



People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research



University of Kasdi Merbah Ouargla

Faculty of New Information and Communication Technologies
Department of Electronics and communication

ACADEMIC MASTER'S Thesis

Field: Science and Technology

Specialty: Electronics of Embedded Systems

Theme

An Arabic Text To Speech Synthesis system based on Tacotron model

Presented by :

- Benettouati Salah Eddine
- Boukhetta Mottaz Bellah &
- Bouzid Mohamed laid

Publicly defended on : 24/06/2024 in front of the jury composed of :

Ms. Nadjla BETTAYEB	Supervisor	MCB	UKM Ouargla
Mr. Fatah HAMMOUCHI	President	MAA	UKM Ouargla
Ms. SayhiaTIDJANI	Examiner	MAB	UKM Ouargla

Academic year: 2023/2024

ملخص

تهدف هذه الأطروحة إلى بناء نظام تحويل النص إلى كلام (TTS) باللغة العربية باستخدام نموذج التعلم العميق Tacotron. تتكون المرحلة الأولى من تدريب نموذج الشبكة العصبية الاصطناعية (ANN) القادر على توليد مخططات طيفية صوتية من النص المعطى. تقوم المرحلة الثانية بتحويل المخطط الطيفي المُنتج إلى إشارات صوتية باستخدام نموذج تحسين الصوت HiFi GAN. تم تحقيق الأهداف المرجوة بالكامل، حيث تحصل الكلام العربي المُركب على تقييم عام بلغ 3.66 من 5 من قبل مقيمي النظام.

الكلمات المفتاحية : Tacotron؛ تحويل النص إلى كلام (TTS)؛ التعلم العميق؛ الشبكة العصبية الاصطناعية (ANN)؛ اللغة العربية.

Abstract

This thesis aims to build an Arabic Text To Speech (TTS) system using the deep learning model Tacotron. The first stage consists of training an Artificial Neural Network (ANN) model able to generate audio spectrograms from a given text. The second stage transforms the generated Spectrogram into speech, using the HiFi GAN audio enhancement model. The desired goals were fully reached, as the synthesized Arabic speech received a general rate of 3.66 over 5, from the system evaluators

keywords: Tacotron; TTS; Deep Learning; ANN; Arabic language.

Résumé

Ce mémoire vise à construire un système de synthèse vocale (TTS) en arabe en utilisant le modèle d'apprentissage profond Tacotron. La première étape consiste à entraîner un modèle de réseau neuronal artificiel (RNA) capable de générer des spectrogrammes audios à partir d'un texte donné. La deuxième étape transforme le spectrogramme généré en parole, en utilisant le modèle d'amélioration audio HiFi GAN. Les objectifs souhaités ont été pleinement atteints, car la parole arabe synthétisée a reçu une note générale de 3,66 sur 5 de la part des évaluateurs du système.

Mots-clés : Tacotron ; TTS ; Apprentissage profond ; RNA ; Langue arabe.

ACKNOWLEDGMENTS

First and foremost, we thank God for giving us the strength, courage, and will to accomplish this work.

We express our sincere gratitude and appreciation to our supervisor Ms. **BETTAYEB Nadjla**, for her availability, patience, understanding, human qualities, and interest in our research topic. We thank her for trusting us.

We extend our thanks to the members of the jury, Mr. **Fatah HAMMOUCHI** and Ms. **Sayhia TIDJANI**, who honor us by judging this thesis.

We also thank our teachers for their efforts throughout our years of study at the university and for their support during the realization of this thesis.



Dedication

b. Mohamed laid

To the soul of my dear father, who has always been a source of inspiration and a strong support in my life. Despite his absence, his memory and guidance continue to light my way.

To my dear mother and all my family, your support, patience, and encouragement will remain the enduring flame before me in my path in life and the primary reasons for my perseverance and struggle.

To all my friends who are always with me even in my hardships, thank you for being there for me.

And finally, I do not forget the most important person who struggled with me and overcame hardships together – myself. The road is still long and full of lessons and challenges, so let's continue this struggle to the end.

With my sincere respects,

M. Laid BZ





Moattaz Bellah

My Parents for their support, love, and wisdom that have enabled me to achieve the
Master's degree and become the person I am.

My three sisters "Z" and "S" and "R" and "N".

The families "Benettouati", "Bouzid"

And all my friends .I express my heartfelt gratitude to Allah Almighty, who inspired
and guided me to complete this thesis. I thank my esteemed supervisor, Dr. Nadjla -
Bettayeb, for her continuous support and valuable guidance, which were fundamental
in overcoming challenges and achieving success. Her patience, knowledge, and
diligent oversight had a profound impact on my work. I dedicate this thesis to her as an
expression of my deep appreciation and great gratitude, praying that Allah grants her
success in her academic journey.

B. Salah Eddine



Table of contents

Abstract.....	II
ACKNOWLEDGMENTS.....	III
Dedication.....	IV
Table of contents	VI
List of figures	IX
List of tables	XI
List of abbreviations	XII

Chapter I: Principles of Speech Synthesis and Arabic Language

General Introduction	1
1 INTRODUCTION	3
2 Concept of Speech	3
2.1 techniques of Speech analysis	3
2.1.1 Fourier Transform	3
2.1.2 Discrete Fourier Transform	4
2.1.3 Wavelet Transform.....	4
2.1.4 coefficients	5
2.2 Levels of speech	7
2.2.1 Physiological level.....	7
2.2.2 Acoustic level	8
2.2.3 Linguistic level	8
2.2.4 Pragmatic level	8
3 Arabic language	8
3.1 Letter Exits for Arabic.....	9
4 The concept of technique TTS	10
4.1 Methods of speech synthesis	10
4.1.1 Concatenative speech synthesis.....	10
4.1.2 Statistical Parametric Speech Synthesis	11
4.1.3 Depp Learning Synthesis	12
4.2 Basic components of text-to-speech conversion	12

4.2.1 Text Analysis Unit.....	12
4.2.2 Acoustic model.....	12
4.2.3 Audio encoder	12
4.3 I.10 A review of some applications that utilize speech generation techniques	13
4.4	14
4.5 Speech synthesis quality.....	14
Conclusion	15

Chapter II: Deep Learning

1 INTRODUCTION.....	17
2 Definition.....	17
3 History	17
4 Biological neuron network	17
5 Artificial Neuron Network (ANN)	21
5.1 Activation functions	22
6 Types of ANN Models	23
6.1 Feedforward Neural Networks (FNN).....	23
6.2 Convolutional Neural Networks (CNNs)	24
6.2.1 Convolutional layer (CONV)	25
6.2.1 Pooling layers	27
6.2.2.1 Max layers	27
6.2.2.2 Average Pooling	27
6.2.2 Input layer for ANN (Flattening).....	28
6.2.3 Fully connected layer.....	28
6.3 Recurrent Neural Networks (RNNs)	29
6.3.1 One to One.....	31
6.3.2 One to Many	31
6.3.3 Many to One	31
6.3.4 Many to Many	32
6.4 Long Short-Term Memory (LSTM) Network.....	32

7	Autoencoder	33
7.1	Architecture of Autoencoder in Deep Learning	34
7.2	Types of Autoencoders	35
7.2.1	Undercomplete Autoencoder	35
7.2.2	Sparse Auto encoder	35
7.2.3	Contractive Auto encoder	36
7.2.4	Denoising Auto encoder	36
7.2.5	Convolutional Autoencoder.....	37
7.2.6	Variational Autoencoder.....	38
7.3	Applications of Autoencoders	38
	CONCLUSION	39

Chapter III: Building Arabic Tacotron

1	INTRODUCTION	41
2	Building Arabic Tacotron.....	41
2.1	Tacotron	41
2.1.1	The first part of Tacotron.....	42
2.1.2	The second part of Tacotron.....	43
2.1.3	Running Tacotron	43
2.2	Arabic adaptation of Tacotron	50
3	Evaluation and results.....	53
3.1	Evaluating speech intelligibility	54
3.2	Evaluating speech naturalness	55
4	CONCLUSION	57
	General Conclusion and Outlook	58

List of figures

Chapter I

Figure1.1	General diagram for calculating the characteristic vector of MFCC values and their derivatives	5
Figure1.2	Vowel spectrum [a]: (a) before pre-emphasis; (b) after emphasis	6
Figure1.3	Anatomy of the Human Speech System	7
Figure1.4	Letter Exits for Arabic	9
Figure1.5	General unit-selection synthesis scheme	11
Figure 1.6	Statistical Parametric Speech Synthesis	11
Figure 1.7	Basic components of text-to-speech conversion	13

Chapter II

Figure 2.1	A neuron in biology with its dendritic arborization	18
Figure 2.2	structure of artificial Neuron	20
Figure 2.3	Neural Networks Architecture	22
Figure 2.4	Structure of a CNN for classification	23
Figure 2.5	Typical max polling results	24
Figure 2.6	Typical average polling results	24
Figure2.7	The input layer for the artificial neural network	25
Figure2.8	fully connected neural network for classification	26
Figure2.9	The difference between RNN and FNN	26
Figure2.10	Types of RNN	27
Figure2.11	Schematic representation of the LSTM recurring layer	28
Figure 2.12	Autoencoder scheme	29

Figure 2.13	Undercomplete autoencoder	31
Figure 2.14	Sparse Autoencoder	32
Figure 2.15	Illustration of Denoising Autoencoder	33
Figure 2.16	Deep Clustering with Convolutional Autoencoders	34
Figure 2.17	Schematic diagram of a variable autoencoder	35
Figure 2.18	Deep Clustering with Convolutional Autoencoders	35
Figure 2.19	Schematic diagram of a variable autoencoder	36

Chapter III

Figure 3.1	Tacotron 2 architecture	42
Figure 3.2	Preparing audio files.	45
Figure 3.3	Update metadata for audio files	46
Figure 3.4	Graphics Properties	46
Figure3.5	Load dataset for the original model	47
Figure3.6	Basic settings for the model	48
Figure3.7	Transformation of sound waves to Mel spectrogram	48
Figure3.8	start training	49
Figure3.9	Part of the basic model warehouse structure	51
Figure 3.10	Spectrogram and training cure of a synthesized of audio from an Arabic text	53
Figure 3.11	Spectrogram and training cure of a synthesized of audio from a phonetic writing	53
Figure 3.12	Box plot graph of evaluating the sentences	56

List of tables

Chapter II

Table 2.1	Some used activation functions	20
-----------	--------------------------------	----

Chapter III

Table 3.1	Python libraries used in our work	43
Table 3.2	Arabic letters and their phonetic writing in our model	51
Table 3.3	Percentages of words used in the evaluation	54
Table 3.4	Evaluate the words of the second test in terms of clarity	55

List of abbreviations

MFCC: Millennium Frequency Coefficients

NLP: Natural Language Processing

ReLU: Rectified Linear Unit

TF: Fourier Transform

TFD : Discrete Fourier Transform

TTS: Text-to-speech

General Introduction

In today's digital age, automatic speech synthesis technologies have emerged as crucial innovations that facilitate interaction between humans and machines, offering an efficient means of engagement with smart devices and applications, enabling individuals to engage with devices and applications in a natural and convenient manner. These systems are particularly valuable for enhancing accessibility and, empowering individuals with disabilities and linguistic users to seamlessly access digital content.

The primary objective of this thesis is to explore and develop a TTS (Text to Speech) system tailored for the Arabic language, the system uses deep learning techniques and bases on the Tacotron model. These later stand as one of the latest advancements in artificial speech synthesis, relying on artificial neural networks to convert written text into natural speech that resonates with diverse contexts and applications.

The thesis is divided into three main chapters. The first chapter deals with the basic principles of speech structure and the Arabic language. It includes the origins of the language's pillars and exits, its classification, and its formation within the Arabic language. In addition, the chapter covers the analysis of linguistic texts and their interaction with text-to-speech systems in a coherent and natural way.

In the second chapter, we explore the theoretical foundations of deep learning, including its definition, historical development, and basic components. We begin by examining the inspiration behind artificial neural networks (ANNs), which are designed based on the structure and function of biological neural networks. This understanding allows us to introduce various types of artificial neural networks, including feedforward neural networks (FNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders.

In the third chapter, we studied the original speech synthesis system devoted to English and understood the principle of its work and adapted it to the synthesis of Arabic speech, then we evaluated the model and studied its effectiveness.

Chapter I:
Principles of Speech Synthesis and
Arabic Language

1. INTRODUCTION

Speech synthesis has a great importance in everyday life. It is an essential part of human-machine interaction systems, and it is used in several areas Electronics, computer science, linguistics, physiology, acoustics. In this first chapter, we will introduce the main concepts and knowledge Necessary to build a speech synthesis system in the Arabic language. First, we begin with a general description of speech and some of its levels. Next, we will give a brief description of the Arabic language and its most important features. Finally, we proceed to explain the Text To Speech (TTS) technology and its most used methods.

2. Concept of Speech

Speech is the process of producing sounds using the oral and laryngeal organs to express thoughts, emotions, and information. Speech involves generating spoken sounds through the voice, which can be deciphered and understood by others In the field of signal processing, speech is also considered an audio signal containing important information that is analyzed and processed. This includes extracting acoustic features such as frequencies and sound waves, converting speech into a digital representation to enable applications like speech recognition, converting speech to written text, sentiment analysis through voice, and many other applications.

2.1. Techniques of speech analysis

Speech is an acoustic signal whose characteristics vary depending on the type of sound (quasi-periodic, random, or pulsed). To analyze this signal and extract its important parameters, temporal analysis techniques can be used to study changes in time or spectral analysis techniques to study changes in frequency. We will start by explaining the most important of these techniques.

2.1.1. Fourier Transform (FT)

In speech processing domains, we use the so-called short-term FT, due to the instability of the speech signal. This weighting factor is obtained by extracting about thirty milliseconds of the signal, and then applying a weighting window (often a Hamming window). Finally, FT is performed on the resulting samples, as shown in Equation (1.1). FT is used to obtain the

spectrum of the speech signal, where the acoustic portions of the signal appear as a series of successive sharp spectral peaks. [1]

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft} dt \quad (1.1)$$

Where:

$X(f)$: Fourier Transform of the signal

$x(t)$: Time – domain function (the original signal)

f : Frequency.

j : Imaginer unit

2.1.2. Discrete Fourier Transform:

The discrete Fourier transform is a discrete version of the FT, used to analyze specific and reproducible signals, as calculated in equation (1.2) and commonly used in digital signal processing

$$X[K] = \sum_{n=0}^{N-1} x[n]e^{-\frac{2\pi}{N}Kn} \quad (1.2)$$

Where:

$K= 0, \dots, N - 1$.

$x[n]$: a digital signal frame of length N .

$X [K]$: The Discrete Fourier Transform (DFT) of $x[n]$.

2.1.3. Wavelet Transform :

Wavelet transform is a mathematical process that converts a signal into a set of small wavelets providing a combined analysis in time and frequency. It is widely used to analyze unstable signals.

$$X(t)\psi \frac{t-b}{a} dt_{-\infty}^{\infty} \frac{1}{\sqrt{a}} = w_{\psi}(a, b) \quad (1.3)$$

$W_{\psi}(a, b)$: Wavelet Transform of the signal $x(t)$ using the mother wavelet ψ

a : Scaling factor.

b : Translation factor.

ψ : Wavelet function (mother wavelet).

2.1.4. Millimeter Frequencies Cepstral Coefficients (MFCC) :

The MFCC represents a set of spectral coefficients plotted on a scale known as the Mel scale. This representation is inspired by the human auditory system, which exhibits a logarithmic sensitivity that decreases with increasing frequency. MFCC values are calculated by following the steps described below (Figure1.1).

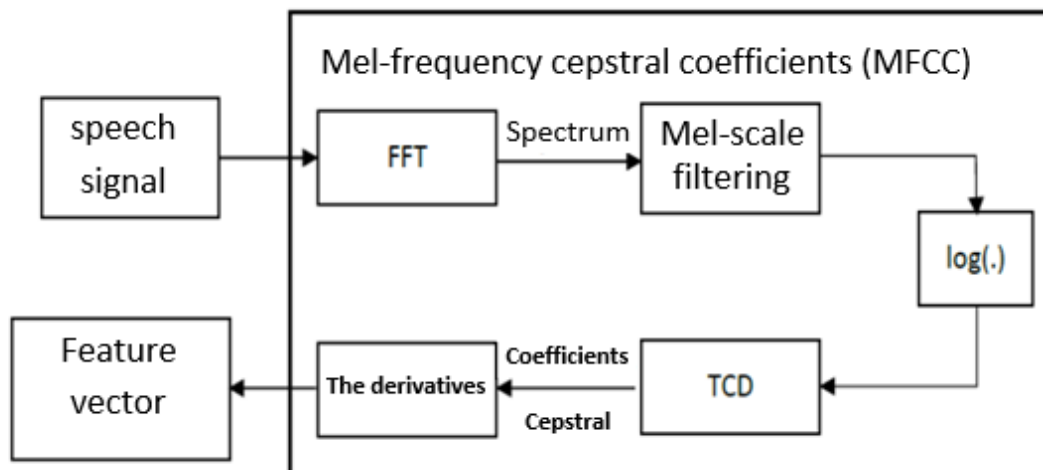


Figure 1.1: General diagram for calculating the characteristic vector of MFCC values and their derivatives

- To better analyze and model the speech signal, a pre-emphasis filter must be applied to enhance the energy of high frequencies. This is because the energy in most voiced sounds tends to decrease at high frequencies due to the nature of the glottal pulse (Figure 2). The filter used is a first-order high-pass filter.

$$(z) = 1 - \alpha z^{-1} \quad (1.4)$$

avec: $0.9 \leq \alpha \leq 1.0$;

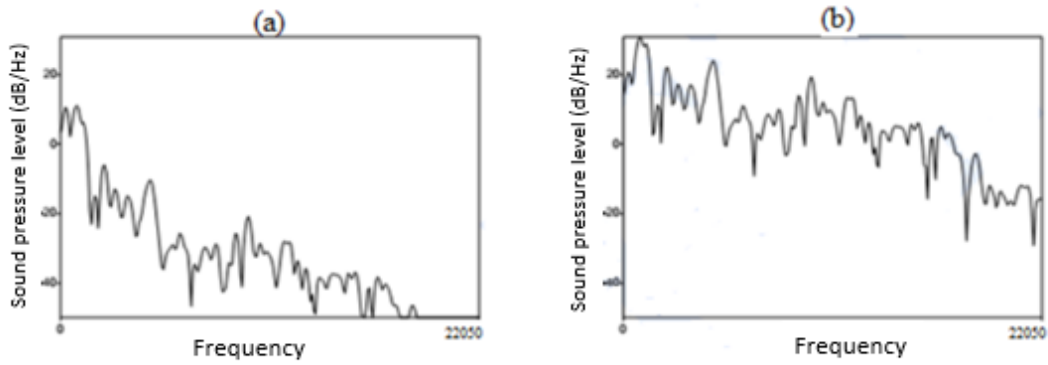


Figure 1.2: Vowel spectrum [a]: (a) before pre-emphasis; (b) after emphasis

- **Frame Division:** Due to the non-stationarity of the speech signal, it needs to be segmented into segments of approximately 20-30 milliseconds with a 50% overlap of the segmentation windows.
- **Applying the Fourier Transform:** Computing the spectrum of each frame using a Discrete Fourier Transform (DFT)

$$x(n) \rightarrow X(F) \quad (1.5)$$

- **Changing the Scale:** The Discrete Fourier Transform (DFT) computes the signal's energy in the frequency domain, but due to the human auditory system's sensitivity to frequencies, this rendering must be done in the tonal scale

$$me(f) = 1127 \ln \left(1 + \frac{f}{700} \right) \quad (1.6)$$

- This scaling process is accomplished by multiplying the spectrum through a set of filters, while capturing the energy of each frequency band individually (according to equation (1.7)). The filters in this set are arranged according to the melodic scale, These filters are characterized by a progressive bandwidth, with a higher density of filters at low frequencies and a decrease in their number at high frequencies;

$$E[k] = \sum_f w_k[f] |x[f]|^2 \quad (1.7)$$

2.2. Levels of speech

Language is a single coherent structure that cannot be fragmented during the actual performance of speech, as all its elements interact with each other and cooperate in achieving linguistic purposes, and it is not possible to pay attention to one of them without the other, despite this strong cohesion between linguistic levels and systems, this did not prevent linguists from studying each level separately. Based on this, We have divided the levels as follow.

2.2.1. Physiological level :

It focuses on the anatomy of the organs responsible for speech production. As shown in the figure [1]

- **Lungs:** where the air that is used to produce speech is generated.
- **Larynx:** Plays a key role in generating sound by conditioning the air passing through the vocal cords.
- **Vocal cords:** They vibrate when air passes through them to produce sound.
- **Palate and tongue:** Used to shape sound into understandable speech.
- **Lips:** They can be used to enhance or alter sounds to produce letters and different sounds in language.

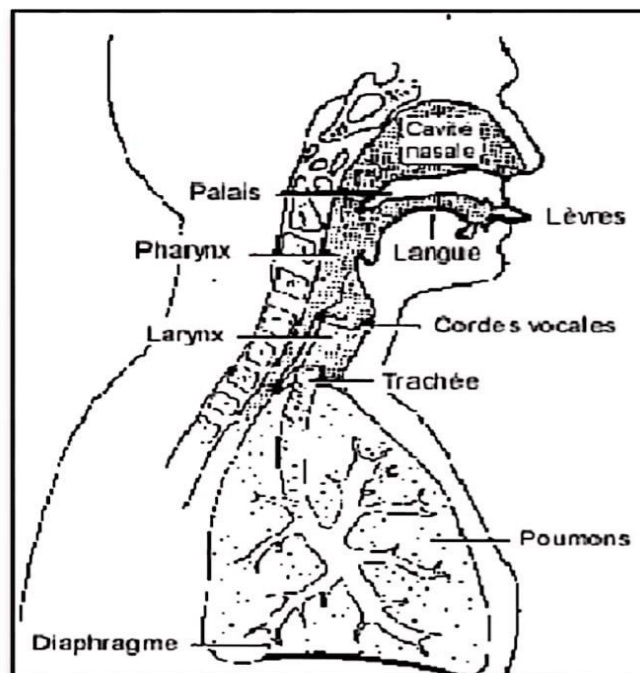


Figure 1.3: Anatomy of the Human Speech System

2.2.2. Acoustic level

Interested in the study of sound itself and its physical properties.

Frequency:

Refers to the number of times the particles vibrate in a unit of time and determines the loudness of the sound, with high frequencies producing loud sounds and low frequencies producing low sounds.

Intensity:

The strength or effective power of a sound, an expression of the energy carried by the sound. It is measured in decibels (dB).

Wavelength:

The distance between two peaks or troughs of a wave, expressed in units of length.

Wave nature:

Sound can be a wave, moving through a physical medium, such as air, and sound can also be light or microphonic oscillations. These physical properties of sound determine how it is shaped and interacts with and how it is perceived and interpreted by the auditory system in the human ear.

2.2.3. Linguistic level:

Studies the phonetic and semantic units of speech, such as phonemes, words, and sentences, and how they are used to communicate and convey meaning.

2.2.4. Pragmatic level:

Concerning the context, purpose, and situation involved in speech, and how these factors affect the linguistic interaction between individuals.

These levels work together to enable speech, comprehension, and the exchange of information between individuals.

3. Arabic language

Arabic is an ancient Semitic language, one of the oldest languages in the world, spoken by about 422 million people in the Middle East and North Africa. It is characterized by a variety of dialects, literary and cultural richness, and is the language of the Holy Quran. Arabic has a

complex grammatical and morphological system and is written from right to left. It is characterized by many letters exits and qualities that make it unique and distinct from other languages.

3.1. Letter Exits for Arabic

The articulation points of letters are fundamental in studying the Arabic language and understanding correct pronunciation. These points rely on different positions in the mouth and throat where sounds are produced, allowing for precise distinction of letters and accurate pronunciation. The articulation points are distributed across various areas, including the oral cavity, throat, tongue, lips, and nasal cavity, with each articulation point playing a specific role in producing the different sounds that distinguish the Arabic language. Knowledge of these points is crucial in several fields, such as Tajwid, where they are essential for accurate recitation of the Quran, language teaching, where they help improve proper pronunciation for children and non-native speakers, as well as in speech correction and training for broadcasters and voice professionals. Studying the articulation points of letters is necessary to ensure correct understanding and clear pronunciation of Arabic, thereby preserving its beauty and precision.

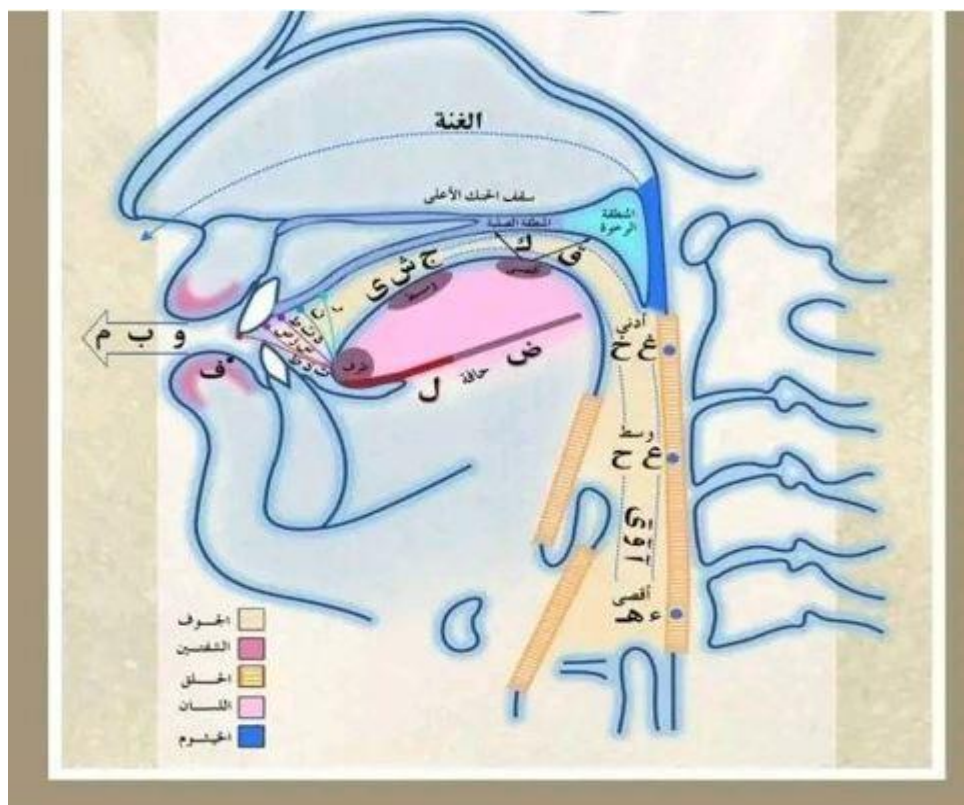


Figure 1.4: Letter Exits for Arabic

4. The concept TTS

TTS or text-to-speech is a technological technique for simulating the human voice using a computer or various speech systems. The main task of a TTS engine is to convert written or stored words in the form of text into spoken words with a human voice, and one of its most important features is that it is rich in different qualities than other technologies such as speaker, language, feeling, and so on. In addition, it could be trained on a huge amount of noisy real-world data, and what makes it a notable technology is that it was originally a simulation of human neural networks that contributed to the improvement of speech quality.

4.1. Methods of speech synthesis

4.1.1. Concatenative speech synthesis

Sequential speech synthesis is a speech synthesis method in which pre-recorded speech units, such as audio clips, syllables, or words, are grouped or strung together to generate speech output. These speech units are usually stored in a database and are dynamically selected and ordered based on the input text. This approach often produces high-quality, natural-sounding speech because it uses actual recordings of human speech. However, it can be computationally intensive and may require large storage space for the large number of recorded speech units. In short, recordings of short audio segments are combined to create speech but it still has some disadvantages [2].

→ Problems:

- Very large data requirements
- Unnatural speech (without emotional and intonation components)
- Long development time
- Language specific

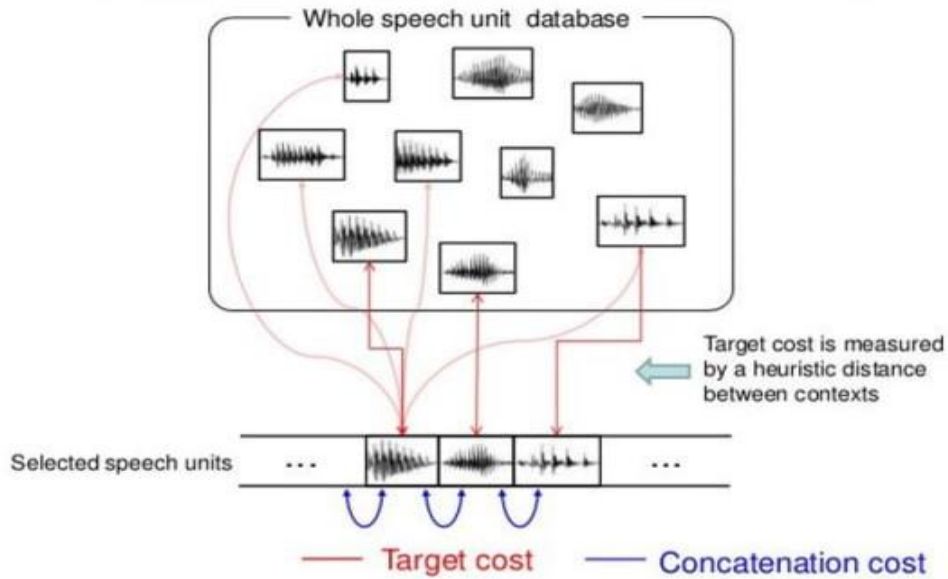


Figure 1.5: General unit-selection synthesis scheme

4.1.2. Statistical Parametric Speech Synthesis

Linguistic features such as phonetics and duration are extracted from the text, while speech parameters such as leakage, frequency, spectral slope and linear spectrum are extracted from the corresponding speech signal the audio encoder, a system that generates the wave signal, encodes these parameters. One of the advantages of this technique is that it allows converting voice characteristics, speaking patterns and emotions, is considered less difficult than traditional sequencing, requires less data for training and supports multiple languages. However, one of the problems faced by this technology is that it may lead to the production of muffled speech or tinnitus [7].

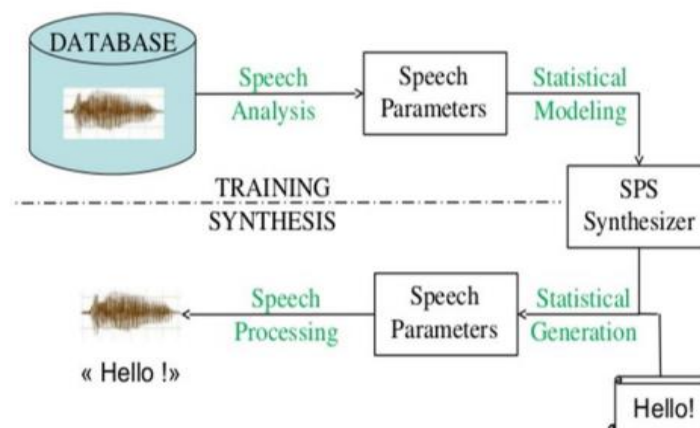


Figure 1.6: Statistical Parametric Speech Synthesis

4.1.3. Deep Learning Synthesis

Deep learning methods include a variety of methods and models that use artificial neural networks to process and analyze data. Among these the best of these methods is the method of repetitive neural networks (RNN) these networks are used to process serial data such as text, speech and video. It is characterized by its ability to remember past information using repetition in its calculations.

4.2. Basic components of text-to-speech conversion:

4.2.1. Text Analysis Unit:

It means the part responsible for analyzing written or typed text. This module uses Natural Language Processing (NLP) techniques to directly understand the text, such as identifying sentences and keywords and recognizing the linguistic structure and vocabulary used. Text analysis can include several tasks such as defining vocabulary, analyzing sentence structure identifying the topic of the text, and many other things that contribute to understanding the meaning of the text completely and accurately.

4.2.2. Acoustic model:

Refers to a model that is used to generate artificial speech or to digitally represent a voice. This model is used in artificial speech generation techniques to accurately and realistically represent human voices. These models are trained on the human voice using a large set of acoustic data to ensure the generation of natural and understandable speech: Refers to the model that is used to generate artificial speech or to digitally represent a voice. This model is used in artificial speech generation techniques to accurately and realistically represent human voices. These models are trained on the human voice using a large set of audio data to ensure the generation of natural and understandable speech.

4.2.3. Audio encoder:

It refers to the process of converting audio signals into an encrypted, for the purpose of protecting audio data from unauthorized access. Voice encryption techniques are used in several scenarios, such as encrypted phone calls and applications that enable secure audio sharing, where audio is converted into an encrypted form before it is transmitted over the network and decrypted when it reaches the recipient

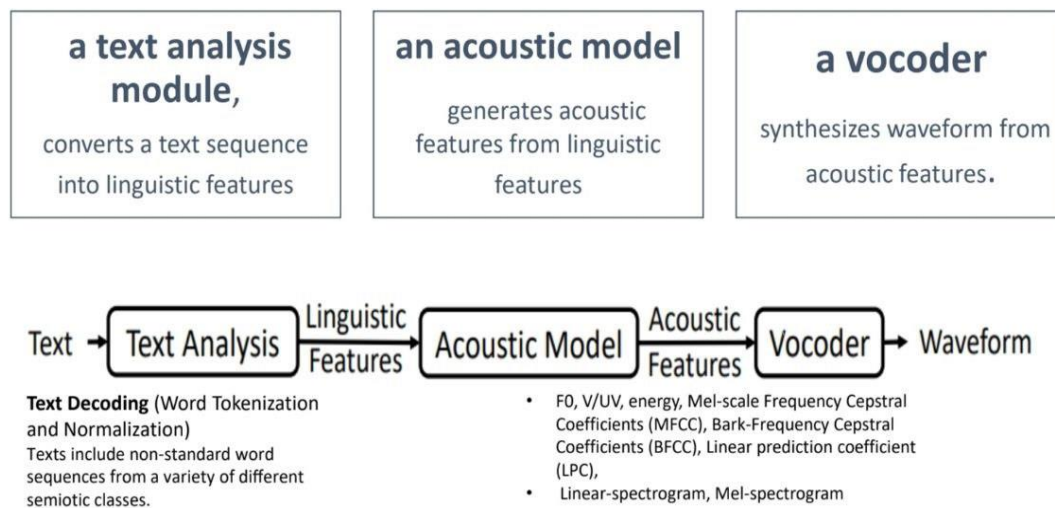


Figure 1.7: Basic components of text-to-speech conversion

4.3. Speech synthesis applications

TTS technologies is used in many applications like:

- **Education and training:** Educational apps that use TTS to convert educational texts into audible speech, making it easier for students to access educational materials in a convenient way. For example, apps for learning languages or apps for reading textbooks
- **Healthcare:** TTS can be used in medical applications to convert medical text into audible speech, making it easier for doctors and patients to better understand medical information.
- **Entertainment and games:** TTS is used in games and entertainment applications to deliver a more interactive and fun experience, such as virtual characters that interact with players with their voice.
- **Digital personal assistants:** Digital personal assistants such as Apple's Siri, Google Assistant, and Amazon Alexa use TTS technologies to respond to user queries and provide information in an audible voice.
- **News and media applications:** TTS can be used to convert newspaper articles, blogs, and media content into audio files, making it easier for people to listen to content while driving, working, or doing daily activities.

4.4. Challenges in Arabic speech synthesis

Developing a Arabic TTS systems may faces several specific challenges as:

- The diverse Arabic language: Arabic has a variety of different dialects and accents, making it a challenge to design a TTS system that works well for all these dialects.
- Pronunciation and articulation:

Some Arabic sounds are not found in other languages and need special processing for correct pronunciation, and this requires complex rules to generate the correct pronunciation of words.

- Phonetic order:

In Arabic, words can change dramatically depending on the order of the letters within them, meaning that the system needs a deep understanding of the language to generate texts correctly.

- Special names and technical terms:

Many special names and technical terms are not available in general dictionaries, making the correct pronunciation of these words an additional challenge.

To overcome these challenges requires continuous research and development in the field of developing Arabic TTS systems, in addition to using artificial intelligence and deep learning techniques to improve the performance of these systems.

4.5. Speech synthesis' quality

The quality of speech synthesis is evaluated based on the voice's ability to resemble a human voice and its ability to be clearly understood. Intelligent programs can convert text to speech, helping people with low vision or reading disabilities to listen to written material on their home computer. Speech synthesis technologies have been included in many operating systems since the early 1990.[3].

5. Conclusion

Text-to-speech technology (TTS) is an important technology in the field of natural language processing, as it converts written text into audible speech. Although it has many challenges, this technology is characterized by several benefits, including facilitating access to information, supporting people with special needs, and enhancing voice interaction in smart devices. It also increases productivity by enabling users to listen to content while doing other activities. Thanks to continuous advances in AI and deep learning technologies, as we will see in the next chapters, the quality and performance of TTS systems are continuously improving, enhancing their ability to deliver a more natural and effective user experience.

Chapter II:

Deep Learning

1. INTRODUCTION

Deep learning, undoubtedly, represents one of the most intriguing fields in artificial intelligence and computer science. In this chapter, we will provide a comprehensive overview of the concept of deep learning, starting from explaining the fundamental principles and theory of operation to its diverse applications in the field of natural language processing. In addition, we will analyze different types of neural networks, including Feed Forward neural networks, recurrent neural networks, convolutional neural networks, autoencoders and highlight the role of each type in solving sensitive problems and complex challenges.

2. Definition

Deep learning is a subset of machine learning that utilizes artificial neural networks with multiple layers to learn and extract high-level features from complex data. Unlike traditional machine learning algorithms that may require feature engineering, deep learning models can automatically discover patterns and representations directly from raw data. These deep neural networks consist of interconnected layers of artificial neurons, each layer building upon the previous one to progressively extract more abstract and complex features [4] .

One of the defining characteristics of deep learning is its ability to handle large volumes of data efficiently, making it particularly effective in tasks such as image and speech recognition, natural language processing, and reinforcement learning [4]. Deep learning has achieved remarkable success in various fields, including computer vision, healthcare, finance, and autonomous driving. [5].

3. History

The history of deep learning is a captivating narrative that traces significant milestones in the evolution of artificial intelligence and machine learning. Originating from the foundational concepts of neural networks pioneered by researchers like Warren McCulloch and Walter Pitts in the 1940s and 1950s, early developments were constrained by computational limitations and data scarcity. Despite promising beginnings, progress in neural networks stagnated during the 1970s and 1980s, a period known as the "AI winter," marked by disillusionment and reduced funding for AI research. However, the resurgence came with the breakthrough development of the backpropagation algorithm by Geoffrey Hinton, David Rumelhart, and Ronald Williams in the 1980s. This algorithm revolutionized neural network training, unlocking the potential for efficient training of multi-layer networks. Concurrently, pioneers like Yann LeCun introduced specialized architectures such as Convolutional Neural Networks (CNNs) in the late 1980s, tailored for tasks like image recognition. Moreover, Recurrent Neural Networks (RNNs) emerged as a solution for processing sequential data with temporal dependencies. The tipping point arrived around the early 2010s, fueled by advancements in computational resources, data availability, and algorithmic innovations, leading to a deep learning renaissance. Breakthroughs in image recognition, natural language processing, and other domains propelled deep learning to the forefront of AI research and applications. Today, deep learning techniques power a myriad of real-world applications across various industries, showcasing the transformative potential of AI. Yet, researchers continue to explore avenues for enhancing the efficiency, interpretability, and ethical considerations of deep learning models, driving the field forward into new frontiers of innovation and discovery. [6]

4. Biological neuron network

A biological neural network is a complex system of interconnected neurons that communicate through electrical and chemical signals to process and transmit information in the brain and nervous system. The provided image illustrates the structure of a single neuron, highlighting its key components and their roles in neural communication. The process begins with the dendrites, which are branching structures that receive incoming signals from other neurons. These signals are transmitted through synapses, such as the axo-dendritic synapse, where the axon terminal of one neuron connects to the dendrites of another. The received signals are then conveyed to the cell body (or soma), which contains the nucleus and is responsible for processing these signals.

From the cell body, if the signal strength is sufficient, it is transmitted along the axon. The axon is a long, slender projection that conducts electrical impulses away from the cell body. The axon can form multiple synapses with other neurons, including axo-somatic synapses, where the axon connects to the cell body of another neuron, and axo-axonic synapses, where the axon connects to another axon.

In summary, a biological neural network functions through the intricate connections and interactions between neurons, facilitated by structures such as dendrites, cell bodies, axons, and various types of synapses. These networks are essential for all brain activities, including sensory perception, motor control, and cognitive processes. The image effectively illustrates these components and their interactions, providing a clear understanding of how neurons operate within the broader neural network., as shown in, Figure 2.1.[7]

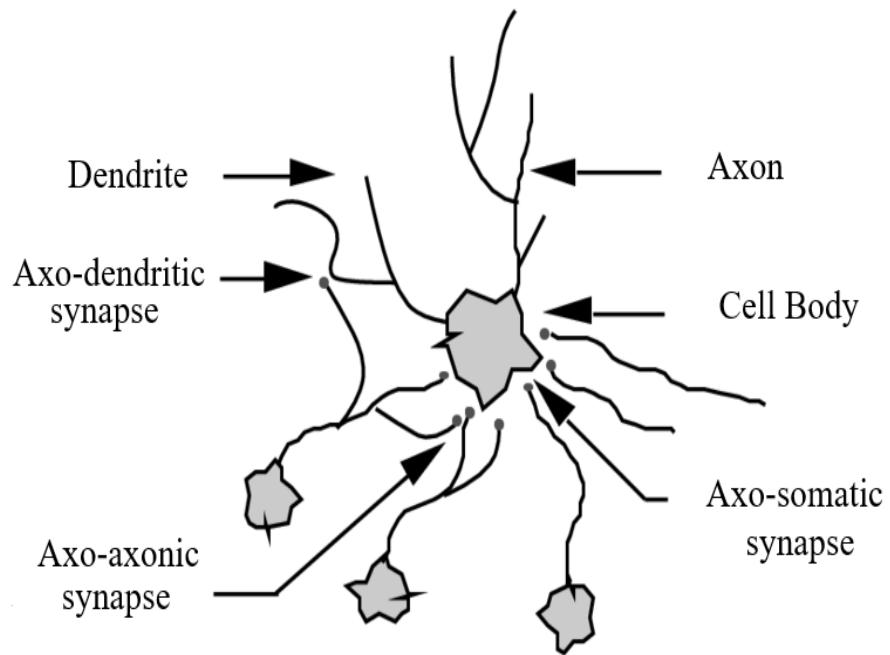


Figure 2.1: A neuron in biology with its dendritic arborization [7]

5. Artificial Neuron Network (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the biological neural networks in the human brain. They consist of interconnected groups of artificial neurons that work together to solve specific problems such as pattern recognition, classification, and prediction. The provided image illustrates a simplified model of an artificial neuron. The process begins with the inputs (x_1, x_2, \dots, x_n) , which represent the signals or features fed into the neuron, where each input corresponds to a specific feature of the data being analyzed.

Each input is associated with a weight (w_1, w_2, \dots, w_n) that determines its importance or strength. These weights are adjusted during the training process to improve the network's performance. The summation function Σ sums the weighted inputs, calculating the total input signal as the weighted sum of all inputs plus a bias term (θ) . The bias is an additional parameter added to the summation function to help adjust the output independently of the input values, allowing the activation function to shift its response curve and aiding the model in learning more effectively.

Next, the activation function $f(x)$ processes the summed input and determines the neuron's output. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU), which introduce non-linearity into the model, enabling the network to learn complex patterns. Finally, the neuron's output is the result of the activation function, which can be passed to other neurons in subsequent layers or can serve as the network's final output.

In summary, an artificial neural network processes input data through layers of artificial neurons, each performing calculations similar to those shown in the image. The interconnected structure and learning algorithms enable the network to model and predict complex relationships within the data, making ANNs powerful tools in various fields such as image recognition, natural language processing, and financial forecasting. [7] [8]

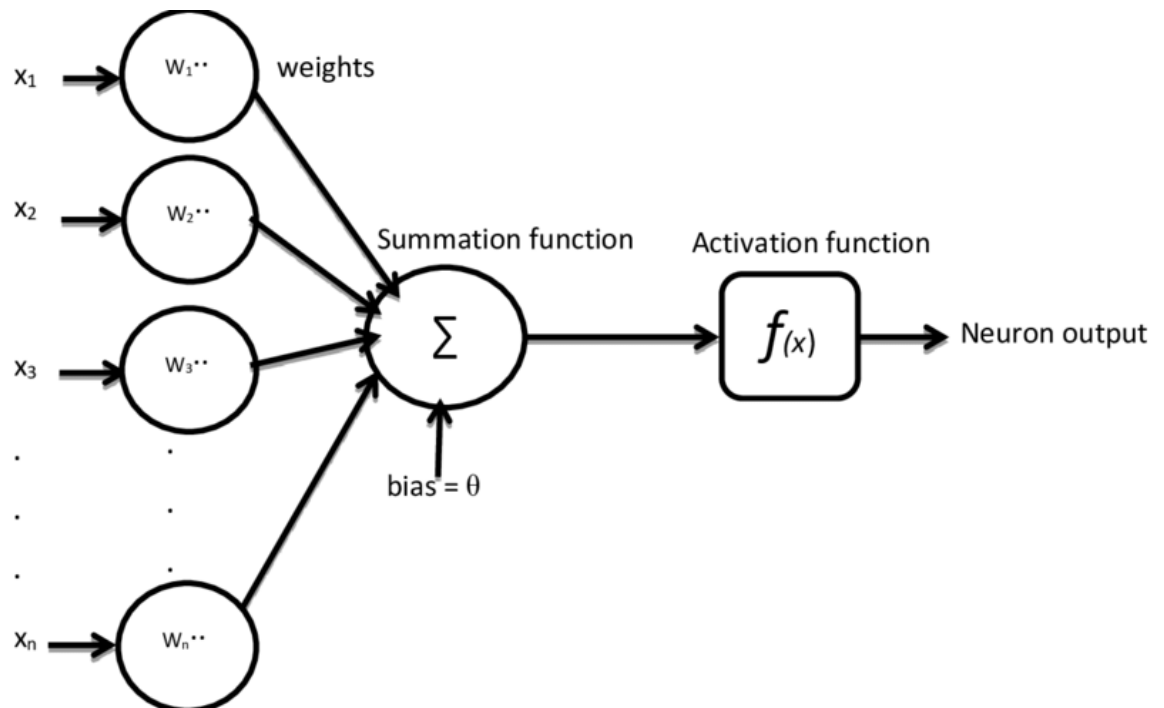
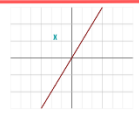


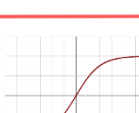

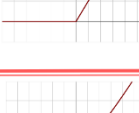
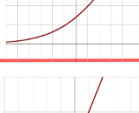



Figure 2.2: structure of artificial Neuron [8]

5.1 Activation functions

Activation functions are mathematical equations that determine the output of a neural network. These functions are applied to express non-linear relationships between input and output. When non-linearity is employed, a neural network becomes a universal function approximator. The neuron's output has no bounds, ranging from $-\infty$ to $+\infty$, so the neuron does not truly have limits. To discern whether a neuron is activated or not, an activation function is introduced. Table II.1 presents the activation functions used. [9][10]

Table 2.1: Some used activation functions [9]

ACTIVATION FUNCTION	PLOT	EQUATION	DERIVATIVE	RANGE
Linear		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary Step		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
Hyperbolic Tangent(tanh)		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
Rectified Linear Unit(ReLU)		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$
Softplus		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Leaky ReLU		$f(x) = \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-1, 1)$
Exponential Linear Unit(ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$f'(x) = \begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$[0, \infty)$

6. Types of ANN Models

6.1 Feedforward Neural Networks (FNN)

ANN comprise artificial neurons known as units, arranged in layers that form the network's structure. The number of units in each layer varies, ranging from a few to millions, depending on the complexity of the network's task. Typically, an ANN includes an input layer, hidden layers, and an output layer. The input layer receives external data for analysis, which then traverses through the hidden layers, undergoing transformations to extract valuable

insights for the output layer. The output layer generates a response based on the input data received.[11]

In most neural networks, units are interconnected between layers, with each connection possessing weights that dictate the influence one unit has on another. As data propagates through the network, these connections enable learning, progressively refining the network's understanding of the data, ultimately leading to an output from the output layer.[12]

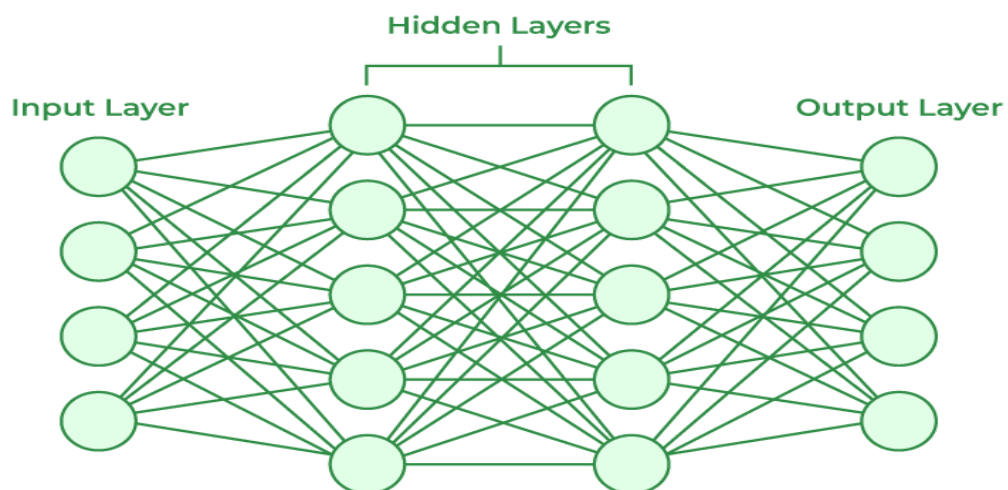


Figure 2.3 Neural Networks Architecture [12]

The architecture and functions of human neurons provide the foundation for artificial neural networks, also known as neural networks or neural nets. In an artificial neural network, the input layer serves as the initial stage, receiving input from external sources and transmitting it to the hidden layer, which follows as the second layer. Within the hidden layer, each neuron receives input from neurons in the preceding layer, computes a weighted sum, and forwards it to neurons in the subsequent layer. These connections are weighted to adjust the impact of inputs from the previous layer, with different weights assigned to each input. During the training process, these weights are optimized to enhance model performance.

6.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) stand as the most proficient models to date for image classification, comprising two distinct components. The initial segment of the CNN is

the convolutional segment proper, functioning as an image feature extractor. Here, the image undergoes a series of transformations via filters, or convolution kernels, generating new images known as convolution maps. Certain intermediary filters may reduce image resolution through local maximization operations. Ultimately, these convolution maps are flattened and consolidated into a feature vector termed the CNN code. Subsequently, this CNN code serves as input for the second segment, which comprises fully connected layers. This section's role is to amalgamate the features of the CNN code to classify the image. The output layer encompasses one neuron per category. Typically, the numerical values obtained are normalized between 0 and 1, with a sum equaling one, to yield a probability distribution across categories. As depicted in (Figure 2.4), a CNN is composed of:

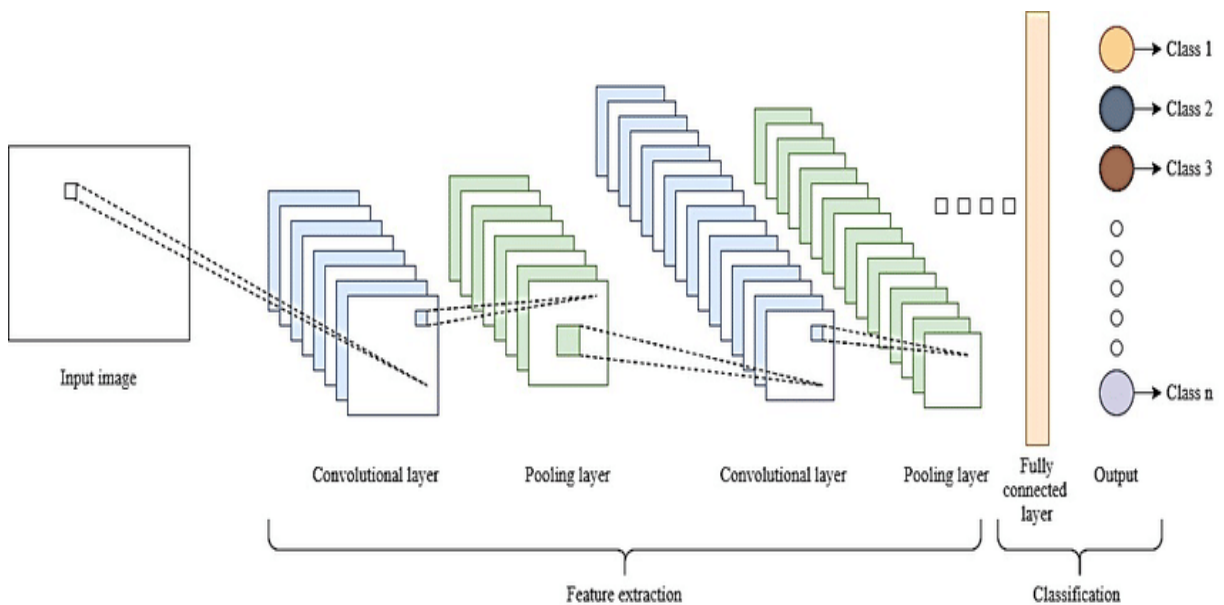


Figure 2.4 Structure of a CNN for classification [13]

6.2.1 Convolutional layer (CONV)

The convolutional layer is responsible for extracting features from the object in the image. Multiple convolutional layers can be employed, each with the functionality to capture high or low-level features such as edges, colors, gradient orientation, etc. This collection of layers forms an overall network for a comprehensive understanding of the image. To grasp how these layers operate, we begin by presenting an example as in (Figure 2.5) [14].

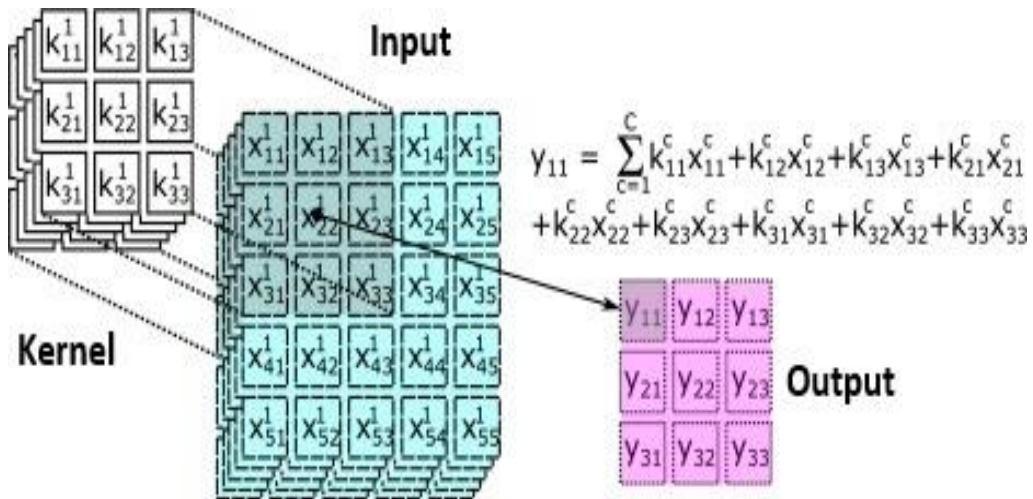


Figure 2.5.: Multichannel convolution [15].

During the convolution operation within a CNN layer, a learnable kernel of size 3x3 traverses the input image element-wise. The kernel performs a dot product with the corresponding receptive field in the image, effectively calculating a weighted sum. This process emphasizes specific features within the local image region, such as edges or textures. The resulting activation is then passed on as input to the subsequent layer. This convolution is repeated for all receptive fields within the input image. This is applied for one layer only, FigureII.6 depict this operation [14].

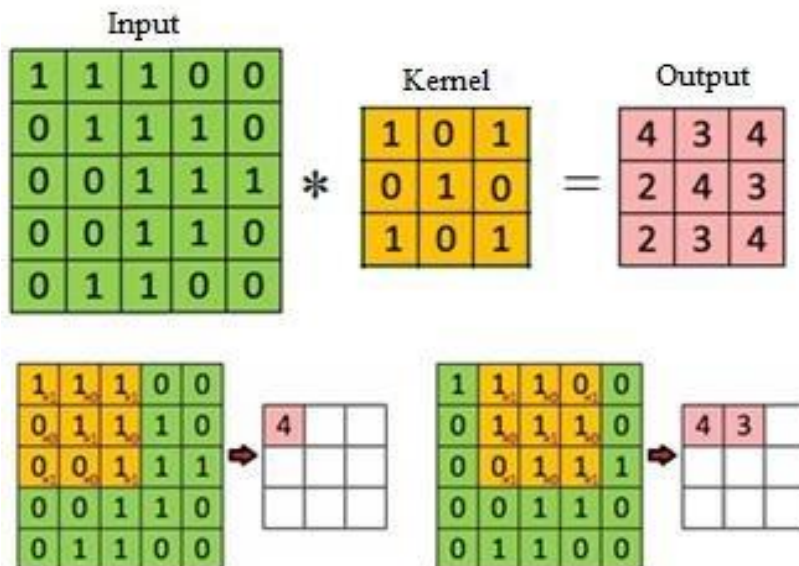


Figure 2.6.: The new matrix after applying the kernel filter [16].

6.2.2 Pooling layers

This type of layer is often placed between two convolution layers. It receives several feature maps as input, and applies the pooling operation to each of them. A pooling layer acts as a reduction layer. It divides the image into blocks and only keeps the maximum of each block. This makes it possible to reduce the size of the image while retaining the most important characteristics. We obtain at the output the same number of feature maps as at the input, but they are much smaller [17].

6.2.2.1 Max Pooling

We iterate the kernel over the matrix and select the maximum value from each window. In the above example, we use a 2x2 kernel with stride two and iterate over the matrix, forming four different windows, denoted by different colors. As shows Figure II.7, in max pooling, we only retain the largest value from each window. This down samples the matrix, and we obtain a smaller 2x2 grid as our max pooling output.[18]

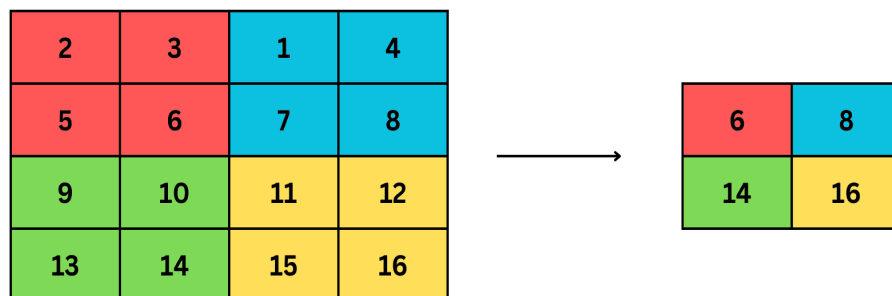


Figure 2.7 : Typical max polling results [18].

6.2.2.2 Average Pooling

In average pooling, we similarly iterate over windows. However, we consider all values in the window, take the mean and then output that as our result.as shown in (Figure 2.8) [19].



Figure 2.8: Typical average pooling results [19].

➔ These types are the most commonly used, but they depend on the size and number of objects in the image

6.2.3 Input layer for ANN (Flattening) :

Once the image features are extracted by the previous layers, the next step involves converting the output of these layers into a suitable structure for the neural network. The extracted features are flattened to form a vector (multi-layer perceptron) for the input layer of this artificial neural network (see Figure 2.9) [20].

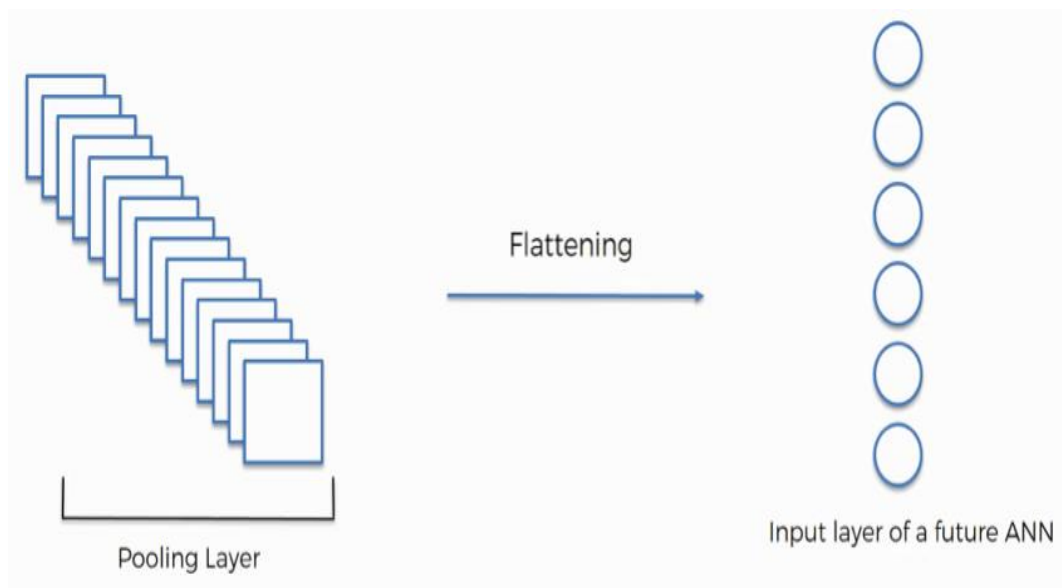


Figure 2.9: The input layer for the artificial neural network [20].

6.2.4 Fully connected layer

This layer comprises an artificial neural network that utilizes the feature vector obtained from the preceding flattening layer as its input. These extracted features are subsequently

amalgamated into a set of higher-level attributes, thereby enhancing the convolutional network's robustness in classifying the input images. The nodes within the network's hidden layer function as feature detectors (activation functions), interconnected via weighted connections (represented by the links connecting the nodes in Figure 2.10). During the training process, the network employs backpropagation at each iteration, enabling the model to effectively learn and distinguish the most dominant features associated with each class, consequently refining classification accuracy and minimizing error. [21]

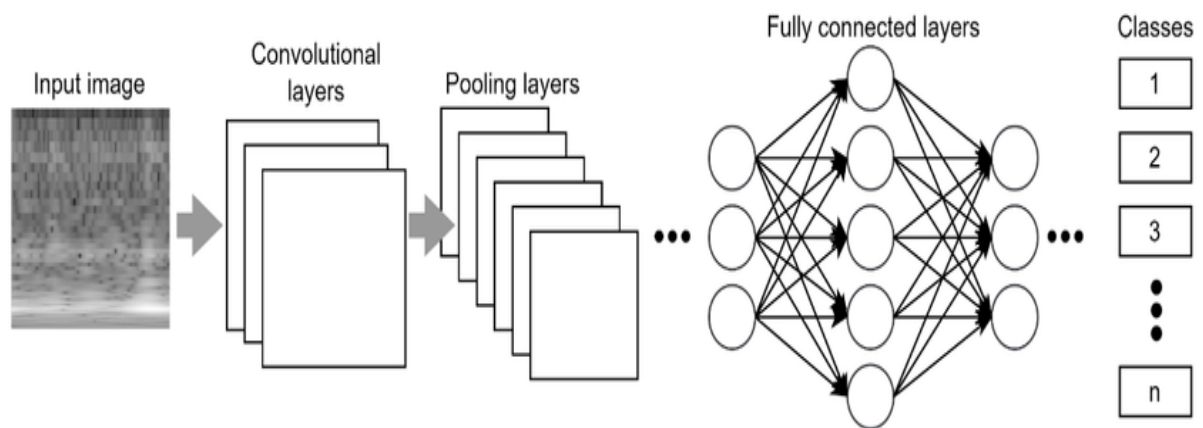


Figure 2.10: A fully connected neural network for classification [21].

6.3 Recurrent Neural Networks (RNNs)

The RNN were developed in 1980 and have recently garnered attention in the field of Natural Language Processing (NLP). RNNs belong to a specific category within the neural network family, tailored for sequential data or data that exhibit interdependence. Examples of sequential data include time series, audio recordings, and text, essentially any data with a discernible order. RNNs diverge from conventional FNN in their information processing mechanism. While FNN process information layer by layer, RNNs adopt a cyclic approach, iterating over the input data. To elucidate these disparities, let's examine the illustration in (Figure 2.11) below [22].

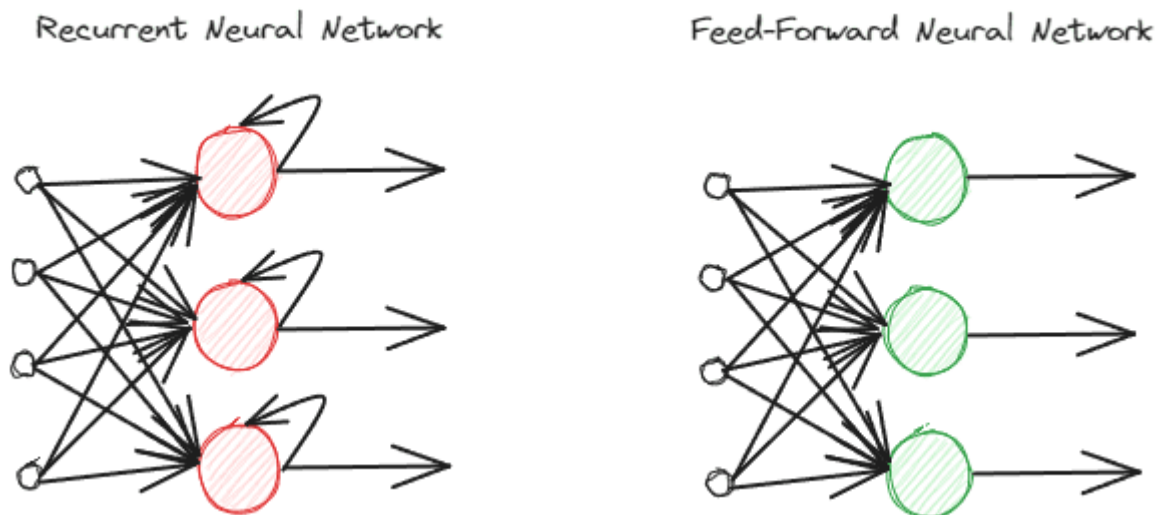


Figure 2.11: The difference between RNN and FNN [22].

As evident, the RNN model operates through a cyclic process during information processing. When handling data, RNNs take into account both the current and previous inputs, rendering them suitable for sequential data of any kind. For instance, in text data, consider the sentence "I wake up at 7 AM," with each word as an input. In a feed-forward neural network, upon reaching the word "up," previous words like "I," "wake," and "up" would be forgotten. However, RNNs utilize outputs from each word and loop them back, preventing such forgetting.

In the realm of Natural Language Processing (NLP), RNNs find extensive application in various textual tasks such as classification and generation. They are commonly employed in word-level applications like Part of Speech tagging and next-word generation. Delving deeper into RNNs within textual data, numerous types of RNNs exist, each tailored to specific requirements (Figure 2.12). [22]

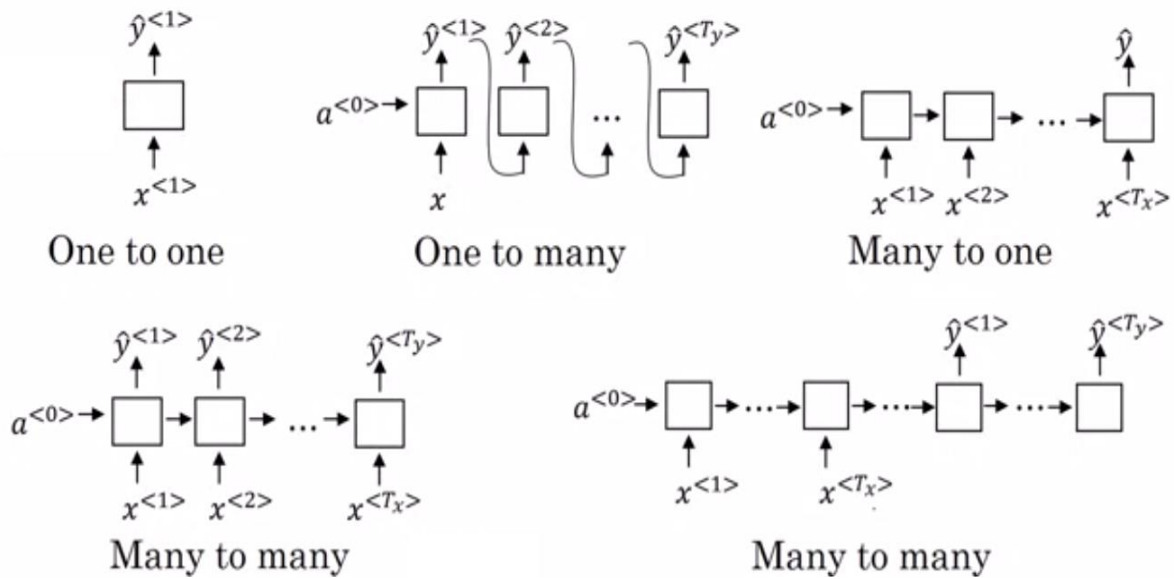


Figure 2.12: Types of RNN [22]

6.3.1 One to One:

In the domain of recurrent neural networks (RNNs), the most fundamental architecture is the one-to-one RNN ($T_x=T_y=1$). This architecture is characterized by processing a single input element ($T_x=1$) and generating a corresponding single output element ($T_y=1$), In the figure shown above [22].

6.3.2 One to Many:

In the domain of recurrent neural networks (RNNs), the one-to-many architecture (characterized by $T_x=1$ and $T_y>1$) excels in tasks demanding the generation of multiple outputs from a single input. A prominent application lies in music generation, where RNN models leverage this architecture to generate entire musical pieces (multiple outputs) from a solitary starting note (single input), [22].

6.3.3 Many to One:

In recurrent neural networks (RNNs), the "many-to-one" architecture ($T_x > T_y = 1$) signifies a scenario where the network processes a sequence of inputs (T_x) with a variable length and generates a single output ($T_y = 1$). This architecture is commonly employed in tasks like

sentiment analysis, where the model analyzes a sequence of words (text) to predict a single sentiment category (positive, negative, neutral), As shown in the illustration above [23].

6.3.4 Many-to-Many

As the notation implies, Many-to-Many RNN architectures process sequences of variable length (denoted by $T_x > 1$) and generate sequences of variable length (denoted by $T_y > 1$) as outputs. These architectures can be categorized into two main types:

- **Equal Input and Output Length ($T_x = T_y$):** In this type, the number of elements in the input sequence is identical to the number of elements in the output sequence. This scenario essentially translates to each element in the input having a corresponding element in the output. A common application of this architecture is Named Entity Recognition (NER), where the model identifies and classifies named entities like names of people, organizations, or locations within a text [23].
- **Unequal Input and Output Length ($T_x \neq T_y$):** Many-to-Many RNNs can also handle situations where the input and output sequences have different lengths. This is particularly useful in tasks like Machine Translation (MT), where the translated sentence might be shorter or longer than the original sentence. For example, the English phrase "I love you" translates to the shorter Spanish phrase "te amo." This demonstrates how MT models leverage the flexibility of unequal length Many-to-Many RNN architectures to produce variable length outputs based on the input language and its translation equivalent. As shown in the graphic above [23].

6.4 Long Short-Term Memory (LSTM) Network:

Long Short-Term Memory (LSTM) networks represent a specific architecture within the realm of recurrent neural networks (RNNs). They are specifically designed to overcome the vanishing gradient problem, a limitation hindering traditional RNNs in capturing long-term dependencies within sequential data. LSTM networks achieve this by employing memory cells equipped with gating mechanisms. These mechanisms enable a selective process of retaining or discarding information over time. Consequently, LSTM networks excel at modeling sequences that exhibit long-range dependencies. This makes them particularly adept at handling tasks like language modeling, machine translation, and time series prediction. By virtue of their

capabilities, LSTM networks have become a foundational element in deep learning, offering a potent tool for tackling sequential data characterized by intricate temporal dynamics [24].

At the core of LSTM is the cell state, a kind of conveyor belt that runs straight down the entire chain of the network. It allows information to flow relatively unchanged and ensures that the network retains and accesses important long-term information efficiently. As presents (Figure 2.13), LSTMs incorporate three types of gates, each playing a crucial role in the network's memory management:

- **Input Gate:** Determines which values from the input should be used to modify the memory.
- **Forget Gate:** Decides what portions of the existing memory should be discarded.
- **Output Gate:** Controls the output flow of the memory content to the next layer in the network [25].

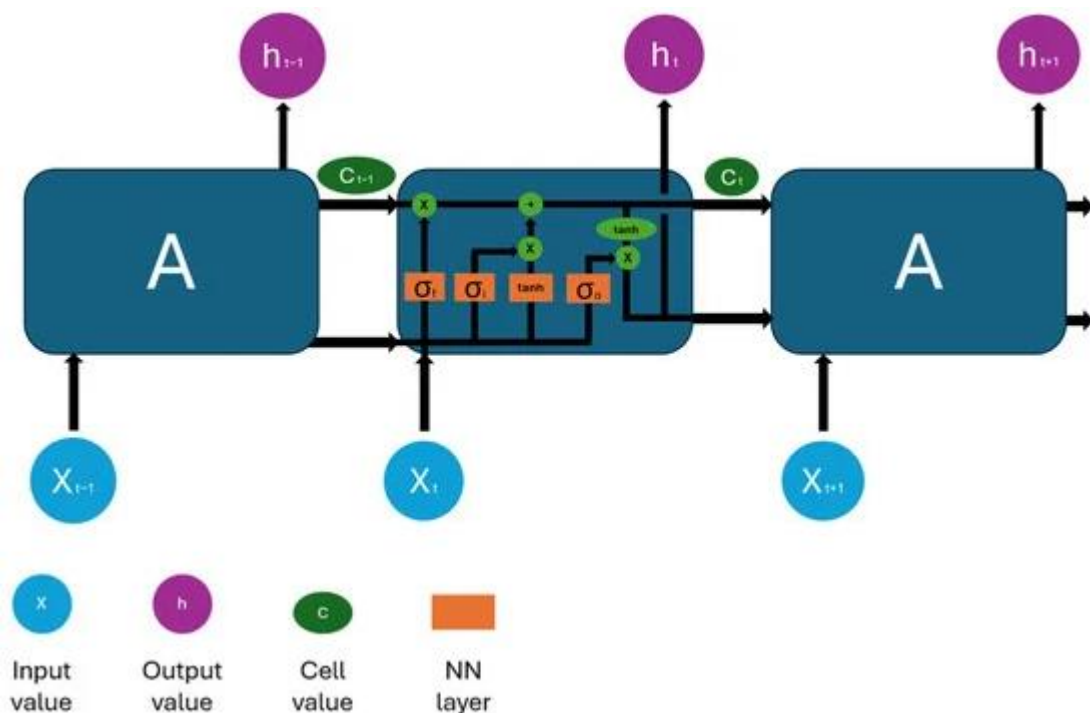


Figure 2.13: Schematic representation of the LSTM recurring layer [26].

7. Autoencoder

Autoencoders, a specialized class of algorithms, that are capable of learning efficient representations of input data without the need for labels. They are a type of ANN designed for unsupervised learning, tasked with compressing and effectively representing input data without

specific labels. This process, which involves encoding and decoding, forms the fundamental principle of autoencoders. The encoder transforms the input data into a reduced-dimensional representation, often termed as "latent space" or "encoding", while the decoder reconstructs the initial input from this representation. This two-fold structure enables the network to extract meaningful patterns from the data, facilitating the identification of essential features [27].

7.1 Architecture of Autoencoder in Deep Learning

The general architecture of an autoencoder includes an encoder, decoder, and bottleneck layer.

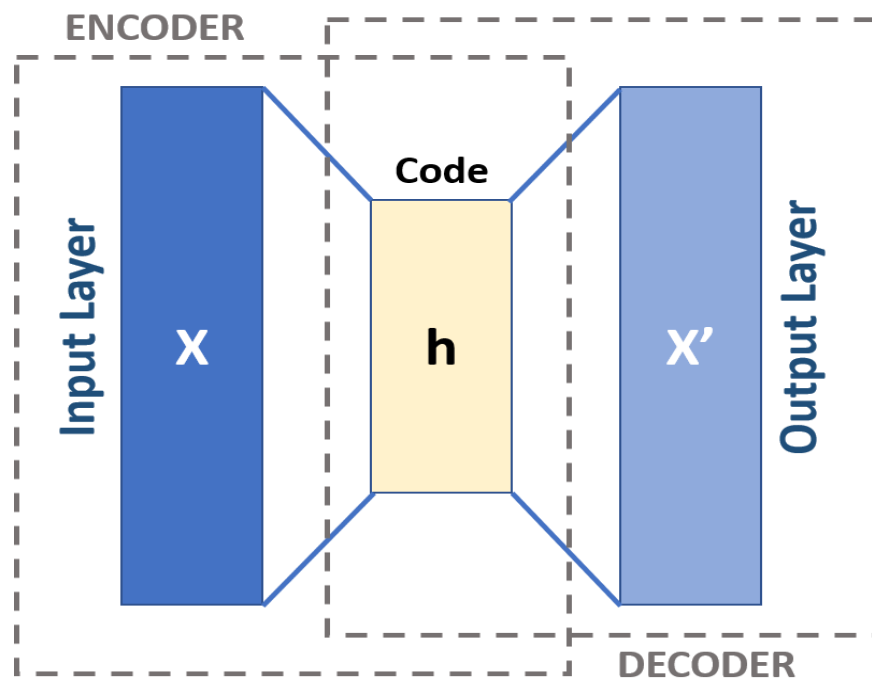


Figure 2.14: Autoencoder Schema [28].

As mentioned earlier, automatic encoding units can be fundamentally divided into three different components: encoding, bottleneck, and decoding units. The encoder layer in the encoding unit is typically a densely connected network. Encoding layers are used to compress incoming data into a latent space representation, resulting in a new representation of the data with fewer dimensions. The bottleneck layers handle the compressed representation of the data and determine the most important parts for reconstructing the data. They are sometimes referred to as the latent representation or latent variables. The decoding layer works to restore the compressed data to a representation with the same dimensions as the original data. This is done

using the latent space representation created by the encoding layer. Ultimately, these units can follow a simple neural network structure, closely resembling a single layer of the sensory perception used in multilayer perception, trained using backpropagation technique [29].

7.2 Types of Autoencoders

7.2.1 Undercomplete Autoencoder

The Undercomplete Autoencoder represents the most basic form of an autoencoder. In this scenario, there isn't an explicit regularization mechanism; however, we maintain the bottleneck's size consistently smaller than the original input to prevent overfitting. Such a setup is commonly employed as a dimensionality reduction method, surpassing PCA due to its capability to capture nonlinearities within the data Figure II.15. In an undercomplete autoencoder, each circle represents a neuron. The encoder translates the high-dimensional input to a low-dimensional encoded vector, and the decoder reverses the process.

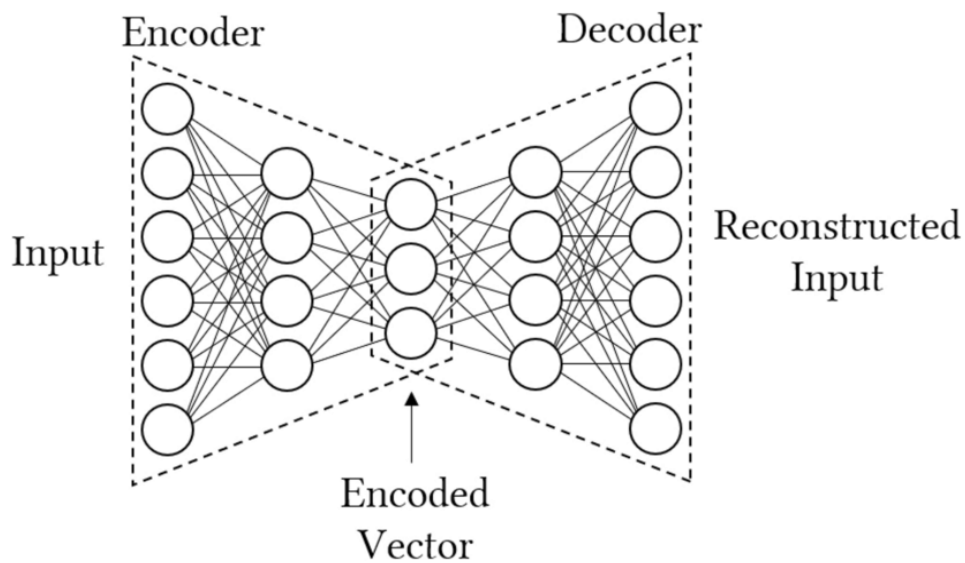


Figure 2.15: Undercomplete Autoencoder [30].

7.2.2 Sparse Autoencoder

Sparse Autoencoder bears resemblance to an Undercomplete Autoencoder, yet their principal contrast arises in the application of regularization. Unlike undercomplete autoencoders, sparse autoencoders may not mandate dimension reduction at the bottleneck; instead, they employ a loss function aimed at discouraging the model from activating all its neurons across various hidden layers (Figure 2.16). This form of penalty is commonly denoted

as a sparsity function, distinct from conventional regularization methods which primarily penalize weight magnitudes rather than the number of activated nodes.[30].

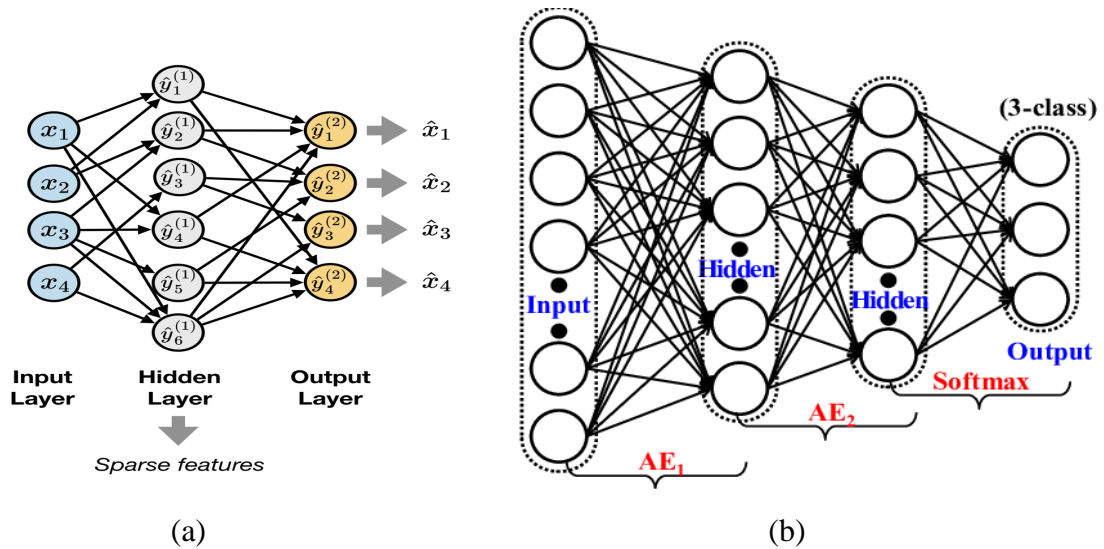


Figure 2.16: Sparse Autoencoder [30]. (a) with one hidden layer. (b) with two hidden layers as two single-layer autoencoders stacked together.

7.2.3 Contractive Autoencoder

Contractive Autoencoders are designed with the primary objective of ensuring that when similar inputs are provided to the model, their corresponding compressed representations also exhibit similarity. This means that regions in the input space, representing similar data points, should map to proximal regions in the output space, ensuring a compact representation. Mathematically, this objective is achieved by imposing constraints on the derivatives of the activations of the hidden layer when similar inputs are fed into the network. By constraining these derivatives to remain small for similar inputs, the autoencoder effectively learns to encode similar inputs into similar representations, thus enforcing a contractive behavior. In essence, this contractive property ensures that the autoencoder can preserve the local structure of the input data in its compressed representation, facilitating tasks such as data clustering or manifold learning [30].

7.2.4 Denoising Autoencoder

Denoising Autoencoders alter the input-output relationship of the model. For instance, the model might receive low-resolution corrupted images as input and aim to enhance their

quality in the output. To evaluate and enhance the model's performance, we would require labeled clean images to compare with the model's predictions FigureII.17. [30]

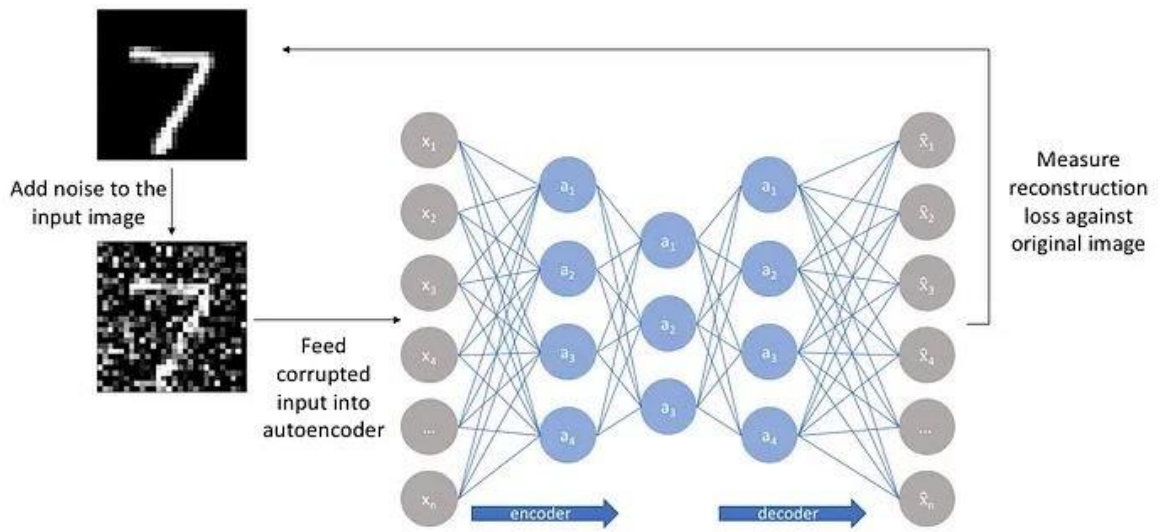


Figure 2.17: Illustration of Denoising Autoencoder [30].

7.2.5 Convolutional Autoencoder :

Convolutional autoencoders employ CNNs as their foundational elements. These autoencoders feature an encoder with several layers, which receive an image or grid as input and process it through various convolutional layers to create a compressed version of that input. The decoder, designed as a reverse counterpart of the encoder, works to decompress this representation and attempts to restore the original image. [30]

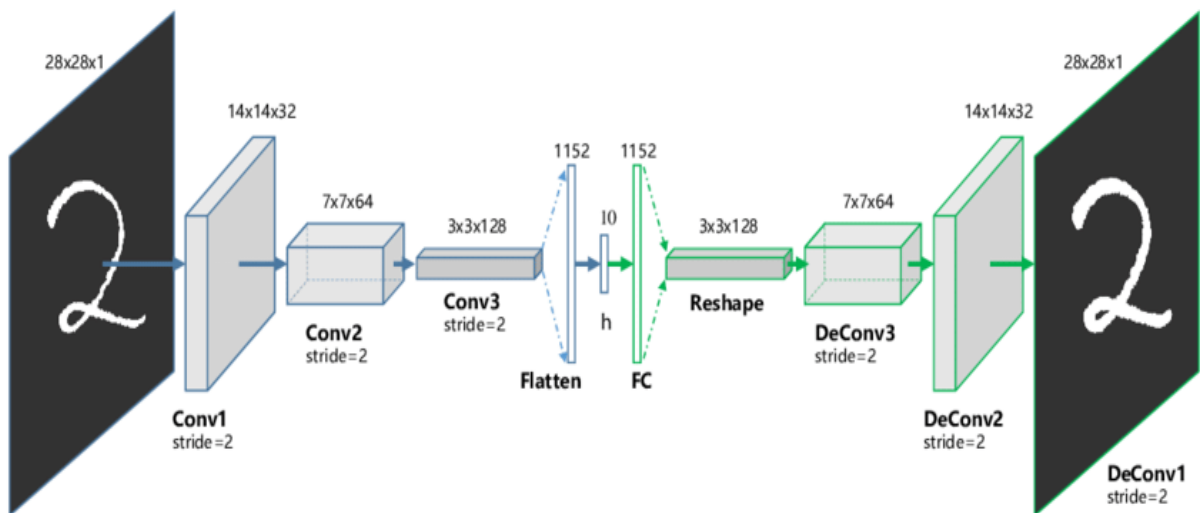


Figure 2.18: Deep Clustering with Convolutional Autoencoders [31].

7.2.6 Variational Autoencoder

Variational autoencoder models are built on substantial presumptions about the distribution of latent variables. These models employ a variational technique for learning latent representations, introducing an extra component of loss and employing a specialized estimator known as the Stochastic Gradient Variational Bayes estimator for training. This approach posits that the data originates from a directed graphical model and the encoder approximates the posterior distribution. Here, Φ and θ symbolize the parameters of the encoder (recognition model) and the decoder (generative model), respectively. Typically, the probability distribution of the latent vectors in a variational autoencoder aligns more closely with the training data than it does in a standard autoencoder [32].

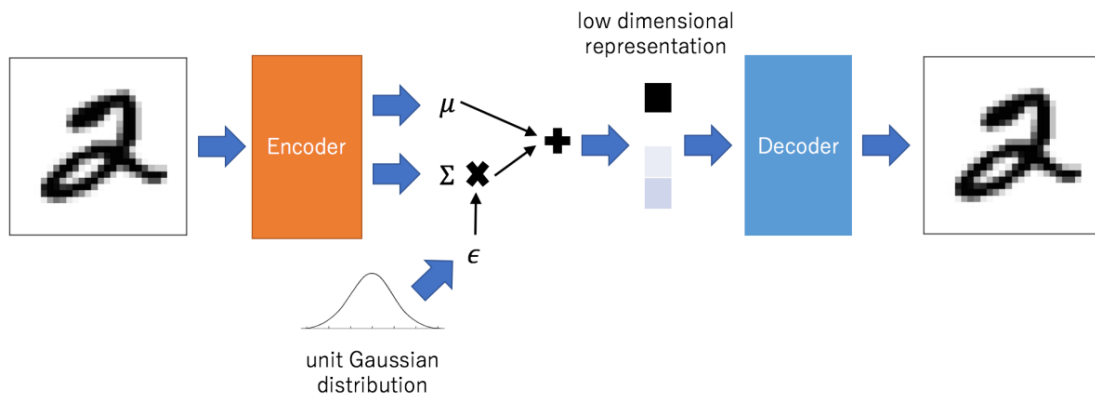


Figure 2.19: Schematic diagram of a variable autoencoder [32].

7.3 Applications of Autoencoders

Automatic encoding devices have found a wide range of applications across various fields due to their ability to efficiently learn data representations. In computer vision, they are widely used for tasks such as image denoising, reconstruction, and compression. By feeding corrupted or noisy images into an automatic encoder, it can learn how to generate clean and high-quality images as outputs. This is particularly useful in medical imaging to enhance diagnostic accuracy and in video surveillance to improve image clarity. In natural language processing, automatic encoders are used for tasks such as text generation, summarization, and

sentiment analysis. They learn how to encode textual data into lower-dimensional representations, which can then be decoded to reconstruct the original text or generate new content or convert text into natural speech. Additionally, automatic encoding devices have applications in anomaly detection, where they can identify unusual patterns or outlier values in data by accurately reconstructing normal data while producing errors in anomalous cases. These diverse applications highlight the versatility and effectiveness of automatic encoding devices across different domains [32].

CONCLUSION

In this chapter, we explored the theoretical foundations of deep learning, including its definition, history, and key components. We discussed the structure and function of biological neural networks that inspired the development of artificial neural networks (ANNs). Various models of artificial neural networks were presented, such as feedforward neural networks (FNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders, highlighting their structures and diverse applications. This comprehensive review demonstrates how deep learning models can be leveraged in natural language processing and other complex tasks, providing a solid foundation for the following chapters. By understanding these fundamental concepts, we can delve into practical applications and advanced techniques that make deep learning a powerful tool in various fields.

Chapter III:
Building Arabic Tacotron

1. Introduction

This chapter delves into practical implementation. We will use the models and techniques explained in previous chapter to build an Arabic text-to-speech system using the Tacotron 2 model.

2. Building Arabic Tacotron

Our work concerns the building of an Arabic TTS system based on the Tacotron model. It consists, firstly, on understanding then running the original Tacotron as it is. Then, secondly, adapt it to Arabic language as follow.

2.1. Tacotron

Tacotron is a TTS system that relies on deep learning techniques to generate natural speech from written text. It uses Recurrent Neural Networks (RNNs) to understand text sequences and analyze context, allowing it to create internal representations of the text. The system then converts these representations into audible sound waves, producing speech that closely resembles the human voice. Tacotron relies on deep learning to enhance the accuracy and quality of the generated speech by training neural networks on large amounts of textual and auditory data.[33]

The Tacotron system consists of two main parts. In the beginning the text is analyzed then it is converted into spectrograms, while the second part involves in converting the generated spectrograms into sound waves. Figure 3.1 presents the general architecture of the Tacotron 2 model.[34]

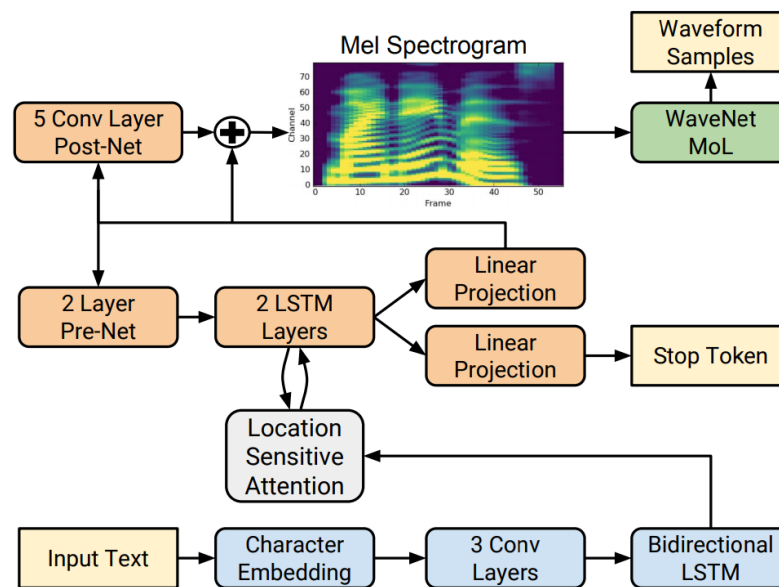


Figure 3.1: Tacotron 2 architecture [34]

2.1.1. The first part of Tacotron

It consists of the text conversion to spectrogram as follow.

The system begins by receiving the written text that needs to be converted into speech ("Input Text"). Each character in the text is then transformed into a numerical representation called embedding, allowing the system to numerically understand the text ("Character Embedding"). These numerical representations are subsequently processed through three layers of convolutional neural networks to extract relevant features from the text ("3 Conv Layers"). The extracted features are then refined through a bidirectional long-short-term memory (LSTM) network, which reads the features both left-to-right and right-to-left to comprehend the full context of the text ("Bidirectional LSTM"). To identify and focus on the most crucial parts of the text while considering spatial information, a location-sensitive attention mechanism is employed ("Location Sensitive Attention"). The processed features are then passed through two additional LSTM layers, further enhancing the model's understanding of the context and temporal patterns of the text ("2 LSTM Layers"). Next, linear projections are used to convert these features into a Mel Spectrogram, a spectral representation of the anticipated audio file. Another linear projection generates the stop token, which signals the end of the text ("Linear Projection" and "Stop Token"). The Mel Spectrogram, which serves as an intermediate stage before being converted into the final audio form, provides a visual spectral representation of the expected audio output ("Mel Spectrogram").

2.1.2. The second part of Tacotron

In this second part, the system refines the Mel spectrograms through five convolutional layers, enhancing their quality and making the features more precise and clearer for the subsequent processing steps ("5 Conv Layer Post-Net"). Finally, the refined Mel spectrograms are converted into audio waveforms using the WaveNet Mixture of Logistics (MoL) model. This final step generates the synthesized speech, producing natural and high-quality audio from the processed spectrograms ("WaveNet MoL").

2.1.3. Running Tacotron:

To run the indicated model, first, several hardware and software installation steps were needed like:

- Personal Computer (PC) to run the model. We used a LENOVO PC, of “Intel(R) Core (TM) i5-7300U CPU @ 2.60GHz 2.71 GHz” processor, 238 SSD memory, and RAM of 8 GB. It operates with Windows 10 pro 64bit.
- Python programming language. Python is a powerful, easy-to-learn, and open-source programming language. It has several libraries, each with a specific function. In addition, it is a very rich language and is increasingly being used. As part of our work, we used the libraries presented in table 3.1:

Table 3.1: Python libraries used in our work

the libraries	The explanation
os	Interacting with the operating system, such as handling files and directories.
io	Handling input and output operations, such as reading and writing data.
gdown	Downloading files from Google Drive using direct links.
argparse	Parsing and handling command-line arguments.
numpy	Working with multi-dimensional arrays and mathematical functions, performing scientific computing.
tqdm	Displaying progress bars for long-running operations to track and estimate remaining time.
matplotlib.pyplot	Creating charts and plots for data visualization.
glob	Pattern matching for file paths to find files and directories.
shutil	Copying, moving, and deleting files and directories.
datetime	Managing dates and times, creating, formatting, and parsing dates.
re	Searching and manipulating text using regular expression patterns.
unidecode	Converting Unicode text to ASCII for simplification.
TensorFlow	Developing machine learning and deep learning models, building and training neural networks.

inflect	Generating linguistic text and changing word forms, such as converting numbers to words.
zip file	Working with ZIP files, creating, and extracting compressed files.
torch	Machine learning and deep learning using PyTorch, building and training models.
finfo	Providing information about floating-point numeric types, such as precision and values.
scipy	Scientific and technical computing, such as integration, spectral analysis, and statistical analysis.
librosa	Analyzing audio and music, loading, analyzing, and processing audio signals.
pillow	Image processing, opening, editing, and saving images in various formats.

- Visual Studio (VS) Code : VS Code is a free, open-source code editor developed by Microsoft. It supports numerous programming languages and features syntax highlighting, debugging, and version control integration. Highly customizable, it allows users to install extensions and themes to fit their needs. Its user-friendly interface and powerful features make it a popular choice for developers
- Google Colab Pro: It is an advanced version of Google Colab that requires a subscription and offers enhanced features for improved performance on programming projects. With Colab Pro, you can access better computing resources, such as high-performance GPUs and advanced CPUs, which allow for faster and more efficient code execution. Colab Pro also provides longer runtime sessions and greater storage capacity, making it suitable for handling large datasets and training deep learning models. These features enable you to accelerate model development and enhance your overall experience when working on complex projects in Google Colab.
- Training database: to run the original Tacotron model we use the LJ-Speech database. It contains public domain speeches comprises 13,100 short audio clips featuring a single speaker reading excerpts from seven non-fiction books. Each clip is accompanied by a transcription. The clips range in length from 1 to 10 seconds, with a total duration of approximately 24 hours. The texts, published between 1884 and 1964, are in the public domain. The audio recordings were made in 2016-17 by the LibriVox project and are also in the public domain. [35]

After finishing the previous preparation, to synthesis using Tacotron, we follow the below steps:

- a) Pre-training preparations

- Processing audio files: We need to arrange the audio files with numbers from one to our last file. Then simplifying the audio by removing the silent part of the audio files and modifying some of the settings, including: subtype='PCM_16' / top_db=20 / sample rate of 22050 Hz. After that, we update the meta data for audio files. At the end, we compressed the final file and uploaded it to Google Drive to work on it from Colab

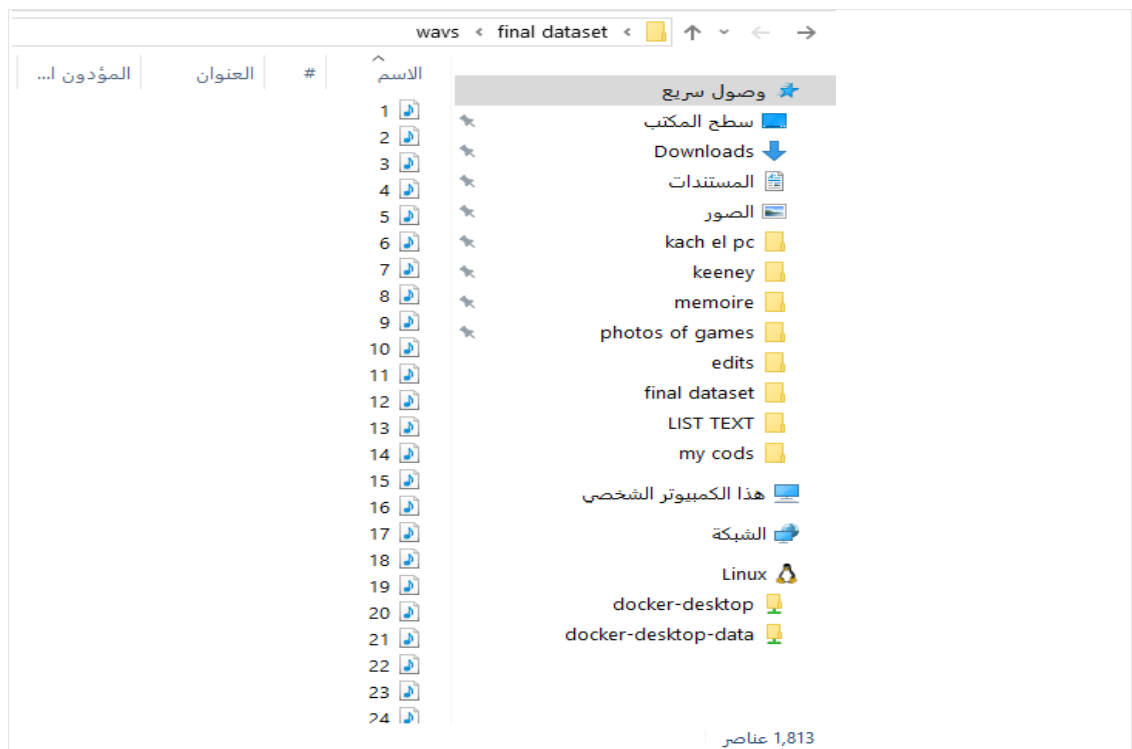


Figure 3.2: Preparing audio files.

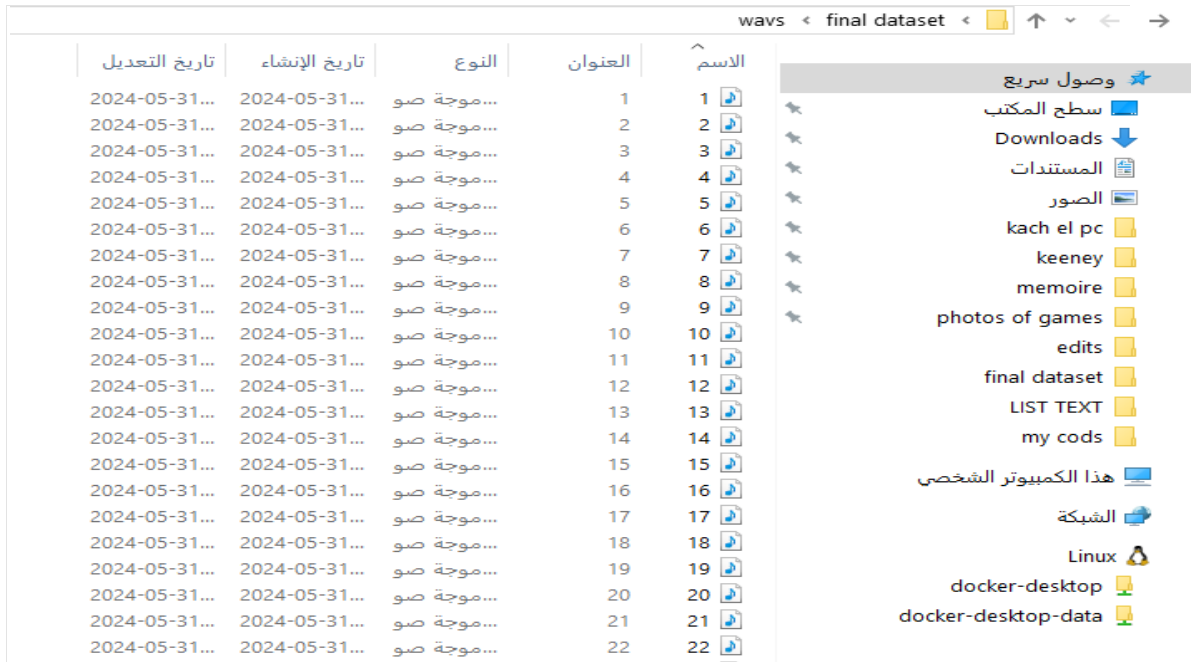


Figure 3.3: Update metadata for audio files

- preparing transcript: The written texts, which we will work with in the form, start with the path of the audio files, then the text corresponding to them, are separated by the sign "|"
- b) Setting the GPU
- c) The first step in running Tacotron is to activate the graphics processor, in which we work with, “L4 GPU” as shown in figure 3.4.

```

+-----+
| NVIDIA-SMI 535.104.05           Driver Version: 535.104.05   CUDA Version: 12.2   |
+-----+
| GPU Name          Persistence-M | Bus-Id      Disp.A | Volatile Uncorr. ECC | |
| Fan  Temp  Perf    Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M. |
|                               |                  |           |    MIG M.           |
+-----+-----+
|   0   NVIDIA L4           Off   | 00000000:00:03:0 Off |             0         | |
| N/A   53C   P8             12W / 72W | 1MiB / 23034MiB |   0%      Default   |
|                               |                  |           |                     |
+-----+-----+

+-----+
| Processes: |
| GPU  GI  CI       PID  Type  Process name                        GPU Memory |
|   ID  ID                                     |            Usage |
+-----+-----+
| No running processes found |
+-----+
    
```

Figure 3.4: Graphics Properties

- d) Connect your Colab account to your Google Drive account.[36]
- e) Import the main repository of the TACOTRON model from the GitHub account and load the pretrained model.
- f) Import the audio files from Google Drive and place them in the wavs folder in the model.



Figure 3.5: Load dataset for the original model

- g) Upload the text document and verify the paths in it to link them with the sentences written in it during training.
- h) Configure the model parameters and adjust some optional settings:
 - model filename: name of the resulting training file
 - Training file: The file path of the text file, which includes the path of each audio file attached to its corresponding sentence, it is considered after being converted to its original location in the model.
 - hparams.A_ = 1e-4: Learning rate represents the initial learning rate and the learning speed. The lower it is, the longer the training takes but it yields more accurate results.
 - hparams.batch_size: It is a parameter in the training model that determines the batch size or the number of samples processed in each training epoch.
 - use_cmudict: Enable English language dictionary or disable English language dictionary.
 - hparams.epochs: Number of training epochs

output_directory: The location of the final training file.

5. Configure the model parameters.

Your desired model name:
 model_filename: "train"

Upload your transcription / text to TTS-TT2/filelists and right click -> copy path:
 training_file: "/content/TTS-TT2/filelists/list.txt"

Lower learning rates will take more time but will lead to more accurate results:
 hparams.A_: 1e-4

Your batch size, lower if you don't have enough ram:
 hparams.batch_size: 10

use_cmudict:

Your total epochs to train to. Not recommended to change:
 hparams.epochs: 200

Where to save your model when training:
 output_directory: "/content/drive/MyDrive/colab/outdir"

Figure 3.6: Basic settings for the model

i) Convert the .wav files to Mel spectrograms and check the files:

This is one of the most important stages in the model, as it converts audio files into a Mel spectrogram [37]

6. Convert the .WAV files to Mel spectrograms and check the files.

Show code

Generating Mels
 100% ██████████ 1813/1813 [00:13<00:00, 148.01it/s]

```

Checking for missing files
Checking Training Files
Checking Validation Files
Finished Checking
  
```

Figure 3.7: Transformation of sound waves to Mel spectrogram

j) Begin training:

At this stage, information about the selected settings are provided as presents Figure3.8.

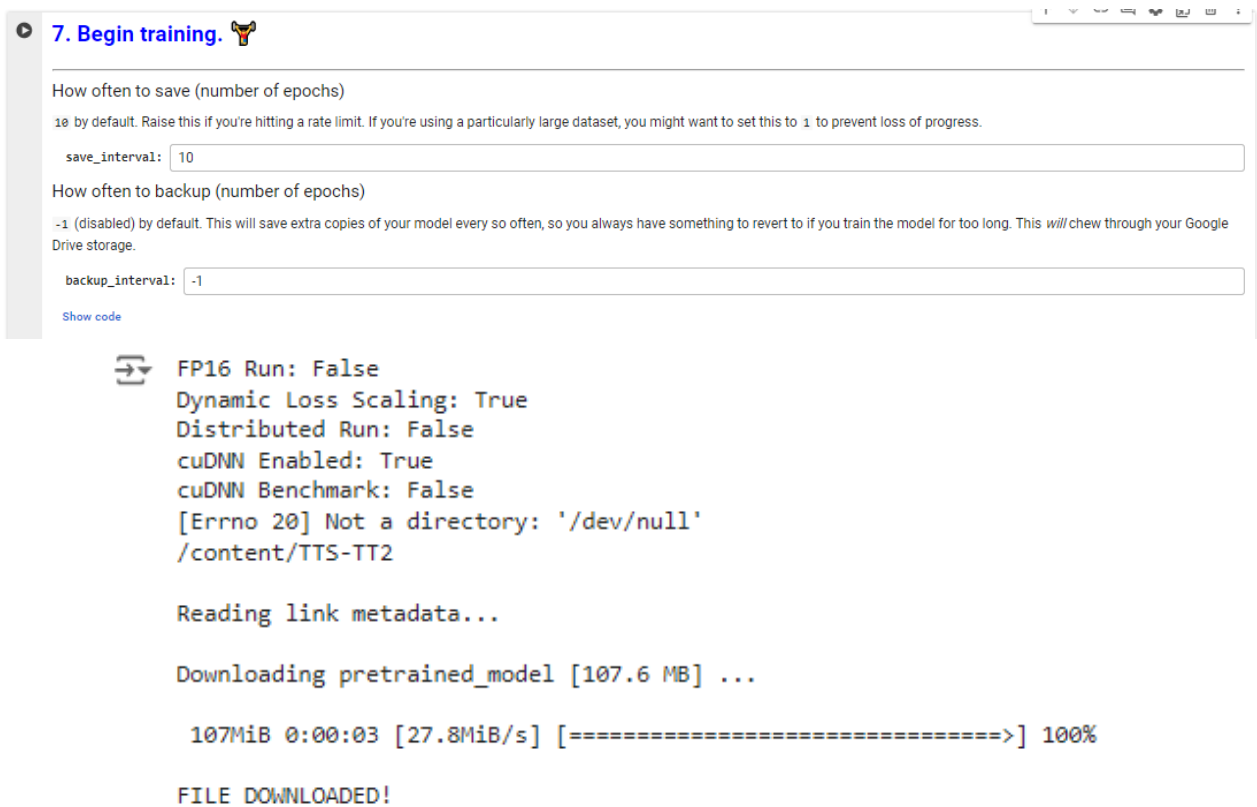


Figure3.8: start training

- k) After the training is completed, our model is saved in the specified path. Then, we go to Google Drive to get the link of the trained model to use in the next step, which is HiFi GAN, to enhance the model and generate the audio.

HiFi-GAN_{v2}: (High-Fidelity Generative Adversarial Networks): is an advanced model in the field of speech synthesis used to convert intermediate outputs from text-to-speech systems, such as Tacotron, into high-quality final audio waveforms. HiFi-GAN transforms Mel Spectrograms into natural and smooth sound, achieving a balance between accuracy and generation speed.

HiFi-GAN is based on a Generative Adversarial Network (GAN) architecture, consisting of a Generator and a Discriminator. The generator works to create sound from intermediate inputs, while the discriminator attempts to distinguish between the generated sound and real sound. This interaction between the generator and the discriminator enables the model to produce highly realistic sounds.

After providing the model's ID and running the program, the necessary files are downloaded. A field will then appear where you can input the text you want to convert to speech. We wait for the Mel Spectrogram to be synthesized, and then the final audio is generated.

2.2. Arabic adaptation of Tacotron

After successfully synthesizing speech using the original model in English, we have completed half of the work we aim to achieve. Now, we move on to adapting the system for Arabic through several steps discussed in the next points.

- Sound DATABASE:

Since Tacotron was trained on a large English corpus, we had to find an equivalent Arabic dataset. We chose the 'Arabic Speech Corpus' [38]

This Speech corpus has been developed as part of PhD work carried out by Nawar Halabi at the University of Southampton. The corpus was recorded in south Levantine Arabic (Damascian accent) using a professional studio. Synthesized speech using this corpus has produced a high quality and natural voice.

After downloading the corpus, we repeated the same previously explained steps to create the compressed file then upload it to Google Drive.

- Preparing the transcription file:

Along with the sound database, preparing the transcription rules and files is very important step in speech synthesis. It consists to prepare the text for the Neural network model to understand. In this part we have modified, Tacotron, programing files that deals with text form and their related audio files.

As figure 3.10 shows, the modified files concerns, deals with number conversion, abbreviation understanding, spaces and special character cleaning, and most importantly changing the writing style according the Arabic reading rules. The characters used in the phonetic transcription of Arabic letters are presented in table 3.2

All these modifications mentioned are at the level of the "text" folder which is in the form's repository

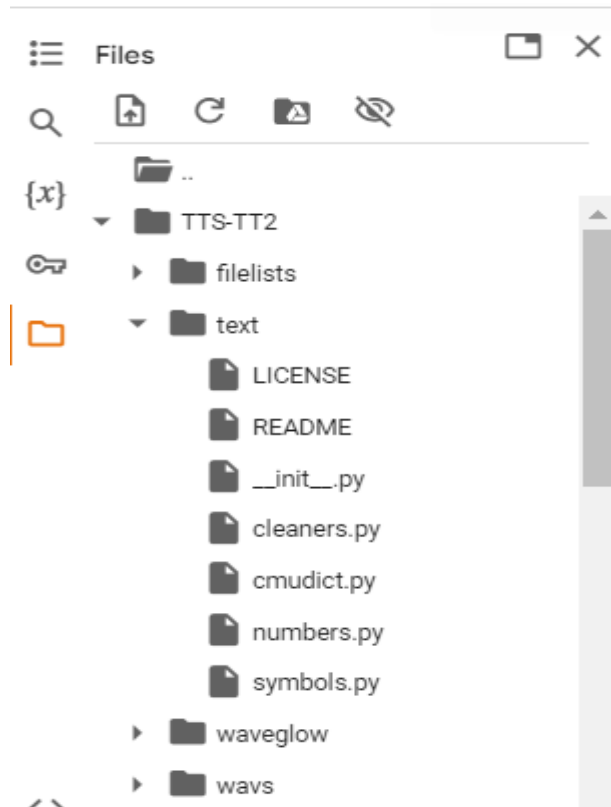


Figure 3.9: Part of the basic model warehouse structure

Table 3.2: Arabic letters and their phonetic writing in our model

The letter in Arabic	The phoneme	The letter in Arabic	The phoneme
الهمزة (ء)	('a)	العين (ع)	(')
الألف (ا)	a	الغين (غ)	gh
الباء (ب)	b	الفاء (ف)	f
التاء (ت)	t	القاف (ق)	q
الثاء (ث)	th	الكاف (ك)	k
الجيم (ج)	j	اللام (ل)	l
الحاء (ح)	H	الميم (م)	m
الخاء (خ)	kh	النون (ن)	n
الدال (د)	d	الهاء (ه)	h
الذال (ذ)	dh	الواو (و)	w
الراء (ر)	r	الياء (ي)	y
الزاي (ز)	z	التاء المربوطة (ة)	t
السين (س)	s	الألف المقصورة (ى)	~
الشين (ش)	sh	الفتحة ()	a
الصاد (ص)	S	الضمة ()	u
الضاد (ض)	D	الكسرة ()	i
الطاء (ط)	T	السكون ()	don't put anything
الظاء (ظ)	Z	الثدة ()	Repeat the same letter

- Apply the same steps to the model:

After preparing the sound and text data, we follow the same steps mentioned in the original model until we get the training model. Then, We take the resulted file model file and proceed to the second step, which involves working with HIFI GAN.

In this case, everything becomes ready for sound generation from the text after entering the Arabic text with diacritics, or with phonetic transcription, for those who are familiar with it. Using the later form they can enhance expression to produce a clearer and more accurate sound from the text.

Figures 3.10 and 3.11 present the training curve and the generated spectrogram of some synthesized sentences, entered using both, diacritic Arabic text and phonetic writing

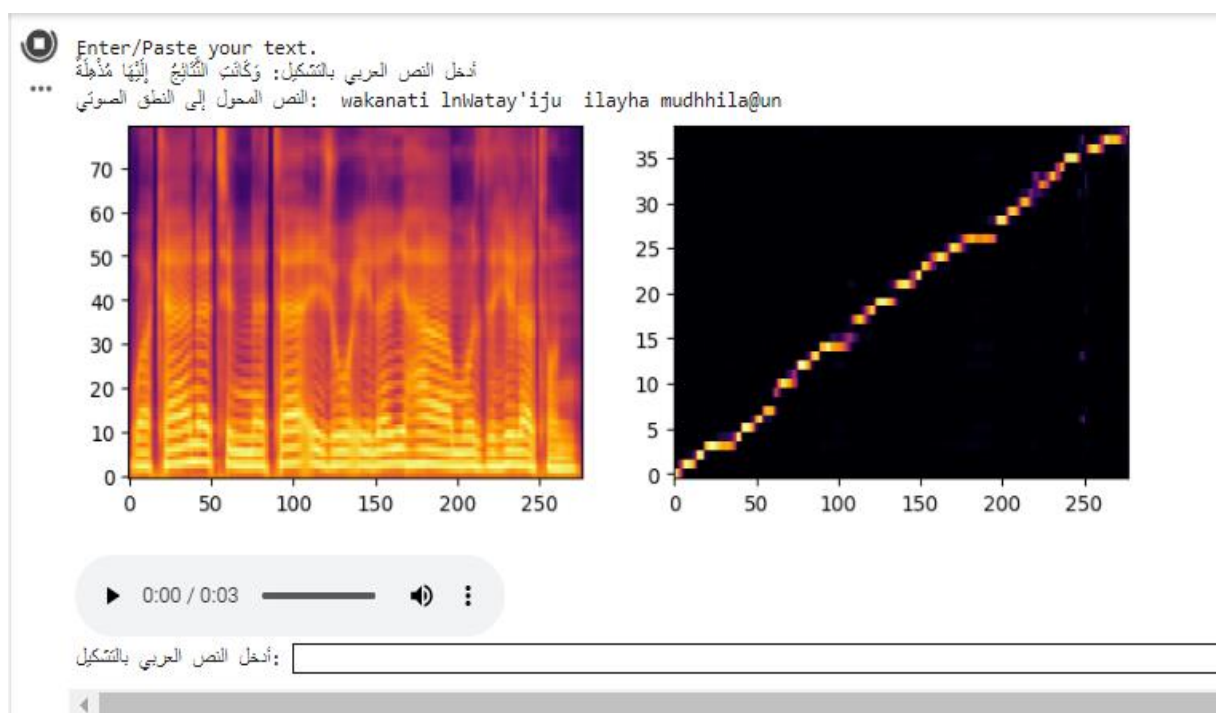


Figure 3.10: Spectrogram and training cure of a synthesized of audio from an Arabic text

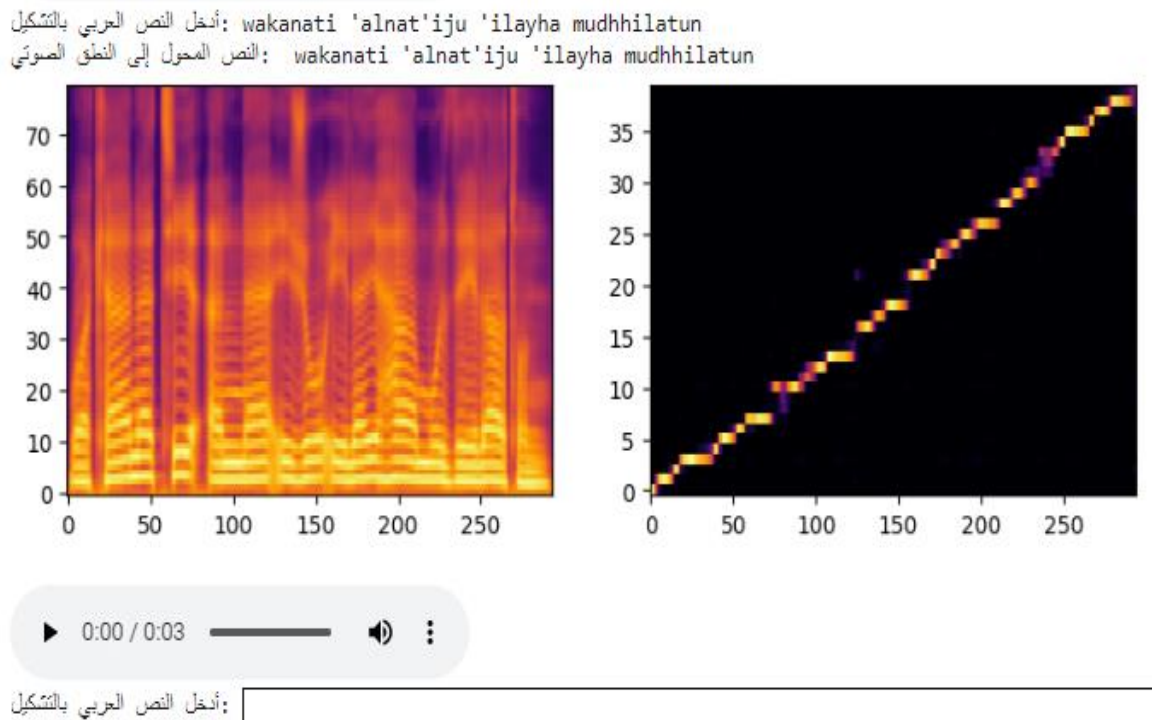


Figure 3.11 : Spectrogram and training cure of a synthesized of audio from a phonetic writing

3. Evaluation and results

Evaluation is a necessary step in building a text-to-speech (TTS) system. Therefore, to build a comprehensive and accurate assessment, we will need to diversify the test examples. For this reason, we chose a set of words to build the understanding rates of the speech produced by the model, and a set of sentences also to build the accuracy of the resulting nature after being heard by real people.

We performed our evaluation to assess two speech qualities Intelligibility and naturalness as will detailed in the next point. We chose to conduct the evaluation on the Google Form platform so that it reaches the largest possible number of people, to get more transparent evaluation. This reached to 25 test participants (13 male and 12 female), aged between 15 and 35 .[39]

3.1. Evaluating speech intelligibility

To assess the synthesized speech intelligibility, we conducted two testes.

- I the first, the participants had to listen to 10 synthesized words, the chose the correct word out of three close options. The goal of this test is to know how well the synthesized sound is distinguished from other close ones. The synthesized words and their clarity

result are presented in table 3.3

Table 3.3: Percentages of words used in the evaluation

Words	Phonetic writing	Comprehension rate
كتاب	<i>kiytaabun</i>	100 %
سيارة	Sayaaratun	100 %
شجرة	Shajaratun	100%
بيت	Baytun	100%
مدرسة	Madrasatun	75%
هاتف	Haatifun	85%
قمر	<i>Kamarun</i>	100%
بحر	baHrun	45%
جبل	Jabalun	100%
طائرة	Ta'iratun	70%

In general, most of words were understood by the listeners, except for some words whose letters are difficult to pronounce, and with the presence of words with similar context and sound, it make it difficult to choose the correct word among the options, such as:

- مدرسة [Madrasatun]: In this word, the model faced difficulty in applying the appropriate diacritics to the word, and also a difficulty to distinguish between the letter “س” and the letter “ص”
 - هاتف [Haatifun]: In the word "phone", the only difficulty we faced was distinguishing between the letter "ه" and the letter "ح", as in most cases the letter "ه" is pronounced as "ح".
 - طائرة [Ta'iratun]: In this word, the problem lies in the pronunciation of the letter “ط” on the basis that it is “ت”
- In the second test, five other synthesized words were presented to the participants, in which they had to listen the rewrite what they heard exactly. the Words used in this second test and their results are presented in table 3.4.

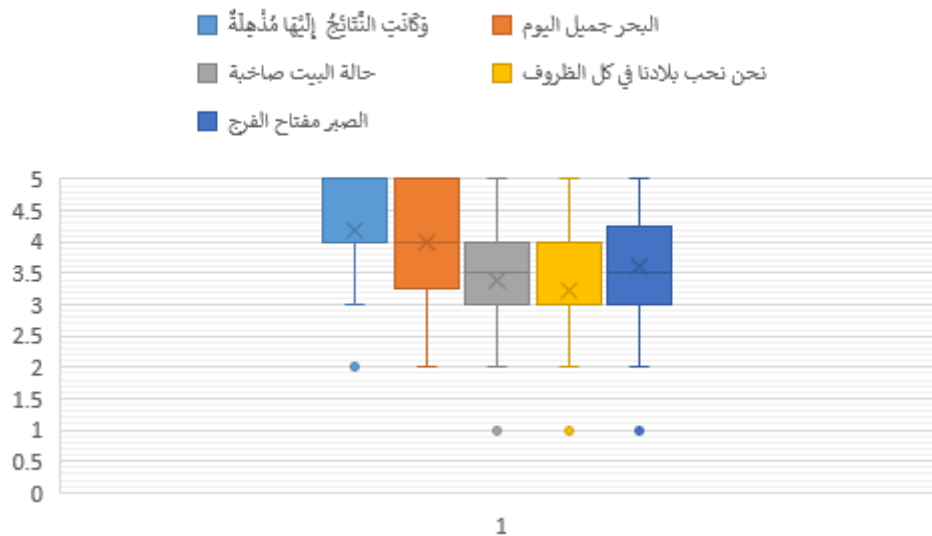
Table 3.4: Evaluate the words of the second test in terms of clarity

Words	Phonemic writing	Comprehension rate
ساعة	Sa: `atun	42.1%
كلب	Kalbun	94.7%
سماء	Sama'un	63.2%
زجاج	Zujajun	89.5%
باب	Baabun	89.5%

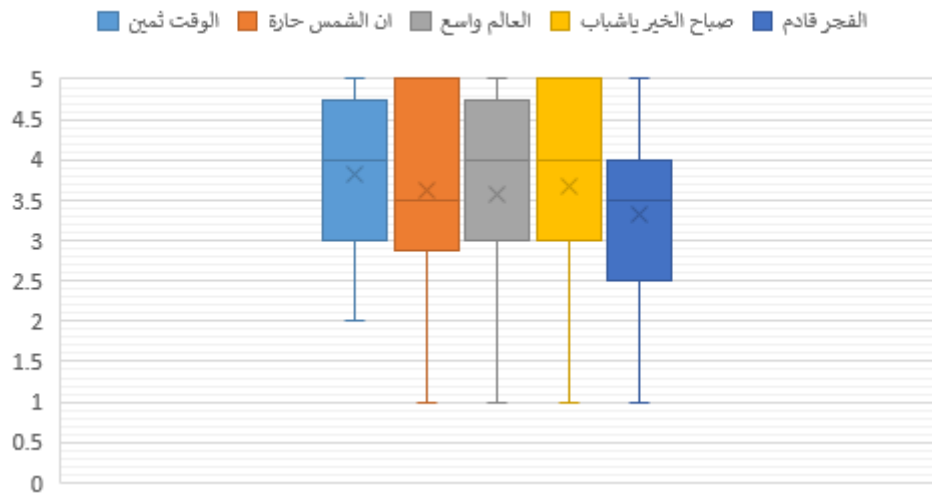
In this test, the assessment was difficult because we encountered spelling errors, which made it difficult to determine whether the problem was in understanding compound words or how to express them. We also noticed a decrease in intelligibility compared to the first test.

3.2. Evaluating speech naturalness

In the naturalness test, listeners listened to ten sentences with a variety of words synthesized by the model. They were asked to give a rating from 1 to 5 on how they felt the nature of the sound was. These ratings were then averaged to evaluate the listeners' understanding of the text and its nature, and the results of this test were summarized in a box plot graph as shown in the Figure 3.12 (a), (b).



(a)



(b)

Figure 3.12: Box plot graph of evaluating the sentences

After evaluating the model and studying the testers' opinions about its accuracy, a rating of 3.64 out of 5 was obtained. The results obtained are acceptable compared to the initial speech synthesis model, but the model still needs improvements. As shown in the graph, there are sentences that received a good rating, such as:

"وَكَاثَتِ النَّتَائِجُ إِلَيْهَا مُذْهَلَةً" The sound was closer to the human voice and clear to listeners.

On the other hand, there are sentences that received a low rating, such as:

"الفجر قادم". The problem with this sentence was the letter "ق". The word "مع" is pronounced because it is followed by "ا". When the two characters met, the model was difficult to assemble, so production quality declined.

Conclusion

In this chapter, we addressed the process of adapting the original Tacotron model for the Arabic language. By analyzing and following the steps of the original model, we understood its working mechanism. Consequently, we were able to input the specific settings for the Arabic language and generate a model trained in Arabic that comprehends words through phonemes. The presence of a massive network of deep learning networks makes the work

more complex but more precise, meaning that better results can be achieved if we ensure the enhancement of the audio files library and intensify the training process.

General Conclusion and Perspectives

In this thesis, we have developed an advanced Text-to-Speech (TTS) system tailored specifically for the Arabic language utilizing the Tacotron model, which represents one of the latest advancements in deep learning for speech synthesis. The main objective was to create a system capable of converting written Arabic text into natural and intelligible speech.

The process of adapting the Tacotron model to the Arabic language involved significant modifications to accommodate the unique linguistic features of Arabic. The adaptation included converting text to phonemic representations and generating spectrograms that are further processed into audible speech using the WaveNet model.

The developed system was evaluated based on speech intelligibility and naturalness, receiving a satisfactory rating of 3.66 out of 5. This demonstrates the model's capability to produce clear and understandable Arabic speech, although further refinements are necessary to improve accuracy and naturalness.

While the results are promising, several challenges were encountered:

- **Pronunciation and articulation:** Certain Arabic sounds, which are not present in other languages, posed difficulties in accurate pronunciation and required complex rule-based adjustments;
- **Special names and technical terms:** Proper handling of unique names and technical terminology remains a challenge due to their absence in general dictionaries.

By addressing these challenges, we aim to develop a more robust and versatile Arabic TTS system that meets the needs of a diverse user base, ultimately contributing to more effective and accessible human-computer interactions.

Continuous advancements in artificial intelligence and deep learning are expected to significantly enhance the quality and performance of developed TTS system. Future research and development will focus on:

- Expanding the dataset to include a broader range of dialects and accents.
- Refining the model to better handle complex texts and special names.
- Improving the naturalness and expressiveness of synthesized speech.

This work underscores the potential of deep learning in revolutionizing speech synthesis, paving the way for more sophisticated and user-friendly applications in various domains.

Bibliography

- [1] BETTAYEB, Nadjla; GUERTI, Mhania. Elaboration d'un Système de Synthèse par Sélection d'Unités en Vue de la Récitation du Saint Coran. PhD thesis, Ecole Nationale Polytechnique, Algiers, 2021.
- [2] KHAN, Rubeena A.; CHITODE, Janardan Shrawan. Concatenative speech synthesis: A review. *International Journal of Computer Applications*, 2016, 136.3: 1-6.
- [3] VAN SANTEN, Jan PH, et al. (ed.). *Progress in speech synthesis*. Springer Science & Business Media, 2013.
- [4] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. *nature*, 2015, 521.7553: 436-444.
- [5] DENG, Li, et al. Deep learning: methods and applications. *Foundations and trends® in signal processing*, 2014, 7.3-4: 197-387.
- [6] ALOM, Md Zahangir, et al. A state-of-the-art survey on deep learning theory and architectures. *electronics*, 2019, 8.3: 292.
- [7] TOUZET, Claude. les réseaux de neurones artificiels, introduction au connexionnisme. Ec2, 1992.
- [8] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep learning*. MIT press, 2016.
- [9] LAU, Mian Mian; LIM, King Hann. Review of adaptive activation function in deep neural network. In: 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). IEEE, 2018. p. 686-690.
- [10] BELZIANE, Fayçal, BNGHARBIA, Billel, et al. « Implémentation d'un réseau de neurones artificiels sur une plateforme FPGA », Master thesis, Université de Medea. 2018.
- [11] JHA, Girish Kumar. *Artificial neural networks and its applications*. IARI, New Delhi, girish_iasri@rediffmail.com, 2007.
- [12] MINSKY, Marvin; PAPERT, Seymour. *An introduction to computational geometry*. Cambridge tiass., HIT, 1969, 479.480: 104.
- [13] KHUDAYAROV, Muzaffar; NORMAMATOV, Nuriddin. Power system steady state calculations using artificial neural networks. In: E3S Web of Conferences. EDP Sciences, 2020. p. 01102.
- [14] BENGIO, Yoshua; GOODFELLOW, Ian; COURVILLE, Aaron. *Deep learning*. Cambridge, MA, USA: MIT press, 2017.
- [15] CHOWDARY, G. Jignesh, et al. Machine learning and deep learning methods for building intelligent systems in medicine and drug discovery: A comprehensive survey. *arXiv preprint arXiv:2107.14037*, 2021.

- [16] RAMOUL, Mohamed Islem« Development of an intelligent system for the detection and classification of softwood seed seeds. Doctoral thesis» University of Quebec at Trois-Rivières. 2021.
- [17] ALQADI, Ziad; JABBER, Qazem. Digital Image Cryptography using Index Keys. 2023.
- [18] IOSIFIDIS, Alexandros; TEFAS, Anastasios (ed.). Deep learning for robot perception and cognition. Academic Press, 2022.
- [19] ABINAYA, R., et al. Acoustic based scene event identification using deep learning cnn. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12.5: 1398-1405.
- [20] DAVENPORT, Thomas H.; PATIL, D. J. Data scientist. Harvard business review, 2012, 90.5: 70-76.
- [21] Muhammad Arham, Machine Learning Engineer at Vyro on September 28, 2023 in Machine Learning.
- [22] SUNARJO, Macellino Setyaji, et al. High-performance convolutional neural network model to identify COVID-19 in medical images. Journal of Computing Theories and Applications, 2023, 1.1: 19-30.
- [23] THUAN, Nguyen Duc; HONG, Hoang Si. HUST bearing: a practical dataset for ball bearing fault diagnosis. BMC research notes, 2023, 16.1: 138.
- [24] FANNI, Salvatore Claudio, et al. Natural language processing. In: Introduction to Artificial Intelligence. Cham: Springer International Publishing, 2023. p. 87-99.
- [25] SHERSTINSKY, Alex. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 2020, 404: 132306.
- [26] TURŠIČ, Niko; KLANČNIK, Simon. Tool Condition Monitoring Using Machine Tool Spindle Current and Long Short-Term Memory Neural Network Model Analysis. Sensors, 2024, 24.8: 2490.
- [27] BERAHMAND, Kamal, et al. Autoencoders and their applications in machine learning: a survey. Artificial Intelligence Review, 2024, 57.2: 28.
- [28] Michela Massi. Autoencoder schema
https://commons.wikimedia.org/wiki/File:Autoencoder_schema.png [accessed 2024/06/12].
- [29] POADE, Donna; CRAWFORD, Russell M. Trickle or Torrent? A Novel Algorithmic Approach to Reclaim Successful Academic Writing in the Face of Artificial Intelligence. Brock Education Journal, 2024, 33.1: 166-179.

- [30] GUO, Xifeng, et al. Deep clustering with convolutional autoencoders. In: Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24. Springer International Publishing, 2017. p. 373-382.
- [31] WU, Edmond Q., et al. Pilots' fatigue status recognition using deep contractive autoencoder network. IEEE Transactions on Instrumentation and Measurement, 2019, 68.10: 3907-3919.
- [32] BARMAN, Dibyendu; HASNAT, Abul; NAG, Rupam. AN INTRODUCTION TO AUTOENCODERS.
- [33] Jonathan Shen et al., «Tacotron 2: Generating Human-like Speech from Text, » arXiv:1712.05884 [cs.CL], 16 Dec 2017, <https://doi.org/10.48550/arXiv.1712.05884> [accessed 2024/06/12]
- [34] Barnekow, & all. Creation and Detection of German Voice Deepfakes. 2021
- [35] «database LJ speech» <https://keithito.com/LJ-Speech-Dataset/> [accessed 2024/06/12]
- [36] Justin John & Mateo Cedillo «Tacotron2 Training Notebook» <https://github.com/justinjohn0306/FakeYou-Tacotron2-Notebook> [accessed 2024/06/12]
- [37] «Bäckström.et al. The cepstrum, mel-cepstrum and mel-frequency cepstral coefficients (MFCCs) » <https://speechprocessingbook.aalto.fi/Representations/Melcepstrum.html> [accessed 2024/06/12]
- [38] Halabi, Nawar «Arabic Speech Corpus», <https://en.arabicspeechcorpus.com/>[accessed 2024/06/12]
- [39] BOUZID, Mohamed Laid «Evaluation tests», <https://forms.gle/NEwBeeLjzDeCreQo8> [accessed 2024/06/12]