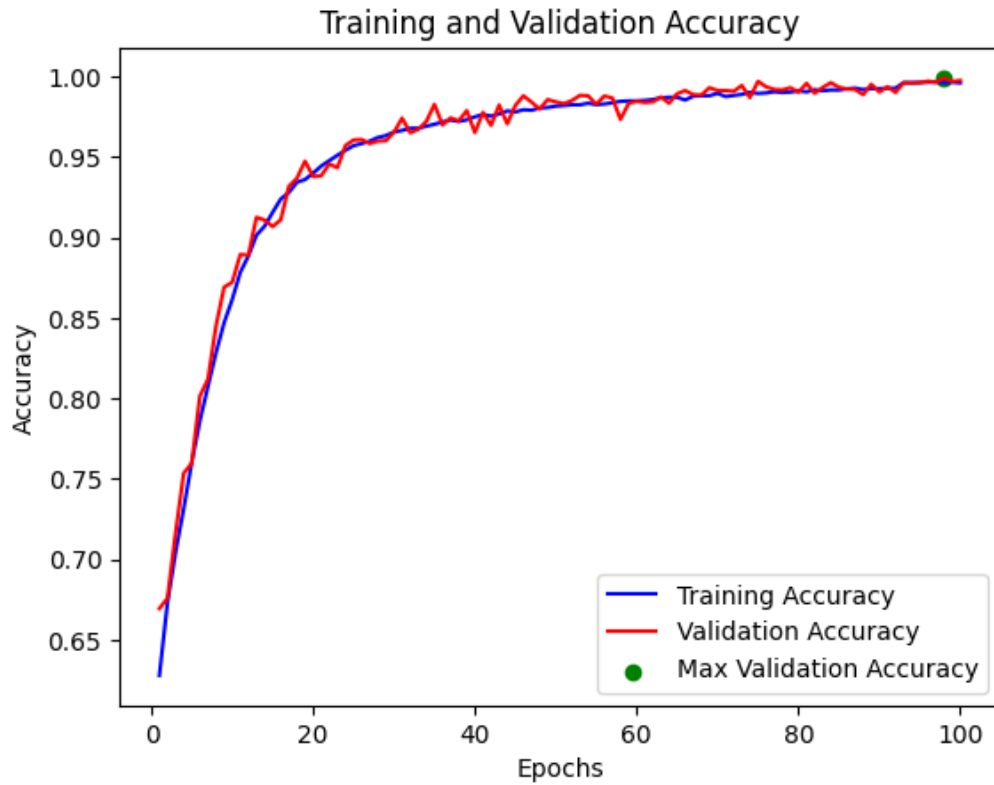


Figure (4.18) The training and validation accuracy of the GAN model

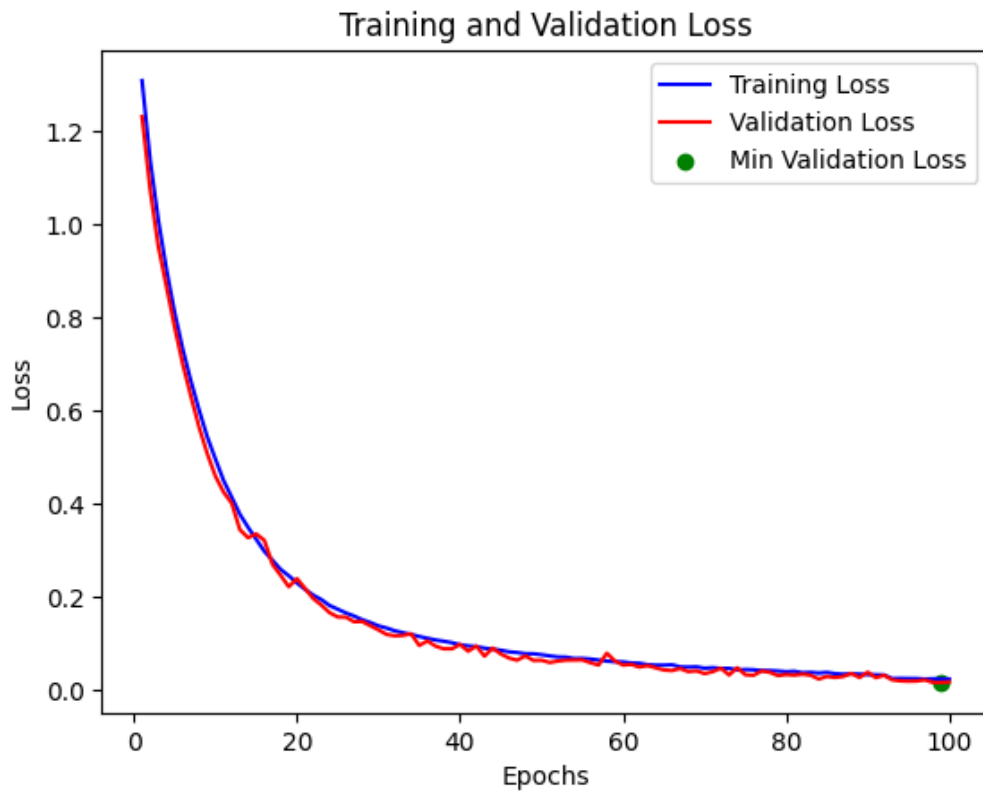


Figure (4.19) The training and validation loss of GAN model

Table (4.3) lists the precision, recall, F1-score, and support measures for the GAN model.

Table (4.3) The precision, recall, F1-score, and support measure for GAN model

classes	precision	recall	f1-score	support
2	0.96	0.94	0.95	284
3	1.00	1.00	1.00	8
4	0.97	0.82	0.89	34
5	1.00	1.00	1.00	38
6	0.95	0.92	0.94	151
8	0.97	0.97	0.97	79
9	1.00	0.60	0.75	15
10	0.98	1.00	0.99	148
11	0.94	1.00	0.97	188
12	1.00	0.84	0.92	32
13	1.00	1.00	1.00	53
14	0.93	0.97	0.95	143
16	0.93	0.94	0.94	251
17	0.97	0.98	0.97	93
18	1.00	0.96	0.98	26
19	0.95	0.99	0.97	296
20	0.99	0.98	0.99	200
21	1.00	1.00	1.00	4
22	0.88	1.00	0.93	7
23	1.00	1.00	1.00	6
24	0.99	0.96	0.98	83
25	1.00	0.93	0.96	28
...				
accuracy			0.96	2380
macro avg	0.98	0.95	0.96	2380
weighted avg	0.96	0.96	0.96	2380

Figure (4.20) shows the confusion matrix for GAN model.

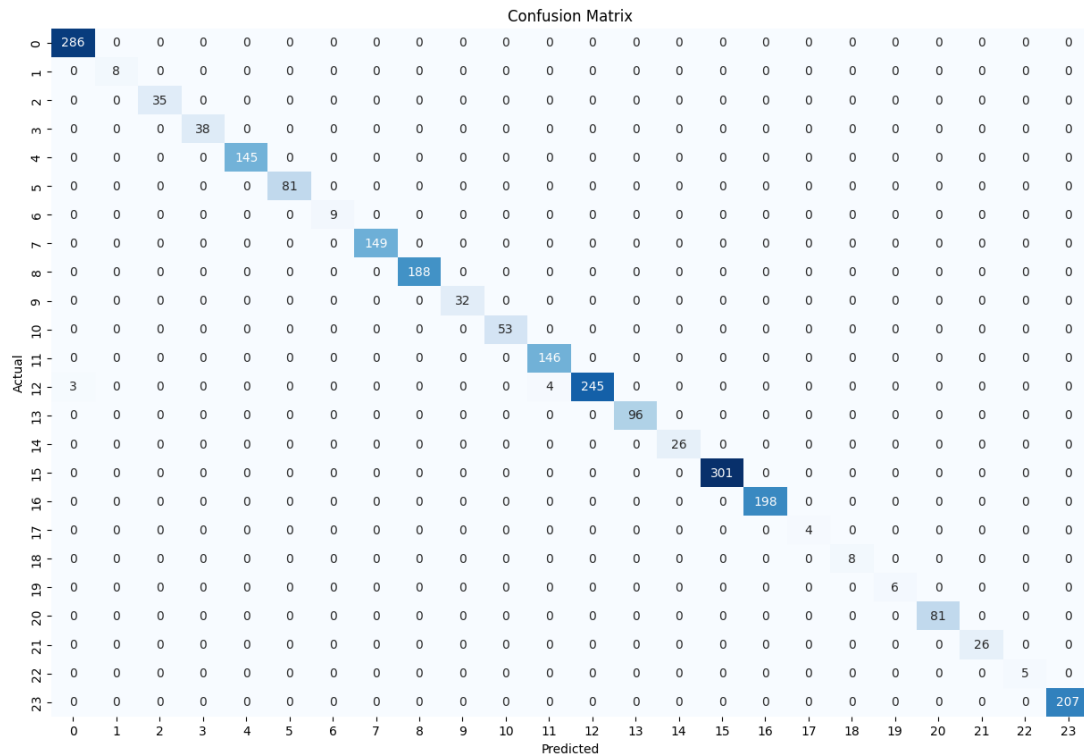


Figure (4.20) The confusion matrix for the GAN model

From Chapter One some related works that used accuracy as a performance measure were used to make a comparison with the obtained result from the proposed model.

Table (4.4) Studies comparison

Authors	Year	Dataset	Methodology	Performance
Q. Zhao [74]	2021	ancient artifacts found throughout China	Decision tree algorithm for recognition and Gradient boosting for perception aspects	98% accuracy
Yuan et al. [77]	2022	Houma Alliance Book	DCF-LAR	84.82% accuracy
L. Jian et al. [78]	2022	English handwriting texts	CNN proposed model	89.1% accuracy
Williams et al. [79]	2023	5,000 annotated tablet images	ResNet	89% accuracy

Papavassileiou et al. [80]	2023	Mycenaean Greek	BRNN	76.68% accuracy
Proposed Model	2023	Codex Sinaiticus dataset	LSTM	86% accuracy
			RNN	92% accuracy
			GAN	98.3% accuracy
Proposed Model	2023	Argonautica	GAN	98.7% accuracy
			LSTM	94% accuracy
			RNN	88% accuracy

4.5 Discussion

Some points need to be discussed after obtaining results above:

1. The rare of ancient language datasets belong to the lake of documentation and recording in ancient ages.
2. The augmentation has the greatest duty for enforcement of low frequency words in the dataset to enhance the sampling process.
3. The proposed model has been modified to be encoder and decoder model. The benefit of GAN makes it better than LSTM, and RNN models in text prediction.
4. By comparing obtained results with results in related work, encoder models have more prediction abilities than other models. This ability came from that encoder have a high ability in feature extraction.

CHAPTER FIVE

CONCLUSION

AND FUTURE

WORKS

5.1 Review

This chapter discusses conclusion and future work where the conclusion discussed the most important key point in the used model such as challenges, preprocessing, training models and augmentation process and its impact on sampling and dataset balancing. While the future work listed some ideas to apply in future.

5.2 Conclusions

The steps of the proposed system have some points that should be concluded and discussed as follows:

1. Restoring missing parts of ancient text represents a challenging problem because of the unavailability of the dataset. Ancient manuscripts may have been written in scripts and languages that are no longer widely used, making decipherment and translation difficult. Some scripts may not have a clear relationship with modern languages, complicating matters.
2. The preprocessing text and encoding process is a very important stage in transforming text to form where models can do mathematics operations on them. Augmentation is also another important step to make all words take a fair chance in the training process.
3. According to results in the previous chapter, the GAN model has obtained a result better than both RNN and LSTM models. The reason for these results is that the GAN has been tuned to work as auto encoder model. The auto encoder models have a high performance in feature extraction and selection.

4. The models that trained and tested the dataset show that the size of dataset and dataset balancing is very important for sampling process in both training and testing samples. Also, augmentation processes are important for datasets in machine learning because they allow for the artificial growth of dataset size, which improves model generalization by exposing it to a variety of cases. It avoids overfitting, increases robustness to changes, balances class distributions, and guarantees that models are trained on actual circumstances. It provides a resource-efficient method of dataset enrichment, hence encouraging cost-effective model training. Finally, augmentation improves performance, especially when working with restricted or imbalanced datasets.

5.3 Suggestions for future works

Some of future works are planned to be next researches:

1. Multimodal approaches: Use photos, maps, and other non-textual material for better understand the context. This may assist in interpreting unclear text.
2. Modern language transfer learning: Explore transfer learning, where models are pre-trained on a big modern language dataset and fine-tuned on smaller ancient language datasets. This could assist in leveraging language syntactic and semantic commonalities.
3. Quantifying uncertainty: Develop methods to estimate and communicate forecast uncertainty. Ancient texts sometimes have gaps or uncertainties, therefore methods that quantify prediction confidence are useful.

REFERENCES

REFERENCES

- [1] V. Romero, A. H. Toselli, L. Rodríguez, and E. Vidal, “Computer assisted transcription for ancient text images,” *Springer-Verlag Berlin Heidelb.*, vol. 4633 LNCS, pp. 1182–1193, 2007, doi: 10.1007/978-3-540-74260-9_105.
- [2] S. R. Narang, M. K. Jindal, and M. Kumar, “Ancient text recognition: a review,” *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5517–5558, 2020, doi: 10.1007/s10462-020-09827-4.
- [3] E. E. Meyer, “South African Old Testament criticism: Squeezed between an ancient text and contemporary contexts,” *HTS Teol. Stud. / Theol. Stud.*, vol. 71, no. 3, pp. 1–7, 2015, doi: 10.4102/hts.v71i3.2876.
- [4] A. D. Mills, “The Ancient Unconscious: Psychoanalysis and the Ancient Text by Vered Lev Kenaan,” *BMCR*, vol. 77, no. 4, pp. 753–765, 2020, doi: 10.1353/aim.2020.0043.
- [5] M. V Mäntylä, D. Graziotin, and M. Kuutila, “The Evolution of Sentiment Analysis,” *Comput. Rev.*, vol. 27, no. February, pp. 16–32, 2018, [Online]. Available: <https://doi.org/10.1016/j.cosrev.2017.10.002>.
- [6] X. P. Qiu, T. X. Sun, Y. G. Xu, Y. F. Shao, N. Dai, and X. J. Huang, “Pre-trained models for natural language processing: A survey,” *arXiv*, vol. 63, no. 10, pp. 1872–1897, 2021, doi: 10.1007/s11431-020-1647-3.
- [7] S. Ruder, M. Peters, S. Swayamdipta, and T. Wolf, “Transfer learning in natural language processing tutorial,” *Assoc. Comput. Linguist.*, no. 2010, pp. 15–18, 2019.
- [8] J. W. Bartstra, W. P. T. M. Mali, W. Spiering, and P. A. De Jong, “Abdominal aortic calcification: From ancient friend to modern foe,” *Eur. J. Prev. Cardiol.*, vol. 28, no. 12, pp. 1386–1391, 2021, doi: 10.1177/2047487320919895.
- [9] C. Park, C. Lee, Y. Yang, and H. Lim, “Ancient Korean Neural Machine Translation,” *IEEE Access*, vol. 8, pp. 116617–116625, 2020, doi: 10.1109/ACCESS.2020.3004879.
- [10] L. T. England, B. Council, and D. Version, “Language Trends 2020,” *Queen’s Univ. Belfast*, no. 2020, 2020.

- [11] G. Carpenè, D. Negrini, B. M. Henry, M. Montagnana, and G. Lippi, “Homocysteine in coronavirus disease (COVID-19): a systematic literature review,” *Diagnosis*, vol. 9, no. 3, pp. 306–310, 2022, doi: 10.1515/dx-2022-0042.
- [12] M. Llamas, M. L. Garo, and L. Giovanella, “Low free-T3 serum levels and prognosis of COVID-19: Systematic review and meta-analysis,” *Clin. Chem. Lab. Med.*, vol. 59, no. 12, pp. 1906–1913, 2021, doi: 10.1515/cclm-2021-0805.
- [13] A. Savelyev and M. Robbeets, “Bayesian phylolinguistics infers the internal structure and the time-depth of the Turkic language family,” *J. Lang. Evol.*, vol. 5, no. 1, pp. 39–53, 2020, doi: 10.1093/jole/lzz010.
- [14] S. Nelson, I. Zhushchikhovskaya, T. Li, M. Hudson, and M. Robbeets, “Tracing population movements in ancient East Asia through the linguistics and archaeology of textile production,” *Evol. Hum. Sci.*, vol. 2, 2020, doi: 10.1017/ehs.2020.4.
- [15] S. Bird, “Decolonising Speech and Language Technology,” *COLING 2020 - 28th Int. Conf. Comput. Linguist. Proc. Conf.*, pp. 3504–3519, 2020, doi: 10.18653/v1/2020.coling-main.313.
- [16] F. Inchingolo *et al.*, “Oral cancer: A historical review,” *Int. J. Environ. Res. Public Health*, vol. 17, no. 9, 2020, doi: 10.3390/ijerph17093168.
- [17] T. Sommerschild *et al.*, “Machine Learning for Ancient Languages: A Survey,” *Comput. Linguist.*, no. March, pp. 1–45, 2023, doi: 10.1162/coli_a_00481.
- [18] Y. Assael *et al.*, “Restoring and attributing ancient texts using deep neural networks,” *Nature*, vol. 603, no. 7900, pp. 280–283, 2022, doi: 10.1038/s41586-022-04448-z.
- [19] C. Levis *et al.*, “Help restore Brazil’s governance of globally important ecosystem services,” *Nat. Ecol. Evol.*, vol. 4, no. 2, pp. 172–173, 2020, doi: 10.1038/s41559-019-1093-x.
- [20] A. Tabassum and R. R. Patil, “A Survey on Text Pre-Processing & Feature Extraction Techniques in Natural Language Processing,” *Int. Res. J. Eng. Technol.*, no. June, pp. 4864–4867, 2020, [Online]. Available: www.irjet.net.

- [21] A. P. Widyassari et al., “Review of automatic text summarization techniques & methods,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 4, pp. 1029–1046, 2022, doi: 10.1016/j.jksuci.2020.05.006.
- [22] D. Suleiman and A. Awajan, “Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges,” *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/9365340.
- [23] R. Muzzammel, “Machine Learning Based Fault Diagnosis in HVDC Transmission Lines,” *Commun. Comput. Inf. Sci.*, vol. 932, pp. 496–510, 2019, doi: 10.1007/978-981-13-6052-7_43.
- [24] D. A. Bhanage, A. V. Pawar, and K. Kotecha, “IT Infrastructure Anomaly Detection and Failure Handling: A Systematic Literature Review Focusing on Datasets, Log Preprocessing, Machine Deep Learning Approaches and Automated Tool,” *IEEE Access*, vol. 9, pp. 156392–156421, 2021, doi: 10.1109/ACCESS.2021.3128283.
- [25] A. R. Murthy and K. M. Anil Kumar, “A Review of Different Approaches for Detecting Emotion from Text,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1110, no. 1, p. 012009, 2021, doi: 10.1088/1757-899x/1110/1/012009.
- [26] N. Alswaidan and M. E. B. Menai, “A survey of state-of-the-art approaches for emotion recognition in text,” *Knowl. Inf. Syst.*, vol. 62, no. 8, pp. 2937–2987, 2020, doi: 10.1007/s10115-020-01449-0.
- [27] A. I. Kadhim, “Survey on supervised machine learning techniques for automatic text classification,” *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 273–292, 2019, doi: 10.1007/s10462-018-09677-1.
- [28] S. Baek, W. Jung, and S. H. Han, “A critical review of text-based research in construction: Data source, analysis method, and implications,” *Autom. Constr.*, vol. 132, no. February, p. 103915, 2021, doi: 10.1016/j.autcon.2021.103915.
- [29] A. B. Bulsari and S. Palosaari, “Application of neural networks for system identification of an adsorption column,” *Neural Computing & Applications*, vol. 1, no. 2, pp. 160–165, 1993, doi:10.1007/BF01414435.
- [30] B. Wicht, “Deep Learning Feature Extraction for Image Processing,” University of Fribourg, 2017.

- [31] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [32] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, “Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges,” *J. Digit. Imaging*, vol. 32, no. 4, pp. 582–596, 2019, DOI: 10.1007/s10278-019-00227-x
- [33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017, DOI: 10.1145/3065386.
- [34] K. R. Hassan and I. H. Ali, “Age and Gender Classification using Multiple Convolutional Neural Network,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 928, no. 3, 2020, doi: 10.1088/1757- 899X/928/3/032039.
- [35] K. R. Raheem and I. H. Ali, “Classifying Facial Expression using Convolution Neural Network,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 928, no. 3, 2020, doi: 10.1088/1757- 899X/928/3/032036.
- [36] N. D. Al-Shakarchy and I. H. Ali, “Open and closed eyes classification in different lighting conditions using new convolution neural networks architecture,” *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 7, pp. 1970–1979, 2019.
- [37] N. D. Al-Shakarchy and I. H. Ali, “Abnormal head movement classification using deep neural network DNN,” *AIP Conf. Proc.*, vol. 2144, no. August, 2019, doi: 10.1063/1.5123123.
- [38] S. Sharma, S. Sharma, and A. Anidhya, “Understanding Activation Functions in Neural Networks,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 4, no. 12, pp. 310–316, 2020.
- [39] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1237, no. 2, 2019, doi: 10.1088/1742-6596/1237/2/022030.
- [40] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit,” *Nature*, vol. 405, no. 6789, pp. 947–951, 2000, doi: 10.1038/35016072.
- [41] A. F. M. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” *arXiv*, no. 1, pp. 2–8, 2018.

- [42] S.-H. Han, K. W. Kim, S. Kim, and Y. C. Youn, “Artificial Neural Network: Understanding the Basic Concepts without Mathematics,” *Dement. Neurocognitive Disord.*, vol. 17, no. 3, p. 83, 2018, doi: 10.12779/dnd.2018.17.3.83.
- [43] P. Amezcua Aguilar, C. Flores Melero, and C. Marín Perabá, “Neurodiversity as a teaching tool for educational inclusion.,” *Rev. Int. apoyo a la inclusión, Logop. Soc. y Multicult.*, vol. 6, no. 1, pp. 88–97, 2020, doi: 10.17561/riai.v6.n1.08.
- [44] C. C. Aggarwal, “*Neural Networks and Deep Learning*”. Springer International Publishing, 2018.
- [45] S. P. Siregar and A. Wanto, “Analysis of Artificial Neural Network Accuracy Using Backpropagation Algorithm In Predicting Process (Forecasting),” *IJISTECH (International J. Inf. Syst. Technol.*, vol. 1, no. 1, p. 34, 2017, doi: 10.30645/ijistech.v1i1.4.
- [46] S. Ruder, “An overview of gradient descent optimization algorithms,” pp. 1–14, 2016, [Online]. Available: <http://arxiv.org/abs/1609.04747>.
- [47] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–15, 2015.
- [48] C. Fougstedt, M. Mazur, L. Svensson, H. Eliasson, M. Karlsson, and P. Larsson-Edefors, “Time-domain digital back propagation: Algorithm and finite-precision implementation aspects,” *Opt. InfoBase Conf. Pap.*, vol. Part F40-O, 2017, doi: 10.1364/OFC.2017.W1G.4.
- [49] D. Rengasamy, M. Jafari, B. Rothwell, X. Chen, and G. P. Figueredo, “Deep learning with dynamically weighted loss function for sensor-based prognostics and health management,” *Sensors (Switzerland)*, vol. 20, no. 3, 2020, doi: 10.3390/s20030723.
- [50] S. I. Granshaw, “Neural networks and neurodiversity,” *Remote Sens. Photogramm. Soc. John Wiley Sons Ltd*, vol. 36, no. 175, pp. 192–196, 2021, doi: 10.1111/phor.12376.
- [51] [1] K. L. Du and M. N. S. Swamy, “*Neural networks and statistical learning*”, second edition. 2019.
- [52] S. Mangrulkar, “Artificial neural systems,” *ISA Trans.*, vol. 29, no. 1, pp. 5–7, 1990, doi: 10.1016/0019-0578(90)90024-F.

- [53] P. Gang et al., “Dimensionality reduction in deep learning for chest X-ray analysis of lung cancer,” *Proc. - 2018 10th Int. Conf. Adv. Comput. Intell. ICACI 2018*, no. c, pp. 878–883, 2018, doi: 10.1109/ICACI.2018.8377579.
- [54] C. Yu, R. Han, M. Song, C. Liu, and C. I. Chang, “A simplified 2D-3D CNN architecture for hyperspectral image classification based on spatial-spectral fusion,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 2485–2501, 2020, doi: 10.1109/JSTARS.2020.2983224.
- [55] S. Shahriar, “GAN computers generate arts? A survey on visual arts, music, and literary text generation using generative adversarial network,” *Displays*, vol. 73, 2022, doi: 10.1016/j.displa.2022.102237.
- [56] B. Li, X. Qi, P. H. S. Torr, and T. Lukasiewicz, “Lightweight generative adversarial networks for text-guided image manipulation,” *Adv. Neural Inf. Process. Syst.*, vol. 2020-December, no. d, pp. 1–12, 2020.
- [57] M. Zhu, P. Pan, W. Chen, and Y. Yang, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks Jun-Yan,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 5795–5803, 2019.
- [58] G. H. de Rosa and J. P. Papa, “A survey on text generation using generative adversarial networks,” *Pattern Recognit.*, vol. 119, 2021, doi: 10.1016/j.patcog.2021.108098.
- [59] R. C. Staudemeyer and E. R. Morris, “Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks,” arxiv, pp. 1–42, 2019, [Online]. Available: <http://arxiv.org/abs/1909.09586>.
- [60] X. H. Le, H. V. Ho, G. Lee, and S. Jung, “Application of Long Short-Term Memory (LSTM) neural network for flood forecasting,” *Water (Switzerland)*, vol. 11, no. 7, 2019, doi: 10.3390/w11071387.
- [61] F. E. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, “Intrusion detection systems using long short-term memory (LSTM),” *J. Big Data*, vol. 8, no. 1, 2021, doi: 10.1186/s40537-021-00448-4.
- [62] T. Xayasouk, H. M. Lee, and G. Lee, “Air pollution prediction using long short-term memory (LSTM) and deep autoencoder (DAE) models,” *Sustain.*, vol. 12, no. 6, 2020, doi: 10.3390/su12062570.
- [63] P. F. Muhammad, R. Kusumaningrum, and A. Wibowo, “Sentiment Analysis Using Word2vec and Long Short-Term Memory (LSTM) for

- Indonesian Hotel Reviews,” *Procedia Comput. Sci.*, vol. 179, no. 2020, pp. 728–735, 2021, doi: 10.1016/j.procs.2021.01.061.
- [64] K. E. ArunKumar, D. V. Kalaga, C. M. S. Kumar, M. Kawaji, and T. M. Brenza, “Forecasting of COVID-19 using deep layer Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) cells,” *Chaos, Solitons and Fractals*, vol. 146, p. 110861, 2021, doi: 10.1016/j.chaos.2021.110861.
- [65] Y. Wang et al., “PredRNN: A Recurrent Neural Network for Spatiotemporal Predictive Learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2208–2225, 2023, doi: 10.1109/TPAMI.2022.3165153.
- [66] K. Shang et al., “Haze prediction model using deep recurrent neural network,” *Atmosphere (Basel)*, vol. 12, no. 12, pp. 1–12, 2021, doi: 10.3390/atmos12121625.
- [67] M. A. Khan, “HCRNNIDS : Hybrid Convolutional Recurrent Neural,” *Multidiscip. Digit. Publ. Inst.*, 2021.
- [68] J. S. Raj and V. Ananthi J, “Recurrent Neural Networks and Nonlinear Prediction in Support Vector Machines,” *J. Soft Comput. Paradig.*, vol. 2019, no. 1, pp. 33–40, 2019, doi: 10.36548/jscp.2019.1.004.
- [69] P.-N. Tan, M. Steinbach, V. Kumar, T. Pang-Ning, M. Steinbach, and V. Kumar, “Introduction to data mining: Instructor’s,” *Pearson Addison-Wesley*, p. 769, 2006.
- [70] P.-N. TAN, M. STEINBACH, A. KARPATNE, and V. KUMAR, “introduction to data mining second edition”, vol. 7, no. 1. 2015.
- [71] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining (New International Edition)*, no. September. 2014.
- [72] Y. Assael, T. Sommerschild, and J. Prag, “Restoring ancient text using deep learning: A case study on Greek epigraphy,” *arXiv*, vol. 1, 2019, doi: 10.18653/v1/d19-1668.
- [73] H. Wang, H. Wei, J. Guo, and L. Cheng, “Ancient Chinese sentence segmentation based on bidirectional LSTM+CRF model,” *J. Adv. Comput. Intell. Intell. Informatics*, vol. 23, no. 4, pp. 719–725, 2019, doi: 10.20965/jaciii.2019.p0719.

- [74] Q. Zhao, “Research Ancient Artifact Identification Methods under Intelligent Perception and Recognition Technology,” *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/9971343.
- [75] S. Lengauer *et al.*, “Context-based Surface Pattern Completion of Ancient Pottery,” *EUROGRAPHICS Work. Graph. Cult. Herit.*, 2022, doi: 10.2312/gch.20221234.
- [76] L. Wartschinski, Y. Noller, T. Vogel, T. Kehrer, and L. Grunske, “VUDENC: Vulnerability Detection with Deep Learning on a Natural Codebase for Python,” *Inf. Softw. Technol.*, vol. 144, no. i, 2022, doi: 10.1016/j.infsof.2021.106809.
- [77] X. Yuan, Z. Zhang, Y. Sun, Z. Xue, X. Shao, and X. Huang, “A new database of Houma Alliance Book ancient handwritten characters and classifier fusion approach,” *arXiv*, pp. 1–11, 2022.
- [78] L. Jian, H. Xiang, and G. Le, “English Text Readability Measurement Based on Convolutional Neural Network: A Hybrid Network Model,” *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/6984586.
- [79] E. C. Williams, G. Su, S. R. Schloen, M. C. Prosser, S. Paulus, and S. Krishnan, “DeepScribe: Localization and Classification of Elamite Cuneiform Signs Via Deep Learning,” *arxiv*, vol. 1, no. 1, pp. 1–31, 2023, [Online]. Available: <http://arxiv.org/abs/2306.01268>
- [80] K. Papavassileiou, D. I. Kosmopoulos, and G. Owens, “A generative model for the Mycenaean Linear B script and its application in infilling text from ancient tablets,” *J. Comput. Cult. Herit.*, 2023, doi: 10.1145/3593431.
- [81] A. Locaputo, B. Portelli, E. Colombi, and G. Serra, “Filling the Lacunae in ancient Latin inscriptions,” *CEUR Workshop Proc.*, vol. 3365, pp. 68–76, 2023.
- [82] Z. Wenjun, S. Benpeng, F. Ruiqi, P. Xihua, and C. Shanxiong, “EA-GAN: restoration of text in ancient Chinese books based on an example attention generative adversarial network,” *Herit. Sci.*, vol. 11, no. 1, pp. 1–13, 2023, doi: 10.1186/s40494-023-00882-y.

- [83] St Catherine's Monaster at Sinai and National Library of Russia., "The Codex Sinaiticus," Codex sinaiticus - home, <https://codexsinaiticus.org/en/> (accessed Jan. 18, 2024).
- [84] R. Apollonius, "Argonautica, 3 / compiled by Pamela Packard," Oxford Text Archive, <https://lds.ling-phil.ox.ac.uk/lds/xmlui/handle/20.500.14106/0221>, (accessed Jan. 18, 2024).

المستخلص

تعد النصوص القديمة مهمة لأنها تربطنا بالحضارات القديمة، والتي من خلالها نكتسب المعرفة الثقافية والدينية والعلمية. غالبًا ما تكون النصوص القديمة، سواء كانت مكتوبة على ورق البردي أو البرشمان أو غيرها، ناقصة أو متآكلة جزئيًا بسبب مرور الزمن. تمثل استعادة هذه النصوص تحديًا كبيرًا للمؤرخين والعلماء، مما يتطلب جهدًا وخبرة يدوية دقيقة.

استعادة النص القديم هو فرع متخصص من علم استعادة النصوص يركز على استعادة المحتوى النصي من الوثائق التاريخية أو القديمة والحفاظ عليه.

تعتمد طرق الاستعادة التقليدية بشكل كبير على التدخل اليدوي من قبل الخبراء، وهو أمر يستغرق وقتًا طويلًا وغالبًا ما يكون صعبًا. في السنوات الأخيرة، أظهرت تقنيات التعلم الآلي (ML) والذكاء الاصطناعي (AI) نتائج واعدة في تكملة عملية الاستعادة وتحسينها.

أظهرت تقنيات التعلم العميق نجاحًا ملحوظًا في مجالات مختلفة، بما في ذلك معالجة الصور ومعالجة اللغات الطبيعية. في هذه الرسالة تم اقتراح نماذج مختلفة لترميم النصوص القديمة باستخدام الشبكات العصبية العميقة.

تم استخدام مجموعتي بيانات لتدريب واختبار النماذج، مجموعة البيانات الأولى هي "المخطوطة السينائية" وهي مخطوطة يعود تاريخها إلى القرن الرابع، وهي قطعة أثرية مهمة لأنها توفر أقدم نسخة كاملة موجودة من العهد الجديد في الكتاب المقدس المسيحي. المادة المكتوبة بخط اليد مكتوبة باللغة اليونانية.

مجموعة البيانات الثانية هي "Argonautica 3" والتي تشير إلى قصيدة ملحمة كتبها الشاعر اليوناني القديم أبولونيوس الرودسي في القرن الثالث قبل الميلاد وهي مكتوبة باللغة اليونانية أيضًا.

تم معالجة البيانات مسبقاً عن طريق ترميز البيانات ثم إزالة الخطوط والأرقام والرموز والأحرف الخاصة. بعد ذلك، تم تقطيع النص الناتج، وإنشاء حرف مفقود وتسمية الفئات، وإجراء تضخيم البيانات لتعزيزها، وثم جعلها متساوية الطول.

تم استخدام ثلاثة نماذج للتنبؤ كنماذج مقترحة لاستعادة النصوص القديمة المفقودة، وهي الذاكرة الطويلة المدى (LSTM)، والشبكات العصبية المتكررة (RNN)، والشبكات الخصومة التوليدية (GAN) وكانت النتائج اختبار الدقة 86%، 92% و 98.3% وفقاً لمجموعة البيانات الأولى و 94% و 88% و 98.7% وفقاً لمجموعة البيانات الثانية على التوالي.

وبمقارنة أداء كل نموذج، أعطى GAN أفضل النتائج من حيث الدقة، وبالتالي أثبتت فعاليته في مجال استعادة النص المفقود. كما تمت مقارنة نتائج النظام المقترح مع تقنيات الاستعادة الأخرى، حيث أظهرت النتائج أن التقنية المقترحة حققت نتائج دقة أعلى من غيرها.

بشكل عام، يساهم هذا العمل في دمج العلوم المختلفة مثل دمج التعلم العصبي العميق مع العلوم الإنسانية الرقمية، مما يوفر حلاً واعدًا لترميم القطع الأثرية النصية القديمة والحفاظ عليها.



جامعة كربلاء

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم علوم الحاسوب

العنوان بالعربي

استعادة النصوص القديمة باستخدام الشبكات العصبية العميقة

رسالة ماجستير

مقدمة الى مجلس كلية علوم الحاسوب وتكنولوجيا المعلومات / جامعة كربلاء وهي جزء من
متطلبات نيل درجة الماجستير في علوم الحاسوب

كتبت بواسطة

علي عباس علي ابو العوب

بإشراف

أ.د بهيجة خضير شكر

أ.د اسيا مهدي ناصر الزبيدي

٢٠٢٤ م

١٤٤٥ هـ