Learning Meters of Arabic Poems with Deep Learning

A Thesis Presented to the Faculty
of
Nile University

In Partial Fulfillment
of the Requirements for the Degree
of Master of SCIENCE

By
Moustafa Alaa Mohamed
July 2018

CERTIFICATION OF APPROVAL

Learning Meters of Arabic Poems with
Deep Learning

By
Moustafa Alaa Mohamed

Dr. Samhaa El-Beltagy (Chair) Professor of Electrical and Computer Engineering	Date
Dr. Waleed A. Yousef Associate Professor of Computer Science	Date
Dr. Mohsen Rashwan Professor of Electronics and Electrical Communications Engineering	Date

ACKNOWLEDGMENT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Thesis Outline

The coming chapters are arranged as follows:

- Chapter 1: Presents some basic introduction and background knowledge as regards the Arabic Poem and its definitions. Also, it contains details about the Arabic language and some feature used during our work.
- Chapter 2: Background related to Al-Arud, Deep Learning fundamentals and Literature Review for the previous work in this topic.
- Chapter 3: Dataset Design and Experiments. It Introduces the Dataset Design acquisition and encoding including the essential pre-processing steps, and the justification for their need. Pre-processing steps are data extraction, data cleansing, data format, data encoding techniques used. Also, it contains comparisons between the three techniques used. It presents the model's details and how we chose the model and the architecture and hyper-parameters details, Model Results, and discussion.

ABSTRACT

People can easily determine whether a piece of writing is a poem or prose, but only specialists can determine the class of poem.

In this thesis, We built a model that can classify poetry according to their meters; a forward step towards machine understanding of Arabic language.

A number of different deep learning models are proposed for poem meter classification. As poetry are sequence data, then recurrent neural networks are suitable for the task. We have trained three variants of them, LSTM, GRU with different architectures and hyper-parameters. Because meters are a sequence of characters, then we have encoded the input text at the character-level, so that we preserve the information provided by the letters succession directly fed to then models. Besides, We introduce a comparative study on the difference between binary and one-hot encoding regarding their effect on the learning curve. We also introduce a new encoding technique called *Two-Hot* which merges the advantages of both *Binary* and *One-Hot* techniques.

Artificial Intelligence currently works to do the human tasks such as our problem here. Our target in this thesis is to achieve the human accuracy which will make it easy for anyone to know the meter for any poem without referring to the language experts or to study the whole field to achieve it.

In this thesis, We will explain how to use the deep learning to classify the Arabic poem to classes. Also, explain in details the feature of Arabic poem and how to deal with this features. Besides, We explain how can anyone work with Arabic text encoding with a dynamic way to encode the text at the character level and deal with the Arabic text feature example the *Tashkeel*.

To the best of the authors' knowledge, this research is the first to address classifying poem meters in a machine learning approach, in general, and in RNN featureless based approach, in particular. In addition, the dataset is the first publicly available dataset ready for the purpose of future computational research.

Chapter 1

INTRODUCTION

In this chapter, We will give an introduction and basic background knowledge as regards the Arabic language. Also, it will contain details about Arabic Poetry history and the aim of this thesis.

Arabic is the fifth most widely spoken language [6]. It is written from right to left. Its alphabet consists of 28 primary letters, and there are 8 more derived letters from the basic ones, so the total count of Arabic characters is 36 characters. The writing system is cursive; hence, most letters join to the letter that comes after them, a few letters remain disjoint.

1.1 Arabic Poetry

Arabic poetry(الشعر العربى) is the earliest form of Arabic literature. It dates back to the Sixth century. Poets have written poems without knowing exactly what rules which make a collection of words a poem. People recognize poetry by nature, but only talented ones can write poems. This was the case until *Al-Farahidi* (718 – 786 CE) has analyzed the Arabic poetry, then he came up with that the succession of consonants and vowels produce patterns or *meters*, which make the music of poetry. He has counted them fifteen meters. After that, a student of *Al-Farahidi* has added one more meter to make them sixteen. Arabs call meters بحور which means "*seas*". The study of Arabic Poetry Meter Classification is named **Al-Arud (العَروض)**. It takes too much time for anyone to be an expert in this field.

1.2 Deep Learning

Deep Learning also named Deep Neural Network is part of Machine Learning algorithms. Deep Learning is trying to simulate the human brain into Neural dependency. Using Deep Learning, we can achieve better learning results from the data. Deep Neural Network needs a huge amount of data to achieve the expected learning curve and results. It also needs a massive amount of computation to build the networks which are based on an artificial neural network. We used the Recurrent Neural Network (RNN) to work on the Arabic Text which shown its ability to achieve outstanding performance over the text problem data. We also used LSTM to solve the long dependency issue in RNN. We will go deep into the Background section (add deep learning section reference).

1.3 Thesis Objectives

In this section, we need to explain the reason behind this project as a research idea. This project aimed to explore the Arabic language feature using well known studied rules which is Al-Arud. This exploration will help us understand how can machine learning model to understand the features of Arabic language without Hand-crafted the feature or static rule-based models. This will be a basic step to get deeper into the language and later we can have more robust machine learning models which can not only understand and classify the Poem/poetry but also, we can generate a similar text with the model without breaking the rules. All the previous are first steps toward a deeper understanding of the Arabic language features and rules using deep learning techniques. This research can open ideas about how the machine understanding the Arabic Grammar rules which will lead to new ideas to generate better Arabic text model.

In this study, we work on Poetry Meter Classification and utilize the latest technologies check the class of poem. We also worked to achieve near human expert results which make our work is a breakthrough in the field concerning the results compared to the current achieved results. Figure 1.1 shows the steps.,

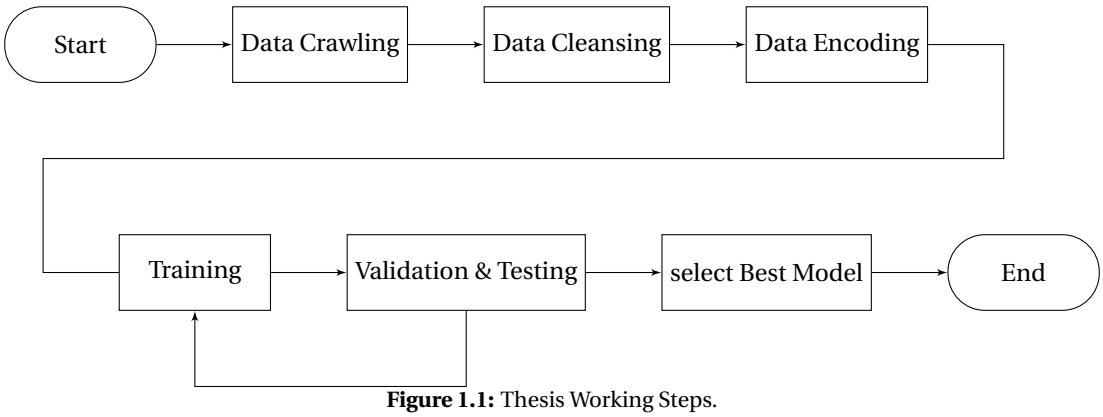


Figure 1.1: Thesis Working Steps.

- Crawling the data from the available sources with labeling.
- Clean and transform the data.
- Encode the data into a way to be input to the model to work on it. We used many encoding methods and compared each of them.
- Train the RNN model into the cleaned data.
- Validate and test the model.
- Enhance the model.

Chapter 2

BACKGROUND

In this chapter, We will give background knowledge as regards the Arabic Poetry and Al-Arud studies rules and details. Also, it will contain basic knowledge of Deep Learning concepts with Literature review for the previous work in this problem area.

Each Arabic letter represents a consonant, which means that short vowels are not represented by the 36 characters, for this reason, the need of *diacritics* arises. *Diacritics* are symbols that comes after a letter to state the short vowel accompanied by that letter. There are four diacritics ◌َ ◌ُ ◌ِ ◌ْ which represent the following short vowels /*a*/, /*u*/, /*i*/ and *no-vowel* respectively, their names are *fat-ha, dam-ma, kas-ra and sukun* respectively. The first three symbols are called *harakat*. Table 2.1 shows the 4 diacritics on a letter. There are two more sub-diacritics made up of the basic four to represent two cases: *Shadaa* and *Tanween*

Definition 1 ***Shadaa***
is To indicate the letter is doubled. Any letter with shaddah (ّ) the letter should be duplicated: first letter with a constant (sukoon) and second letter with a vowel (haraka) [7]; Table 2.1 shows the dal with shadda and the original letters.

Definition 2 ***Tanween***
is doubling the short vowel, and can convert Tanween fathah, Tanween dhammah or Tanween kasrah by replacing it with the appropriate vowel (◌ُ – dhammah, textarabic◌◌ – fathah or ◌ِ –kasrah) then add the Noon letter with constant to the end of the word [7]. Table 2.2 shows the difference between the original letter and the letter with Tanween

Diacritics	*without*	*fat-ha*	*kas-ra*	*dam-ma*	*sukun*	*letter with Shadda*	*letters without shadaa*
Shape	د	دَ	دِ	دُ	دْ	دَّ	دْدَ

Table 2.1: Diacritics on the letter د

Diacritics	*letter with tanween*	*letters without tanween*
Tanween Fat-ha	دً	دَ+نْ
Tanween Dam-ma	دٌ	دُ+نْ
Tanween Kas-ra	دٍ	دِ+نْ

Table 2.2: Tanween diacritics on the letter د

Arabs pronounce the sound /*n*/ accompanied *sukun* at the end the indefinite words, that sound corresponds to this letter نْ, it is called *noon-sakinah*, however, it is just a phone, it is not a part of the indefinite word, if a word comes as a definite word, no additional sound is added. Since it is not an essential sound, it is not written as a letter, but it is written as *tanween* ◌ً ◌ٌ ◌ٍ. *Tanween* states the sound *noon-sakinah*, but as you have noticed, there are 3 *tanween* symbols, this because *tanween* is added as a diacritic over the last letter of the indefinite word, one of the 3 harakat*harakat* accompanies the last letter, the last letter's *harakah* needs to be stated in addition to the sound *noon-sakinah*, so *tanween* is doubling the last letter's *haraka*, this way the last letter's *haraka* is preserved in addition to stating the sound *noon-sakinah*; for example, رَجُلْ + نْ is written رَجُلٌ and رَجُلِ + نْ is written رَجُلٍ.

Those two definition, Definition 1 and Definition 2 will help us to reduce the dimension of the letter's feature vector as we will see in *preparing data* section.Diacritics makes short vowels clearer, but they are not necessary. Moreover, a phrase without full diacritics or with just some on some letters is right linguistically, so it is allowed to drop them from the text. In Unicode, Arabic diacritics are standalone symbols, each of them has its own unicode. This is in contrast to the Latin diacritics; e.g., in the set *{ê, é, è, ë, ē, ĕ, ě}*, each combination of the letter *e* and a diacritic is represented by one unicode.

2.1 Arabic Arud

Definition 3 ***Arud***
In Arabic Arud natively has many meanings (the way, the direction, the light clouds and Mecca and Madinah [1] *[8]. Arud is the study of Arabic Poem meters and the rules which confirm if the Poem is sound meters & broken meters.*

The Author of Arabic Arud is *Al-Farahidi* (718 – 786 CE) who analyzed the Arabic poetry; then he came up with that the succession of consonants and vowels produce patterns or *meters*, which make the music of poetry. He was one of the famous people who know The melodies and the musical parts of speech. He ha counted them fifteen meters. After that, a student of *Al-Farahidi* has added one more meter to make them sixteen. Arabs call meters بحور which means "*seas*" Poets have written poems without knowing exactly what rules which make a collection of words a poem.

The Reasons which makes *Al-Farahidi* put this rules is

- Protect the Arabic Poetry from the broken meters.
- Distinguish between the original Arabic Poetry and the not original Arabic Poetry or from the prose.
- Make the rules clear and easy for anyone who needs to write a poem.

Some people said that the one-day Al-Farahidi was walking into the metal-market and he was said some of the poetry and for some reasons the knock of the metals matched the musical sound of the poetry he was saying then he got an idea to explore the Arud of the poetry.

There are many reason for Arud name

- It named Arud because some people said he put this rules in Arud place العَروض *with fat-ha, not with dam-ma such as the rules name* العُروض between Mecca and Al-Ta'if[8].
- Arud in Arabic is noun come from verb يعرض which means here to be assessed. They said because of Any poem should be assessed by Al-Arud rules so, it named Al-Arud [9].

2.1.1 Al-Farahidi and Pattern Recognition

This subsection is our opinion in Al-Farahidi and his method he followed during working on Arabic Poetry Meter Classification.

1. Al-Farahidi thought there is a pattern for every collection of the poetry by chance; however, He rigorously worked into this problem. He started analyzing the poetry and add every group with the same tafa'il to the same class.
2. He analyzed the outliers and the particular case from every class and added it to his model.
3. He revised the meters and get the cases and generalize his case to be fit into all poetry.
4. His student once he found some poetry which was not fit into any model to be a model for a new class.

The best essential point which made us admired by Al-Farahidi is his way of research and his passion for getting an indeed succession model. Also, his model is general and followed all the steps currently any Data scientist follows to explore new pattern. Some people state that He died when he was thinking about the problem he hit a wall which made trouble for him. His die story shows that he was thinking in profoundly about this problem. One of the most interest thing I found during this research is how he found this pattern and Al-Farahidi's way to find a new thing.

2.1.2 Feet Representation

A meter is an ordered sequence of feet. Feet are the basic units of meters; there are ten of them.

Definition 4 ***Feet***
A Foot consists of a sequence of ***Sukun*** *(Consonants) represented as (0) and* ***Harakah*** *(Vowels) (/). Traditionally, feet are represented by mnemonic words called tafa'il* تفاعيل.

[1] *Mecca and Madinah are two cities in Saudi Arabia.*

feet consists of three parts (Asbab (the word abstract means Reasons in English but it means in Arabic connection between two things.) أسباب, Awtaad (means Wedges in English) أوتاد, Fawasel (means Breaks in English) فواصل).

- **Asbab (أسباب)**: It has two types
  1. **Sabb Khafeef (سبب خفيف)** which happens when we have the first letter is harakah and the second is sukun (/0) example (هَبْ, لَمْ).
  2. **Sabb Thakeel (سبب ثقيل)** which happens when we have two harakah letter (//) example (لَكَ, بِكَ).
- **Awtad (أوتاد)**: It has two types
  1. **Watd Magmo'a (وتد مجموع)** which happens when we have two harakah letters followed by sukun (//0) example (مَشَى, عَلَى).
  2. **Watd Mafrouq (وتد مفروق)** which happens when we have two harakah and in between a sukun letter (/0/) example (مُنْذُ, مِصْرُ).
- **Fawasek (فواصل)**: It has two types
  1. **Faselah Soghra (فاصلة صغرى)** which happens when we have three harakah letters followed by a sukun letter (///0) example (ذَهَبُوا, سُفُناً).
  2. **Faselah Kobra (فاصلة كبرى)** which happens when we have four harakah letters followed by a sukun letter (////0) example (جَعَلَهُمْ).[2]

2.1.2.1 Rules for Arabic Letters Representation

Arabic Arud has one general rule in the poetry representation which is we represent only the letters which is (spoken) not the written which means the letters with phonatics not the written. We have give the below rules as a results of the general rule.

- Any letter with *harakah* represented as (/).
- Any letter with *sukun* represented as (0).
- Any letter with shaddah represented by two letters the first one will be *sukun* and the second letter will be *harakah* represented as (0/) example (مُحَمَّد) will be (//0//0).
- Any letter with tanween represented by two letters the first one is *haraka* (/) and the second is *sukun*.
- Alef without hamze (همزة الوصل) and Wow Algmaa are not represented example (وُاعلَموا) will be (/0//0)
- If we have a letter which is not written but (spoken) so, we will represent it example (هذا) it include Alef but not written (هاذا) the representation will be (/0/0).
- If we have *Meem Aljamaa* with harakah so, it represented with *Mad* example (هُمُ) will be (//0).
- *Alef Mad* (آ) will be two letters *Alef with harakah* and *Alef with sukun* example (آدَمُ) will be (/0//).
- if the verse ended with *harkah* we will add *sukun* to it.

Example: (note: the below representation first line is simliar the second one but with Arud language style).

أرَاكَ عَصِيَّ الدّمعِ شِيمَتُكَ الصّبرُ، أما للهوى نهيٌ عليكَ ولا أمرُ ؟
أراكَ عصيَ دمعِ شِيمتكَ صبرو، أما للهوى نهيَ عليكَ ولا أمرو ؟

#	Feet	Scansion	Construction
1	فَعُولُنْ	0/0//	*Watd Magmo'a* (فعو) and *Sabb Khafeef* (لن)
2	مَفَاعِيلُنْ	0/0/0//	*Watd Magmo'a* (مفا) and two *Sabb Khafeef*s (عي) (لن)
3	مُفَاعَلَتُنْ	0///0//	*Watd Magmo'a* (مفا), *Sabb Thakeel* (عل) and *Sabb Khafeef* (تن)
4	فَاعِلاَتُنْ	0/0//0/	*Sabb Khafeef* (فا), *Watd Magmo'a* (علا) and *Sabb Khafeef* (تن)
5	فَاعِ لاتُنْ	0/0//0/	*Watd Mafrouq* (فاع) and two *Sabb Khafeef* (لا)(تن) [3]
6	فَاعِلُنْ	0//0/	*Sabb Khafeef* (فا) and *Watd Magmo'a* (علن)
7	مُتَفَاعِلُنْ	0//0///	*Sabb Thakeel* (مت), *Sabb Khafeef* (فا) and *Watd Magmo'a* (علن)
8	مَفْعُولاَتُ	0//0///	two *Sabb Khafeef* (عو)(مف) and *Watd Mafrouq* (لات)
9	مُسْتَفْعِلُنْ	0//0/0/	two *Sabb Khafeef* (تف)(مس) and *Watd Magmo'a* (علن)
10	مُسْتَفْعِ لُنْ	0//0/0/	*Sabb Khafeef* (مس), *Watd Mafrouq* (تفع) and *Sabb Khafeef* (لن)[4]

Table 2.3: The ten feet of the Arabic meters.

2.1.3 Arabic Poetry feet

Arabic poetry feet has ten tafa'il تفاعيل (scansion) any peom constructed from these feet. They are eight from writing (syntax) perspective, But it ten in the rules.

Definition 5 ***Meter***
Poetic meters define the basic rhythm of the poem. Each meter is described by a set of ordered feet which can be represented as ordered sets of consonants and vowels [10]. A meter in Arabic named Bahr (بحر)

ولد الهدى فالكائنات ضياء وفم الزمان تبسم وثناء انشاء
الروح والملأ الملائك حوله للدين والدنيا به بشراء

Definition 6 ***Arabic Verse***
refers to "poetry" as contrasted to prose. Where the common unit of a verse is based on meter or rhyme, the common unit of prose is purely grammatical, such as a sentence or paragraph [11]. A verse in Arabic named Bayt(بيت)

Definition 7 ***Shatr***
A verse consists of two halves, each of them is called shatr and carries the full meter. We will use the term shatr to refer to a verse's half; whether the right or the left half.

Definition 8 ***Poem***
is a set of verses has the same meter and rhyme.

2.1.4 Arabic Poetry Meters

2.1.4.1 Al-Taweel الطويل

tafa'il

فعولن مفاعيلن فعولن مفاعلن فعولن مفاعيلن فعولن مفاعيلن

Example:

أَمَاوِيَّ إِنَّ المَالَ غَادٍ وَرَائِحٌ وَيَبْقَى مِنَ المَالِ الأَحَادِيْثُ وَالذِّكْرُ
أماوي ي إنن لما ل غادن ورائحن ويبقى من لمال لْ أحاديـ ـث وذكرو
//0/0/0 //0/0/0 //0/0/0 //0/0/0
فعولن مفاعيلن فعولن مفاعلن فعولن مفاعيلن فعولن مفاعيلن

[2] *Some of Arab linguistic scientist assume the Faselah Soghra as a combination between Sabb Thakeel and Sabb Khafeef. Same for the Faselah Kobra to be a combination between Sabb Thakeel and Watd Magmo'a. So, they didn't assume we have three types of feet it is only pure two and any other feet constructed from this two. In this thesis we assume there are three feet .*

[3] *We separated the letters (ع) and (لا) in (فاع لاتن) to show that this part is Watd Mafrouq and distinguish between this feet and (فاع لاتن) which contains Watd Magmo'a .*

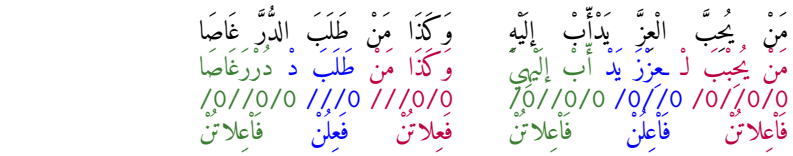
[4] *We separated the letters (ع) and (ل) in (مستفع لن) to show that it ends with a Watd Mafrouq and distinguish between this feet and (مستفعلن) which contains Watd Magmo'a*

2.1.4.2 Al-Madeed المديد

tafa'il

فاعلاتن فاعلن فاعلاتن فاعلاتن فاعلن فاعلاتن

Example:



2.1.4.3 Al-Baseet البسيط

tafa'il

مستفعلن فعلن مستفعلن فاعلن مستفعلن فاعلن مستفعلن فاعلن

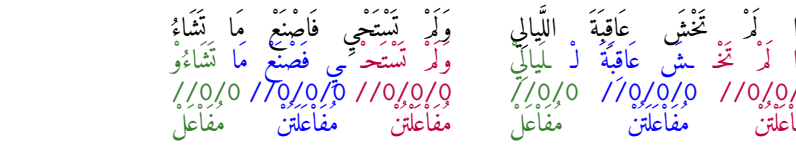
Example:

2.1.4.4 Al-Wafer الوافر

tafa'il

مفاعلتن مفاعلتن مفاعلتن مفاعلتن مفاعلتن مفاعلتن

Example:

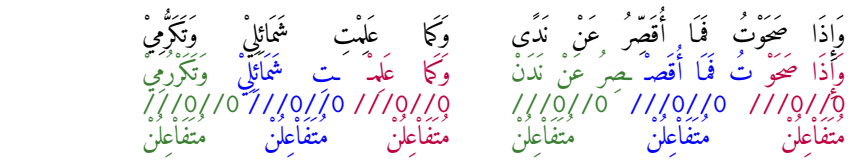


2.1.4.5 Al-Kamel الكامل

tafa'il

متفاعلن متفاعلن متفاعلن متفاعلن متفاعلن متفاعلن

Example:



2.1.4.6 Al-Hazaj الهزج

tafa'il

مفاعيلن مفاعيلن مفاعيلن مفاعيلن

Example:

فَهُبُّوْا يَا بَنِيْ أُمِّيْ إِلَى الْعَلْيَاءِ بِالْعِلْمِ
فَهُبُّوْ يَا بَنِيْ أُمِّيْ // 0/0/0/0 //0/0/0/0
مَفَاعِيْلُنْ مَفَاعِيْلُنْ مَفَاعِيْلُنْ مَفَاعِيْلُنْ

2.1.4.7 Al-Rejz الرجز

tafa'il

مستفعلن مستفعلن مستفعلن مستفعلن مستفعلن مستفعل

Example:

لاخَيْرَ فِيْمَنْ كَفَّ عَنَّا شَرَّهُ إِنْ كَانَ لِايُرْجَى لِيَوْمٍ خَيْرُهُ
لاخير فِيْ من كَفْف عَنْ نا شَرْرهو إِنْ كَانَ لا يرجى ليو مِنْ خيرهو
/0/0//0 /0/0//0 /0/0//0 /0/0//0 /0/0//0 /0/0//0
مستْفعِلْن مستْفعِلْن مستْفعِلْن مستْفعِلْن مستْفعِلْن مستْفعِلْن

2.1.4.8 Al-Raml الرمل

tafa'il

فاعلاتن فاعلاتن فاعلاتن فاعلاتن فاعلاتن فاعلاتن

Example:

قَادَنِيْ طَرْفِيْ وَقَلْبِيْ لِلْهَوَى كَيْفَ مِنْ قَلْبِيْ وَمِنْ طَرْفِيْ حَذَارِيْ
قَادَنِي طَرْ فِيْ وَقَلْبِي لَلْهوى كَيف مِن قَلْ بِي ومِن طَرْ فِي حذَارِي
/0//0/0 /0//0/0 /0//0 /0//0/0 /0//0/0 /0//0/0
فَاعِلاتُنْ فَاعِلاتُنْ فَاعِلا فَاعِلاتُنْ فَاعِلاتُنْ فَاعِلاتُنْ

2.1.4.9 Al-Sarea السريع

tafa'il

مستفعلن مستفعلن مفعولات مستفعلن مستفعلن مفعولات

Example:

وَمَنْ دَعَا النَّاسَ إِلَى ذَمِّهِ ذُمُّوْهُ بِالْحَقِّ وبِالْبَاطِلِ
ومَنْ دعَا نَ ناسَ إِلَى ذَمِّهِى ذمُّوه بِلْ حقِّ وبِلْ بَاطِلِ
/0//0 /0////0 /0//0 /0//0/0 /0////0 /0//0
مُتَفْعِلْنْ مُسْتَعِلْنْ مَفْعَلا مُسْتَفْعِلْنْ مُسْتَعِلْنْ مَفْعَلا

2.1.4.10 Al-Monsareh المنسرح

tafa'il

مستفعلن مفعولات مستفعلن مستفعلن مفعولات مستفعلن

Example:

إِنَّ ابْنَ زَيْدٍ لاَزَالَ مُسْتَعْمِلا لِلْخَيْرِ يُفْشِيْ فِيْ مِصْرِهِ الْعُرُفَا
إِنْ بْنَ زَيْ دِنْ لاَزَالَ مُستعمِلَنْ لِلْخَيْ رِيُفْ شِي فِيْ مِصْرِ هِ لْعُرفَا
/0//0/0 /0/0/0 /0//0 /0//0/0 /0/0/0 /0//0
مُسْتَفْعِلْنْ مَفْعُولاتُ مُسْتَفْعِلْنْ مُسْتَفْعِلْنْ مَفْعُولاتُ مُسْتَفْعِلْنْ

2.1.4.11 Al-Khafeef الخفيف

tafa'il

فاعلاتن مستفع لن فاعلاتن فاعلاتن مستفع لن فاعلاتن

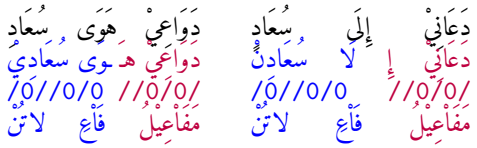
Example:

2.1.4.12 Al-Modarea المضارع

tafa'il

مفاعلين فاع لاتن مفاعلين فاع لاتن

Example:

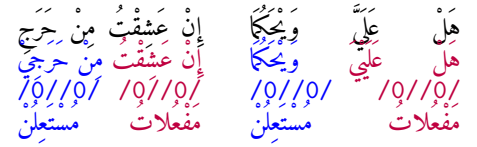


2.1.4.13 Al-Moktadeb المقتضب

tafa'il

مفعولات مستفعلن مفعولات مستفعلن

Example:



2.1.4.14 Al-Mojtaz المجتث

tafa'il

مستفع لن فاعلاتن مستفع لن فاعلاتن

Example:

2.1.4.15 Al-Motaqareb المتقارب

tafa'il

فعولن فعولن فعولن فعولن فعولن فعولن فعولن فعولن

Example:

2.1.4.16 Al-Motadarek المتدارك

tafa'il

فَاْعِلُنْ فَاْعِلُنْ فَاْعِلُنْ فَاْعِلُنْ فَاْعِلُنْ فَاْعِلُنْ فَاْعِلُنْ فَاْعِلُنْ

Example:

2.1.5 Meters Relation

Al-Arud classes have a relation between each other. The author of this relations is Al-Farahidi. He designed this relation to show the similarity and the difference between the meters. In this section, we will explain in brief this relation between them, and we will demonstrate its effect in Section 3.4. There are five groups of relationships between the meters. The details about how this relations arises will be omitted as it need more explanation in Arud details. But we will show brief about each one.

- **Al-Mokhtalef المُخْتَلِف**: This group also know as Al-Taweel group. As all the meters inside it is sub-child from Al-Taweel from its tafa'il. It contains three meters Al-Taweel, Al-Madeed and Al-Baseet. Table 2.4 show this group relations.
- **Al-Mo'talef المُؤْتَلِف**: This group contains two classes Al-Kamel and Al-Wafer. Table 2.4 shows this relation between them.
- **Al-Mojtaleb المُجْتَلَب**: This group contains three classes Al-Raml, Al-Rejz and Al-Hazaj. Table 2.4 shows this relation between them.
- **Al-Moshtabeh المُشْتَبِه**: This group contains six classes Al-Sarea, Al-Monsareh, Al-Khafeef, Al-Modarea, Al-Moktadeb and Al-Mojtaz. Table 2.4 shows this relation between them.
- **Al-Motafeq المُتَّفِق**: This group contains two classes Al-Motaqareb and Al-Motadarek. Table 2.4 shows this relation between them.

Group Name	Meter	Feet
Al-Mokhtalef	Al-Taweel	فَعُولُنْ مَفَاعِيلُنْ فَعُولُنْ مَفَاعِيلُنْ
	Al-Madeed	فَاْعِلاتُنْ فَاْعِلُنْ فَاْعِلاتُنْ فَاْعِلُنْ
	Al-Baseet	مُسْتَفْعِلُنْ فَاْعِلُنْ مُسْتَفْعِلُنْ فَاْعِلُنْ
Al-Mo'talef	Al-Wafer	مُفَاعَلَتُنْ مُفَاعَلَتُنْ مُفَاعَلَتُنْ
	Al-Kamel	مُتَفَاعِلُنْ مُتَفَاعِلُنْ مُتَفَاعِلُنْ
Al-Mojtaleb	Al-Raml	فَاعِلاتُنْ فَاعِلاتُنْ فَاعِلاتُنْ
	Al-Rejz	مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ
	Al-Hazaj	مَفَاعِيلُنْ مَفَاعِيلُنْ مَفَاعِيلُنْ
Al-Moshtabeh	Al-Sarea	مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ مَفْعُولاتُ
	Al-Monsareh	مُسْتَفْعِلُنْ مَفْعُولاتُ مُسْتَفْعِلُنْ
	Al-Khafeef	فَاعِلاتُنْ مُسْتَفْعِلُنْ فَاعِلاتُنْ
	Al-Modarea	مَفَاعِيلُنْ فَاعِلاتُنْ مَفَاعِيلُنْ
	Al-Moktadeb	مَفْعُولاتُ مُسْتَفْعِلُنْ مُسْتَفْعِلُنْ
	Al-Mojtaz	مُسْتَفْعِ لُنْ فَاعِلاتُنْ فَاعِلاتُنْ
Al-Motafeq	Al-Motaqareb	فَعُولُنْ فَعُولُنْ فَعُولُنْ فَعُولُنْ
	Al-Motadarek	فَاعِلُنْ فَاعِلُنْ فَاعِلُنْ فَاعِلُنْ

Table 2.4: Meters Group.

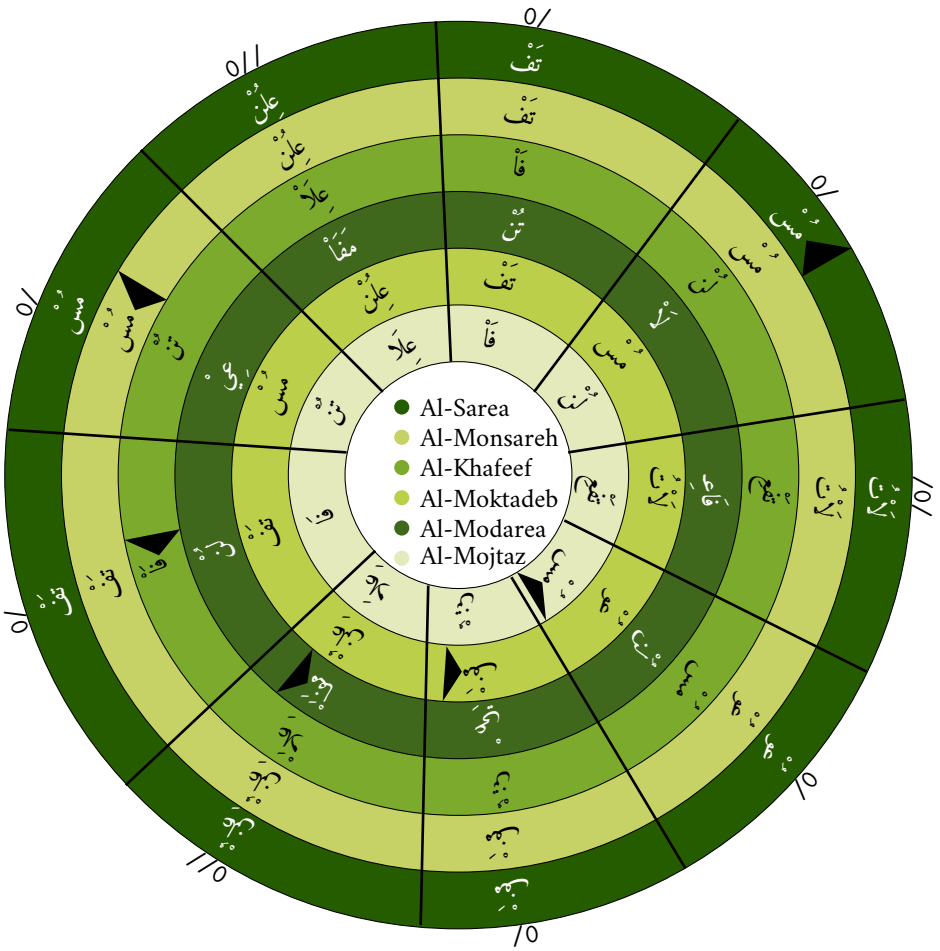
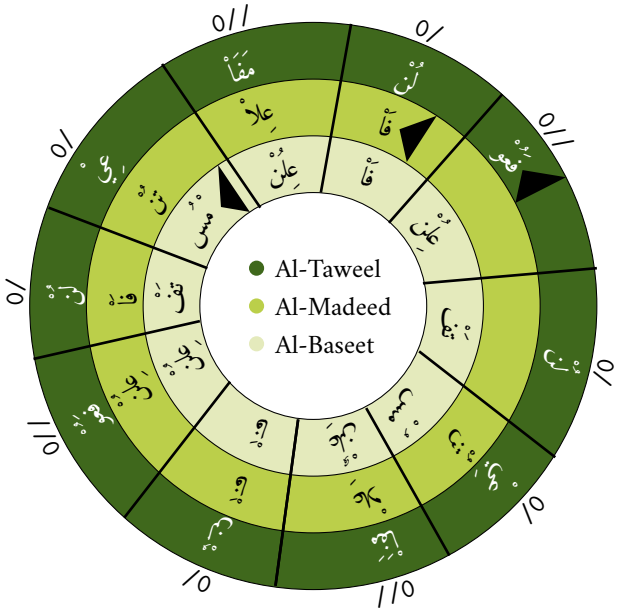
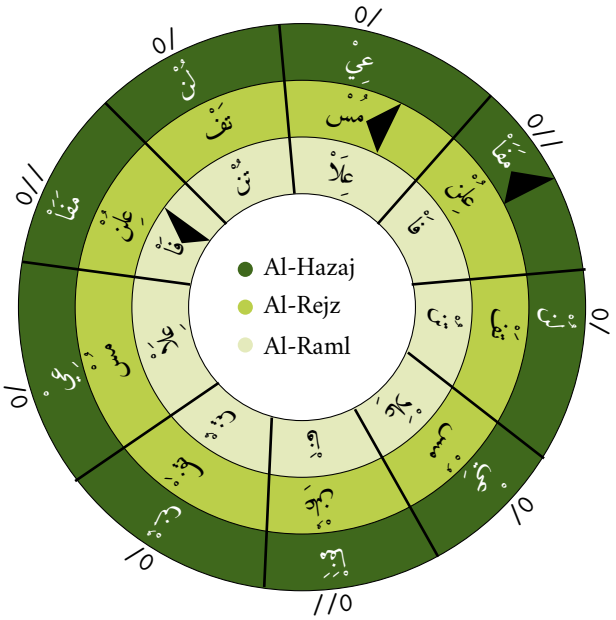


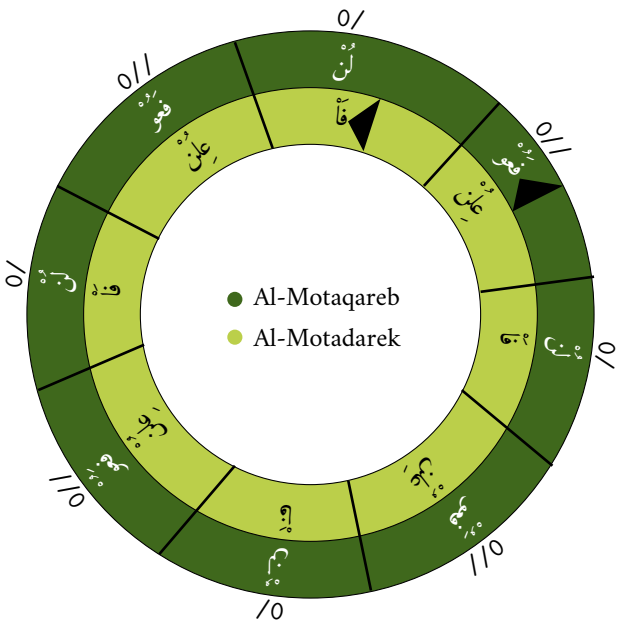
Figure 2.1: Al-Moshtabeh Meter Group.



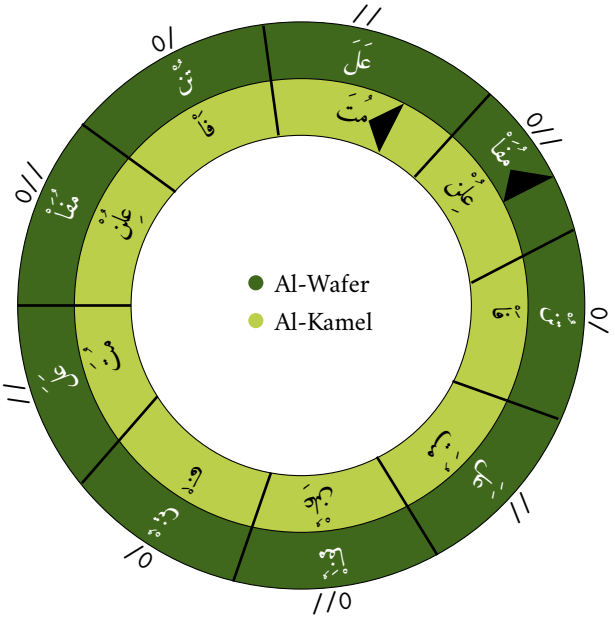
(a) Al-Mokhtalef Meters Group.



(b) Al-Mojtaleb Meters Group.



(c) Al-Motafeq Meters Group.



(d) Al-Mo'talef Meters Group.

2.2 Deep Learning Background

What is Deep Learning? *Deep Learning is a new approach of Machine Learning research which focus on learning and understanding from the data without the needs for the human operator to formally specify all the knowledge that the computer needs. This method built using a hierarchy of concept which enables the computer to learn complex concepts by building them layer by layer from simpler ones. If there is a graph which shows how this concept built we will figure out a very deep graph with many layers, for this reason, we call this approach to AI deep learning [1]*

There was many of early trials to utilize the AI into real life problems. For Example, IBM's Deep Blue chess-playing system which defeated world champion Garry Kasprov in 1997 (Hsu , 2002).

Another approach which used to use AI but using hard-code knowledge about the world informal language. A computer can understand statements from the formal language automatically using logical inference rules. This is known as the knowledge base approach to artificial intelligence rules. None of these projects has achieved significant success. For Example, Cyc is tried to gather a comprehensive ontology and knowledge base about the basic concepts about how the world works Cyc (Lenat and Guha, 1989). Cyc is an inference engine and a database of statements in a language called Cycl. A staff of human supervisors enters these statements. People struggle to devise formal rules with enough complexity to describe the world accurately[1].

The difficulty faced in the previous system is due to the hard-coded knowledge has shown up the AI need to acquire their knowledge from the data itself. This capability is known as machine learning. This approach has introduced some algorithms which solve and tackle the problems from which we can, for example, check the email is spam or not. Also, it used for other problems for price predictions for housing Example of this algorithms is (Naive Bayes, Logistic regression).

This simple machine learning approach is working in the data but not with its original format it required some different representation to be input for the model. This different representation named feature engineering. Feature Engineering example: in case of email spam or not spam example it can be word frequency, char frequency, class attributes, capital letters frequency, some other data processing such as remove stop words from the input lemmatization. So, all the previous feature provided by a human expert which know the problem in details and analyzing which features it affect the data then add it as a feature to the input model.

However, for many tasks, it is difficult to identify the features which should be extracted. For example, we need to detect cars in photographs. We know every car have wheels. So, to detect cars, we can check if there is a wheel to be a feature for car detection. However, to detect or to describe wheels in terms of pixel values is a difficult task. The image may be not clear or may be complicated by shadows, the sun glaring off the metal parts of the wheel, the blurring in images may not make it clear sometimes, and so on[1].

One solution to solve this problem is to use machine learning itself to discover not only the output of the model but also the features which are the input for the model. This approach is known as representation learning. Learned representation can achieve better results than hard-designed representation. This approach also allows AI systems to rapidly adapt to new tasks or be automatically identify it from any new data. A representation learning can discover many features automatically fast or can take more times in case complex tasks, but at least it will get an excellent set of features which adapt for any complex problem without the need for manual features. In this research, we used the AI to identify the features for our model which make this model get a breakthrough results than the old fashion of manual feature machine learning used.

If we go back to the image example, we can show that it is not an easy task to extract features to detect the car from an image. So, Deep learning is trying to solve this problem in feature engineering by introducing representation learning that are build complex representations in terms of another simpler layer of representations Figure 2.2 shows how deep learning represents an image of a person by combining simpler representation example the edges and contours which led to understanding complex representations. The benefit from allowing the computer to understand the data and building the representation is the ability now for building and understanding very complex representation and also, to utilize and combine features from simpler to deep representations with many ways such as recurrent or sequences.

Modern deep learning provides a compelling framework for learning data problems. This model becomes more complex by the adding more layers and more units within a layer. Deep Learning model is working perfectly on the big dataset which allows the model to learn the data features in a good way.

In the remaining parts in this section we will start introducing the main concepts and component used in deep learning, Also the basic unit into Recurrent Neural networks and LSTM.

Figure 2.2: Illustrations on how can Deep Learning work based on images figure presented from [1] [2].

2.2.1 Logistic Regression

Logistic Regression is a machine learning algorithm which we can assume has the basic idea behind the deep learning we will explain it later. Also, Logistic Regression is one of the most used machine learning techniques for binary classification.

A simple example of logistic regression it would be if we have an algorithm for fraud detection. It takes some raw data input and detect if it is a fraud case or not let's assume fraud case is one and a non-fraud case is zero. David Cox developed logistic regression in 1958 [12]. The logistic name came from its core function logistic function which also named as *Sigmoid function* function in Equation (2.1). The Logistic function is shaped as S-shape.

Also, one of these function features it can take any input real number and convert it into a value between 1 and 0.

Let's take an Example, Given x, we want to get the predictions of $\hat{y}$ which is the estimate of y when $\hat{y}$ is presented in Equation (2.2). So,to calculate the output function for logistic regression using Equation (2.3). Note: if we remove the Sigmoid function σ from the equation it will be Linear Regression model and $\hat{y}$ can be greater than 1 or negative. Figure 2.3 show the Sigmoid function output.

$$x = \frac{1}{1-e^{-x}} \quad \text{where} \quad x \in \mathbb{R}^{n_x} \tag{2.1}$$

$$\hat{y} = P(y=1|x) \quad \text{where} \quad 0 \leq \hat{y} \leq 1 \tag{2.2}$$

$$\hat{y} = \sigma(w^t x + b) \quad \text{where:} \quad \sigma(z) = \frac{1}{1-e^{-z}},\ w \in \mathbb{R}^{n_x},\ b \in \mathbb{R} \tag{2.3}$$

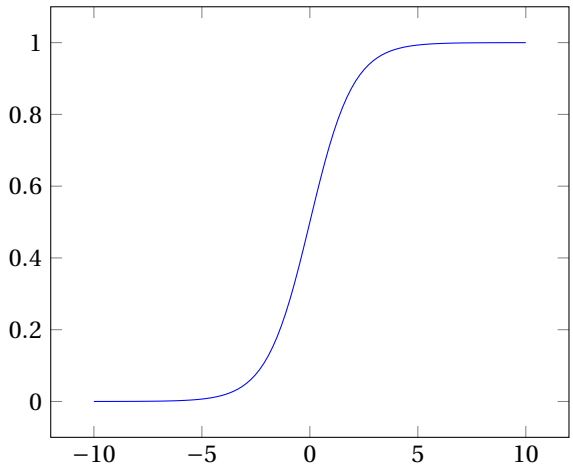


Figure 2.3: Logistic Regression Function (S-Shape)

2.2.1.1 Loss Error Function

Loss Error Function is the function which describes how well our algorithm can understand $\hat{y}$ y b when the true label is y. It also can be defined as the difference between the true value of y and the estimated value of $\hat{y}$. [5]. Equation (2.4) describe the loss function for Logistic Regression. There are another functions can represent the loss functions but we take the below as example. As we know y is the label which should be 1 or 0. So, The reason why this function make sense to describe the loss function as below

- in case (y = 1) Equation (2.5) we need $\hat{y}$ to be big as possible to be equal or near y true which is 1. So, $-(\log \hat{y})$ will get the value. Note as explained before Sigmoid function can't be greater than 1 or less than 0.
- in case (y = 0) Equation (2.6) we need $\hat{y}$ to be small as possible to be equal or near y true which is 0. So, $-\log(1-\hat{y})$ will get the value.

$$\ell(y,\hat{y}) = -(y\log\hat{y} + (1-y)\log(1-\hat{y})) \tag{2.4}$$

$$\begin{aligned} \text{(if y = 1)} \quad \ell(y,\hat{y}) &= -(y\log\hat{y} + (1-y)\log(1-\hat{y})) \\ &= -(1\log\hat{y} + (1-1)\log(1-\hat{y})) \\ &= -(\log\hat{y}) \end{aligned} \tag{2.5}$$

$$\begin{aligned} \text{(if y = 0)} \quad \ell(y,\hat{y}) &= -(y\log\hat{y} + (1-y)\log(1-\hat{y})) \\ &= -(0\times\log\hat{y} + (1-0)\log(1-\hat{y})) \\ &= -\log(1-\hat{y}) \end{aligned} \tag{2.6}$$

2.2.1.2 Cost Function

To predict y from $\hat{y}$ we learn from the input parameters in this case it will be ***(w,b)*** from Equation (2.3) as ***(w,b)*** is the parameters which define the relation between input dataset X and the output Y. So, Cost Function will measure how well you are doing an entire training set and the ability to understand the relation between X,Y.

Cost function J in Equation (2.7) is the average of loss function applied to every training example which equal the sum of the lost for each training example divided on the total number of training example.

$$\begin{aligned} J(w,b) &= \frac{\sum_{i=1}^{m}\ell(y^i,\widehat{y^i})}{m} \quad \text{where m is the total number of training example} \\ &= \frac{-\sum_{i=1}^{m}[(y^i\log\widehat{y^i} + (1-y^i)\log(1-\widehat{y^i}))]}{m} \end{aligned} \tag{2.7}$$

[5] *Parts of this subsections are explained into Andrew NG Coursera courses in deep learning and It written using our understanding to this topic but the equations and the idea taken from the course*

2.2.1.3 Convex Function vs Non-Convex Function

In this subsection we will give an overview about the convex and non-convex functions and its relation with deep learning. We will not explain the proofs or write it. We will explain in general about the definition and its related features to our topic.

Convex Function : In mathematics, a real-valued function defined on an n-dimensional interval is called *convex* if the line segment between any two points on the graph of the function lies above or on the graph, in a Euclidean space (or more generally a vector space) of at least two dimensions [13]. More generally, a function $f(x)$ Figure 2.4(a) is *convex* on an interval $[a,b]$ if for any two points x_1 and x_2 in $[a,b]$ and any λ where $0 < \lambda < 1$ [14],

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \tag{2.8}$$

For a twice differentiable function of a single variable, if the second derivative is always greater than or equal to zero for its entire domain then the function is *convex*. Well-known examples of convex functions include the quadratic function X^2 Figure 2.6(a). So, If $f(x)$ has a second derivative in $[a,b]$, then a necessary and sufficient condition for it to be *convex* on that interval is that the second derivative $f''(x) \geq 0$ for all x in $[a,b]$. [15].

If the inequality above is strict for all x_1 and x_2, then $f(x)$ is called **strictly convex**. *Convex* function on an open set has no more than one minimum. So, in strictly convex the local minimum = global minimum. This feature is very important feature for any optimization problem. As we will see most of deep learning problems are related to how to optimize the function to find the minimum point in this function. It will be nice and easy problem to face a convex function but in real world most of the cases is non convex functions.

Non-Convex Function : In mathematics, a Non-Convex (also named concave) function is the negative of a *convex function*. A function $f(x)$ is said to be concave on an interval $[a,b]$ if, for any points x_1 and x_2 in $[a,b]$, the function $-f(x)$ is convex on that interval.

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2) \tag{2.9}$$

The function is also called strictly concave if,

$$f(\lambda x_1 + (1-\lambda)x_2) > \lambda f(x_1) + (1-\lambda) f(x_2) \tag{2.10}$$

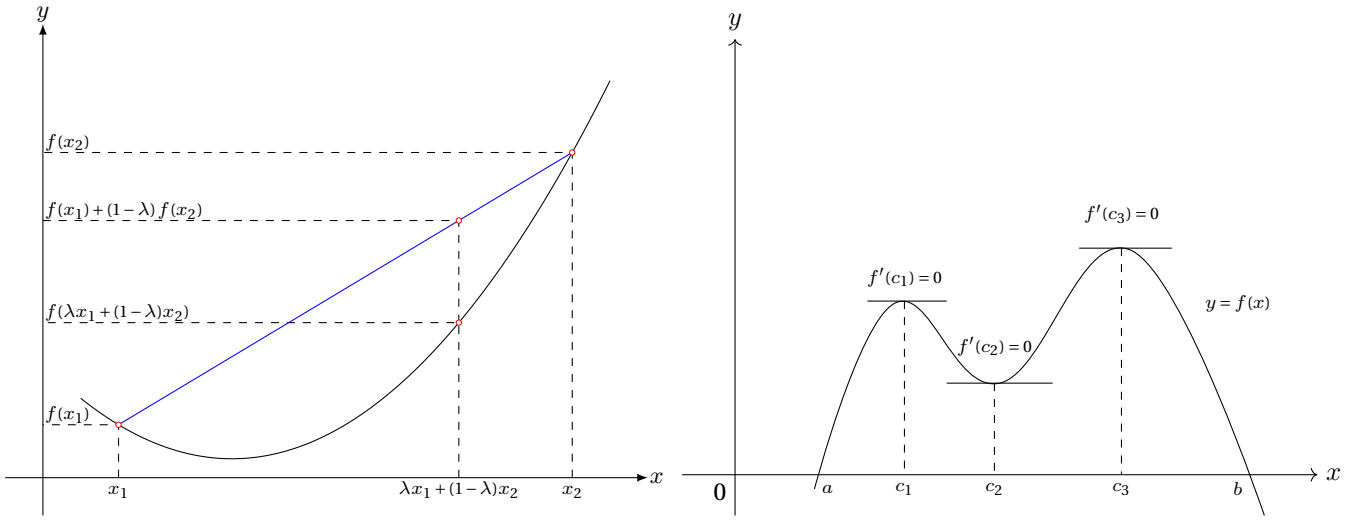
If f is twice-differentiable, then f is concave if and only if f'' is non-positive (or, informally, if the "acceleration" is non-positive). If its second derivative is negative then it is strictly concave, but the opposite is not true, as shown by $f(x) = x^4$ [16]. Also, a differentiable function f is (strictly) concave on an interval if and only if its derivative function f' is (strictly) monotonically decreasing on that interval, that is, a concave function has a non-increasing (decreasing) slope. The problem for non-convex optimization is to find the local minimum. Sometimes you will find many local minimum and it will be hard to optimize this function Figure 2.4(b).

2.2.1.4 Gradient Descent

As we explained in the previous parts, we need to find the relation between X,Y from the input parameters ***(w,b)*** which will make the cost function in Equation (2.7) to the minimum. In other words we need to find the best value of ***J(w,b)*** which will represent the relation and reduce the error between y and $\widehat{y}$ So, we need to minimize ***J(w,b)***. To illustrate the relation between ***J(w,b)*** we will assume for simplicity the relation will be function of one variable ***J(w)***. As shown in Figure 2.5 we have a curve which represent the function ***J(w)*** we need to find the minimum point in this curve which is the local minimum (red point in the previous figure) assuming it is a ***convex function***. We will use in Equation (2.11) to find the local minimum.

To explain how this equation works let's take simple function $f(x) = x^2$ then select a random point P_1 from Figure 2.6(a) then pick another point P_2 let's take derivative *(which by definition is the slope of the function at the point which also the change between these two points)* The slope of this function is the height (3) divided by the width (1) this is the tangent of $J(w) = \frac{3}{1}$ at this point. If the derivative is positive so, w will be update minus the derivative multiplied by learning rate alpha α as in Equation (2.11). We will repeat the previous step until value of w get the lowest minimum. When w get the lowest minimum the derivative will be negative so, w will start to increase again at this step the algorithm will stop. Also, we can demonstrate the effect of different α values and its impact on the function we can show this effect in Figure 2.6(b) but the main point it is not always a happy scenario sometimes the high $alpha$ is not a good idea, and it depends on the problem and the dataset.

Now, Let's generalize the above equation assume we have two parameters ***(w,b)*** and we need to calculate the cost function for ***J(w,b)*** we will work on as two steps first function in Equation (2.12a) wrt ***(w)*** and second function in Equation (2.12b) wrt ***(b)***



(a) Convex Function Example

(b) Concave Function Example

Figure 2.4: Convex and Concave functions examples.

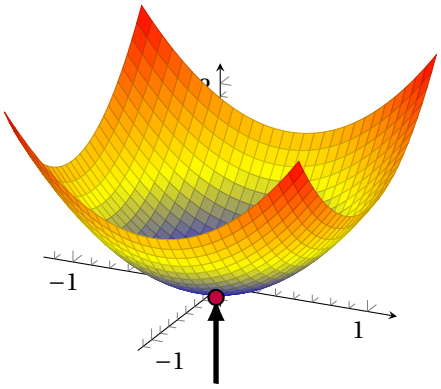
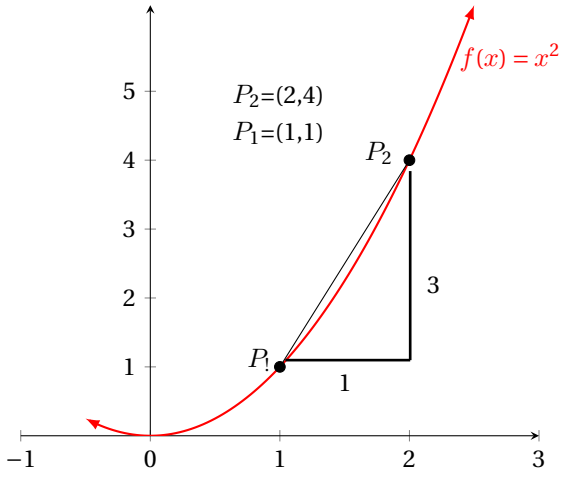
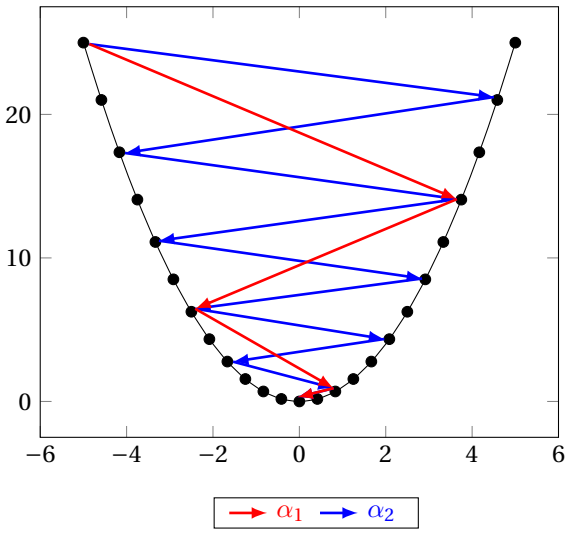


Figure 2.5: Gradient Decent



(a) Derivative Example of function $f(x) = x^2$

Derivative Example with different α values



(b) Derivative Example of function $f(x) = x^2$ where $\alpha_1 > \alpha_2$

$$
\begin{aligned}
w &:= w - \alpha dw \quad \textit{alpha is learning rate} \\
&:= w - \alpha \frac{dJ(w)}{dw} \quad \textit{d represent the derivative wrt w}
\end{aligned}
\tag{2.11}
$$

$$w := w - \alpha \frac{dJ(w,b)}{dw} \tag{2.12a}$$

$$b := b - \alpha \frac{dJ(w,b)}{db} \tag{2.12b}$$

2.2.1.5 Logistic Regression derivatives

As described we need to calculate the gradient descent to get the best $\widehat{y}$ which minimizes the total cost in equation (2.13). So, we will do backpropagation to get the value of dz we need to calculate da in Equation (2.14) then we will calculate dz based on the output of da from Equation (2.15). After that, We will start to take the derivative for z function parameters w_1, w_2, b. Once we got the values of dw_1, dw_2, db we can use it to calculate the estimated values of w_1, w_2, b in the Equations (2.16), (2.17), (2.18)

$$\boxed{\widehat{y} = \sigma(z) = a} \longrightarrow \boxed{z = w^t x + b = w_1 x_1 + w_2 + x_2 + b} \longrightarrow \boxed{\ell(a,y)} \tag{2.13}$$

$$\boxed{da = \frac{d\ell}{da} = \frac{d\ell(a,y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}} \tag{2.14}$$

$$\boxed{dz = \frac{d\ell}{dz} = \frac{d\ell(a,y)}{dz} = \frac{d\ell}{da}.\frac{da}{dz}} = \boxed{(-\frac{y}{a} + \frac{1-y}{1-a}).a(a-1)} = \boxed{a-y} \tag{2.15}$$

$$\boxed{dw_1 = \frac{\partial \ell}{dw_1} = x_1 dz} \longrightarrow \boxed{w_1 := w_1 - \alpha dw_1} \tag{2.16}$$

$$\boxed{dw_2 = \frac{\partial \ell}{dw_2} = x_2 dz} \longrightarrow \boxed{w_2 := w_2 - \alpha dw_2} \tag{2.17}$$

$$\boxed{db = \frac{\partial \ell}{db} = dz} \longrightarrow \boxed{b := b - \alpha db} \tag{2.18}$$

2.2.1.6 Implementing Logistic Regression on m example

To implement a simple 1 iteration example below sample code simulate the program structure. First, assume $J = 0, dw_1 = 0, dw_2 = 0, db = 0$. Then calculate the feedforward step. Then backpropagation calculate. Finally, update the parameters. We can transfer the above equation into the below python sample code.

```
import numpy as np
J = 0, dw_1 = 0, dw_2 = 0,db = 0, alpha = .02
# FEED FORWARD PROPAGATION
A = 1 / (1 + np.exp(-(np.dot(w.T,X) + b))#    Z = np.dot(w.T,X) + b
cost = (- 1 / m) * np.sum(Y * np.log(A) + (1 - Y) * (np.log(1 - A)))
# BACKWARD PROPAGATION (TO FIND GRADIENT)
dw = (1 / m) * np.dot(X, (A - Y).T) #     dz = A - Y
db = (1 / m) * np.sum(A - Y)
# UPDATE THE PARAMETERS
w = w - alpha * dw
b = b - alpha * db

```

2.2.2 The Neuron

As we all know, Most computer research is trying to simulate the human brain as it is the most advanced smartest creation. If we are trying to check how the model understands the new information regarding for example bananas photo we can give a baby two bananas then ask him about it baby can remember it with all it new shapes. Same case if you inform any human about some information and trying to get a new inference it will automatically detect this information. So, The new research trying to simulate the human brain model into an Artificial Intelligence model to trying to get this performance. In this subsection, we will try to give an overview of the relation between the new research era and the human brain.

The neuron is the foundation unit of the brain. The size of the brain is as about the size of a grain of rice. The brain contains more over 10000 neurons with average 6000 connections with other neurons [17]. These massive networks allow our brain to build its

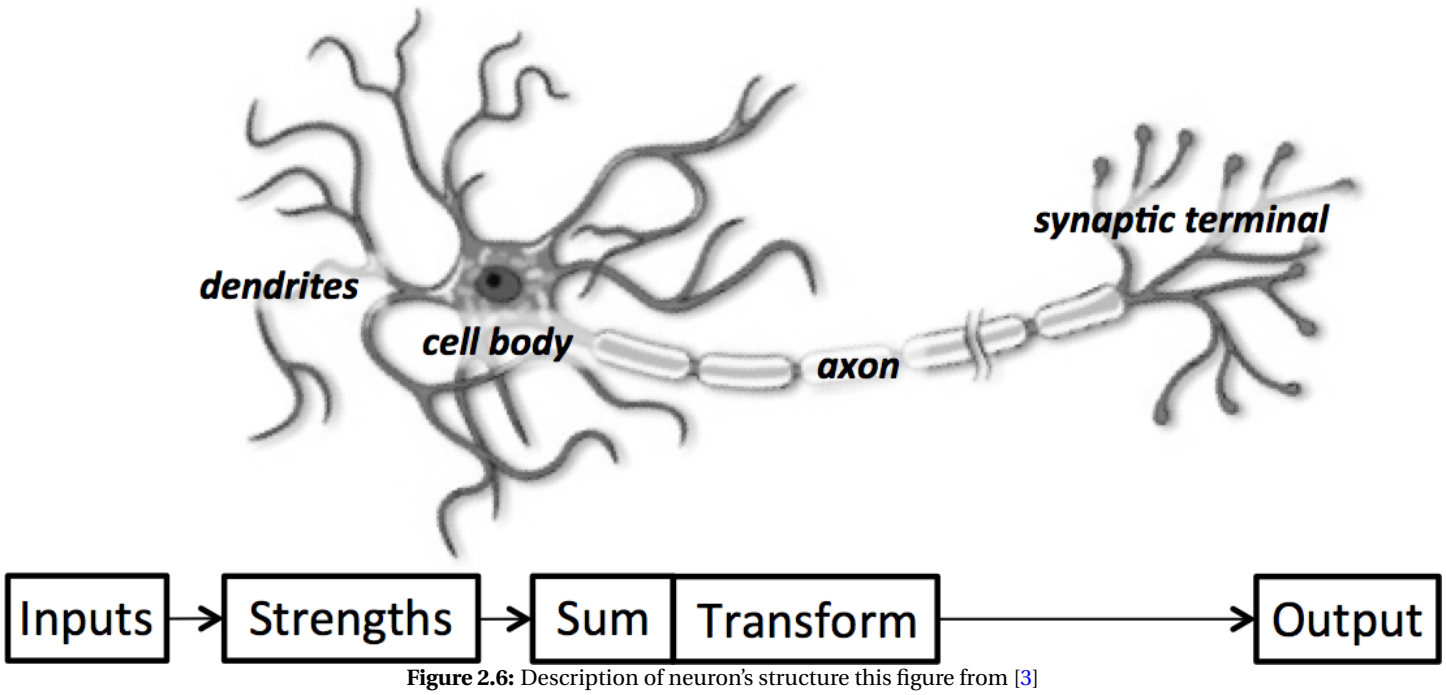


Figure 2.6: Description of neuron's structure this figure from [3]

knowledge about the world around us. The neuron is work by receiving the information from other neuron and process it uniquely then pass the output to other neurons this process is shown in figure 2.6.How do we learn a new concept? *The neuron receives its input from dendrites. The incoming neuron connection is dynamically strengthened or weakened based on how often it is used, and the strength of each connection determines the contribution of the input to the neuron's output. Based on the connection strength it will have weight then the input is summed in the cell body. This sum is transformed into a new signal which is propagated along the cell's axon and sent to other neurons[3].*

The above biological model can be translated into an Artificial Neural Network as described in figure 2.7 We have an input $x_1, x_2, x_3, ..., x_n$ every input has its own strength (weight) $w_1, w_2, w_3, ..., w_n$. We Sum the multiplication of X and W to get the logit of the neuron, $z = \sum_{i=0}^{n} x_i w_i$. The logit is passed throw a function f to produce the output $y = f(z)$ the output will be the input to other neurons. Note: In many cases, the logit can also include a bias constant. So, in this case the function will be

$$y = f(\sum_{i=0}^{n} x_i w_i + b)$$

2.2.3 The Neural Network Representation

As explained previously, We have been trying to simulate the human brain model into our research work in Deep Neural Network. So, We will have multi-layers to allows the model to get in-depth knowledge and more computation performance to simulate the human brain. Now, we will represent the functions per layer as below equations where *l is refer to layer number, i refer to the node number in the layer*(2.19)

$$\boxed{z^l = W^l x + b^l} \longrightarrow \boxed{a_i^l = \sigma(z^l)} \longrightarrow \boxed{\ell(a^l, y)} \tag{2.19}$$

What is the Neural Networks component?

Figure 2.7 represents an Example of Neural Networks representations. It consists of

- **Input Layer:** Input layers is the input data raw for the network it is denoted as a^0.
- **Hidden Layers:** The layers between the input layers and the output layer it can be any number of layers. It also has a set of weighted input and produces an output through an activation function. Every layer in the hidden layer transmits the output to the other hidden layer as an input feature.
- **Output Layer:** It is one output layer with have the final results from the hidden layers.

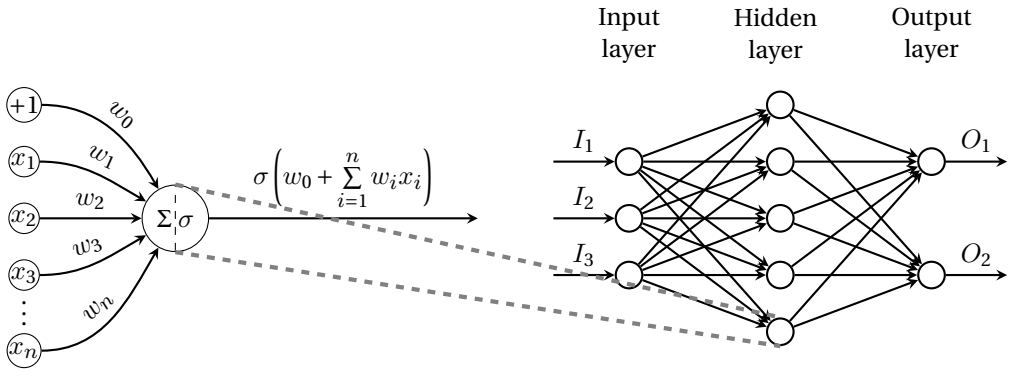


Figure 2.7: Neural Networks Example Input layer with N input samples, One hidden Layer, and an Output Layer.

2.2.4 Neural Network Computation

In this subsection, We will show as example on how we can compute the Neural Networks for every layer. In figure 2.7 we have an example of Input layer of N input, one hidden layer, and one output layer. We will continue explain on this example in Equation (2.20).

$$Z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, a_1^{[1]} = \sigma(Z_1^{[1]}) \tag{2.20a}$$
$$Z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, a_2^{[1]} = \sigma(Z_2^{[1]}) \tag{2.20b}$$
$$Z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, a_3^{[1]} = \sigma(Z_3^{[1]}) \tag{2.20c}$$
$$Z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, a_4^{[1]} = \sigma(Z_4^{[1]}) \tag{2.20d}$$

If we need to compute the above equations it will be simply be represented as vectorized way below matrix shows how we can implement it.

$$z^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_3^{[1]} \end{bmatrix}$$

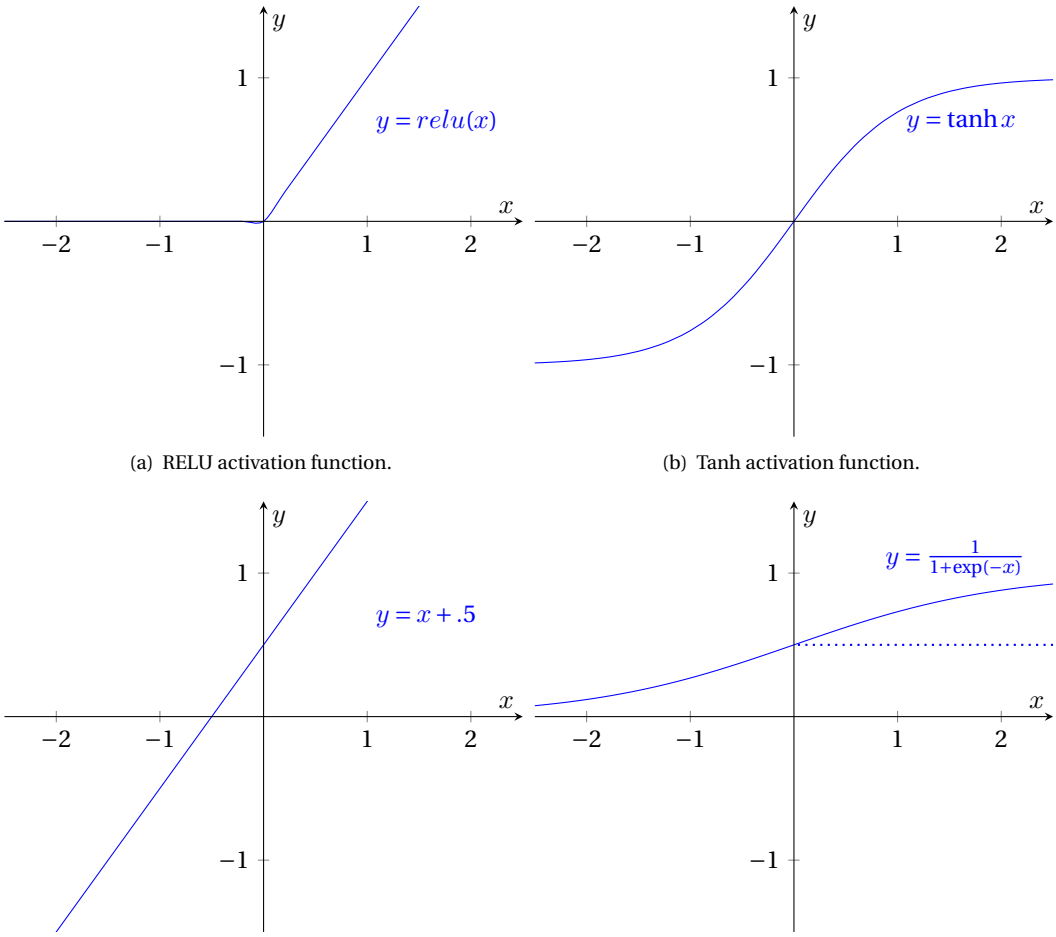
2.2.4.1 Linear Neurons and Their Limitations

Now, We explained the equations for the feedforward Neural Network. We have only one point we need to discuss it which is the Activation function. Let's assume we will continue use linear function in Figure 2.8(c) which represent linear equation $y = wx + b$. So, if we have mutli-layer networks for example Equation (2.21) it will end as linear function because composition of two linear function will be linear function. So, we will not compute deep computation and we will get limited information from the networks. So, to be able to detect the deep information we will use different functions for the hidden layers example: tanh Figure 2.8(b) and its Equation (2.22). Another option is Sigmoid Figure 2.8(d) and Equation (2.1). Finally, Relu Figure 2.8(a) Equation (2.23). Most of binary classification problems use Sigmoid function for output layer. Also, we can use the same functions for the output but we can also use the linear for activation function in some cases.

$$Z^{[1]} = w_1^{[1]T} x + b_1^{[1]}, a_1^{[1]} = \sigma(Z_1^{[1]}) \tag{2.21a}$$
$$Z^{[2]} = w^{[2]T} a^1 + b^{[2]} = w^{[2]T}(w^{[1]T} x + b^{[1]}) + b^{[2]} \tag{2.21b}$$
$$= (w^{[1]T} W^{[2]T}) x + (w^{[2]} b^{[1]} + b^{[2]}) \tag{2.21c}$$
$$= W' x + b' \tag{2.21d}$$

$$a = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.22}$$

$$a = \max(0, z) \tag{2.23}$$



(a) RELU activation function.

(b) Tanh activation function.

(c) Linear activation function.

(d) Logistic sigmoid activation function.

Figure 2.8: Common used activation functions include the logistic sigmoid $\sigma(z)$, the hyperbolic tangent $\tanh(z)$, the rectified hyperbolic tangent RelU $Relu(x)$, and linear function.

2.2.4.2 Softmax Output Layers

Sometimes our problem has multi-output results not only 1 or 0. For example, we have a problem to recognize the characters from 0 to 9 in MNIST dataset, But we will not be able to recognize digits with 100% confidence. So, we will use the probability distribution to give us a better idea of how confident we are in our predictions. The result will be an output vector of the form of the $\sum_{i=0}^{9} P_i = 1$ This is achieved by using a special output layer named softmax layer. This layer is differ from the other as the output of a neuron in a softmax layer is depending on the output of all the other neurons in its layer. This because its sum of all output equal 1. If we assume z_i be the logit of i^{th} softmax neuron, we can normalize by setting its output to represented from Equation (2.24):

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.24}$$

The strong prediction will have a value entry in the vector close to 1, while the other entries will be close to 0. The weak prediction will have multiple possible labels has almost the equal values[3].

2.2.4.3 Forward-Propagation in a Neural Networks

As an example, in Figure 2.9 we need to calculate the Forward propagation we will follow the below equation (2.25). Note: we assume $X = a^{[0]}$ as initial function notation. Also, $\widehat{Y} = g(Z^{[4]} = A^{[4]}$ as the final output layer.

$$Z^{[l]} = w^{[l]} a^{l-1} + b^{[l]}, A^{[1]} = g^l(Z^{[l]}) \tag{2.25}$$

2.2.4.4 Back-Propagation in a Neural Networks

We explained previously, how neural networks could learn their weights using gradient descent algorithm. In this part, we will explain how to compute the gradient of the cost function.

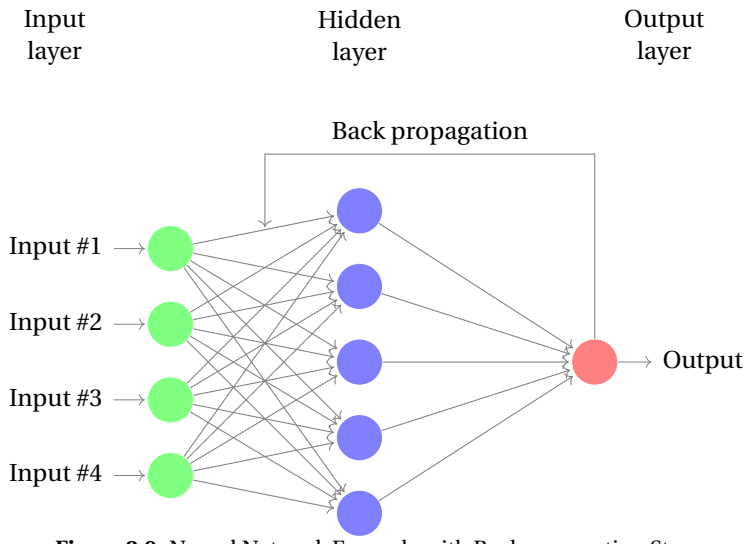


Figure 2.9: Neural Network Example with Backpropagation Step.

To compute the gradient descent in Neural Networks, we use an algorithm named *backpropagation*. The backpropagation algorithm was initially invented in the 1970s, but it wasn't shining until one of the most important papers in this field published in 1986 which describes several neural networks where backpropagation has a significant performance better than the earlier approaches and making it possible to use neural networks to solve problems which were previously not possible to be solved. Now, the backpropagation is the backbone for the learning in neural networks.

The backpropagation not only an algorithm which gives us the expression for partial derivative of the cost function C with respect to wights w and bias b but also it gives is an intuations about the change of the cost function while changing its variables w&b and its effect to the overall network 2.9.

As explained in logistic regression section (2.2.1.5) how we can calculate the derivatives for logistic regression with one layer using this equations(2.13),(2.14),(2.15),
(2.16),(2.17),(2.18).
We will generalize the derivatives equations to be for l layers from the below equations(2.26).

$$dz^{[l]} = da^{[l]} \times g^{[l]'}(z^l) \tag{2.26a}$$
$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]} \tag{2.26b}$$
$$db^{[l]} = dz^{[l]} \tag{2.26c}$$
$$da^{[l-1]} = W^{[l]T} \cdot dz^{[l]} \tag{2.26d}$$

We can vectorize the above equation for Neural Network implementation as below equations(2.27).

$$dz^{[l]} = dA^{[l]} \times g^{[l]'}(z^l) \tag{2.27a}$$
$$dw^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T} \tag{2.27b}$$
$$db^{[l]} = \frac{1}{m} \text{np.sum}(dz^{[l]}, \text{axis=1}, \text{keepdims} = \text{true}) \tag{2.27c}$$
$$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]} \tag{2.27d}$$

If we checked the input variable in the backpropagation we will find it is da^l and this is the derivative of (2.4) which we can get it as explained previously from (2.14) this is the formula for final layer in the feedforward step. If we need to calculate the vectorize version of this equation we can use equation(2.28)

$$da = \frac{d\ell}{da} = \frac{d\ell(a,y)}{da} = (-\frac{y^{[1]}}{a^{[1]}} + \frac{1-y^{[1]}}{1-a^{[1]}} \ldots - \frac{y^{[m]}}{a^{[m]}} + \frac{1-y^{[m]}}{1-a^{[m]}}) \tag{2.28}$$

2.2.4.5 How we Initialize the Wights

As we explained previously in Logistic regression, We initialized the weights to Zero. However, in Deep Neural Networks it will not work. Note: It is okay to initialize the Bias to Zero but the wights it will not works. Let's see what will happen if we initialize the weights and Bias to Zero.

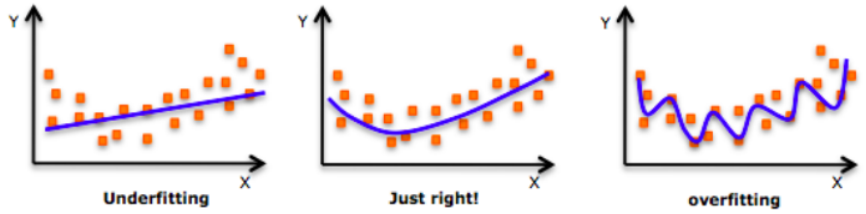


Figure 2.10: Comparison Between different fitting types

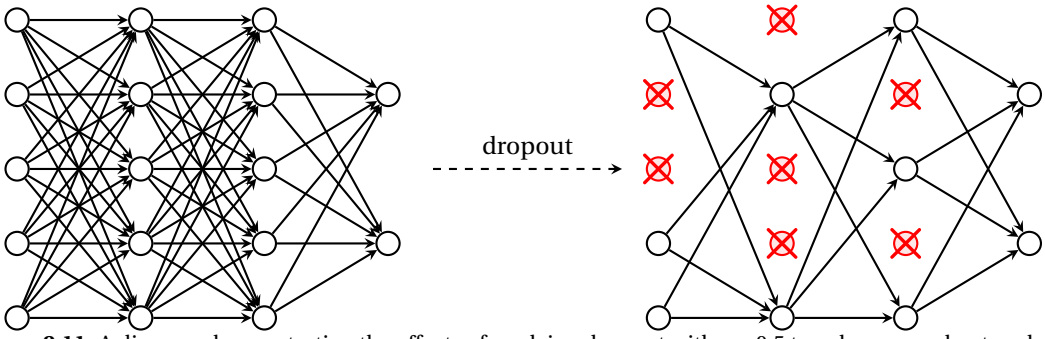


Figure 2.11: A diagram demonstrating the effects of applying dropout with p = 0.5 to a deep neural networks [4]

Assume we have two input vectors x_1, X_2 if we initialize $W^{[1]}$ to Zero from equation(2.29) and $b^{[1]}$ to Zeros. So, $a_1^{[1]} = a_2^{[1]}$ because both of the hidden units compute the same functions. Also, $W^{[2]} = [00]$ Then when we will compute the backpropagation we will find that $dz_1^{[1]} = dz_2^{[2]}$. So, After every iteration, we will find that the two hidden units calculate the same function and we will not get more information from this Deep Neural Network. We need to highlight that the main idea from Neural Networks as explained before is every hidden unit should work to get a new piece of information. The more hidden unit, the more hidden information we will get but if we initialize it to Zero. It will be the same function which is calculated, and we will not get any new information.

$$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.29a}$$

$$W^{[2]} = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{2.29b}$$

$$a_1^{[1]} = a_2^{[1]} \quad dz_1^{[1]} = dz_2^{[1]} \tag{2.29c}$$

$$dw = \begin{bmatrix} u & u \\ v & v \end{bmatrix} \quad W^{[1]} = W^{[1]} - \alpha dw \tag{2.29d}$$

To initialize weights and to get the maximum value of the neural network computation we should initialize the weight by any small random numbers to avoid the big weights which will tend to get the small slope from the Z where $Z^{[1]} = W^{[1]}X + b^{[1]}$ For example, if we use tanh we will get the big tail values $a^{[1]} = g^{[1]}(Z^{[1]})$. So, the big weights we more likely to get slow learning rate.

2.2.4.6 Regularization

One of the most common problems anyone working with data science in general or deep learning face is the overfitting problem. In statistics, overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may, therefore, do not fit additional data or predict future observations reliably [18]. An overfitted model is a statistical model that contains more parameters that can be justified by the data. The essence of overfitting is to have unknowingly extracted residual variation (i.e. the noise) as if that variation represented underlying model structure. In a practical example When model performed working fine on training data but could not predict test data. This means that the models are often free of bias in the parameter estimators, but have estimated (and actual) sampling variances that are needlessly large of the training and not able to be generalized to predict testing data. In figure 2.10 It shows different fitting ways, We need to avoid both underfitting and overfitting. We were trying to enhance the results in our training sets the model is more complex. So that, the training error reduces but the testing error doesn't. While building the neural networks the more complex models the more overfitting problem which we can face.

We can enhance the results on testing data by getting more sample data. So, the model can learn the pattern from the data in a better way. But sometimes it is difficult to get more data, or it is more expensive than the enhancement you need to do. So, we work to apply regularization techniques to prevent the overfitting. Regularization is a technique which applies slight changes or modifications to the algorithm to be more generalized. Sometimes we try to hide information or adding noise on the training data to avoid the overfitting on training phase. This made the model performance in the testing data or unseen data much better.

Different Regularization Techniques in Deep Learning

In this part, We will explain how to apply different regularization techniques in deep learning.

- **L2 & L1 Regularization** are the most common types of regularization. They update the general cost function by adding another term (2.30) known as the regularization term [19].

$$J(w,b) = \ell(y,\widehat{y}) + Regularization \tag{2.30}$$

Addition of this regularization term decrease the values of weight matrices. This happens because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent. So, assume we reduce the value of λ to be close to zero this will produce a simpler network and lots of nodes will be equal zeros. This will reduce the high variance so, less prone to overfitting.

1. L2 regularization is the most common type of regularization. In L2 Regularization We add λ as regularization parameter. This hyperparameter value is optimized for better results (2.31). It also known as weight decay as it forces the weights to decay towards zero (but not exactly zero) [19].

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\ell(y,\widehat{y}) + \frac{\lambda}{2m} \times \|w\|^2 \tag{2.31}$$

2. In L1 Regularization, We penalize the absolute value of the weights (2.32). If we use L1 regularization W can end up being sparse which means w vector will have a lot of zeros. This can help to compress the model because the set of parameters are zero and we will need less memory to store the model. Otherwise, we usually prefer L2 over it.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\ell(y,\widehat{y}) + \frac{\lambda}{2m} \times \|w\| \tag{2.32}$$

- **Dropout** Dropout is one of the most powerful techniques to apply regularization. It produces excellent results and is consequently the most frequently used regularization technique in the field of deep learning.

  Figure 2.11 shows the idea of dropout it randomly selects some nodes and removes them along with all of their incoming and outgoing connections. Dropout techniques randomness make the models results seems to be a more generalized model. For each node, we will have a probability of some dropout percentage that this node will be removed with its connections. So, we will have some simpler network architecture. Also, Dropout force the algorithm to not rely on any feature. So, have to spread the weights and make the network trying to learn a different type of features. Due to these reasons, dropout is usually preferred when we have a large neural network structure in order to introduce more randomness.

- **Data Augmentation** is a technique to augment your training data when there is no way to get more data for the more generalized model. Assume we are working on Image classifier problem, for example, we can flipping the images horizontally and adding that also with your training set. So now instead of just this one example in our training set, we can add this to our training example. So by flipping the images horizontally, we could double the size of our training set. Also, we can do different techniques such as rotating the image, flipping, scaling, shifting the images. All of these techniques can help to get a better-generalized model.

- **Early stopping** is a type of cross-validation way where we split some of our training data to be set as the validation set. So, When we show that our performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

After we finalize this part, We need to note that: In our problem based on text, we found the most effective regularization technique was *Dropout*. After we applied the dropout in our experiments most of the results increased by at least 1% and some of the experiments increased by 3.5%.

2.2.4.7 Optimization Algorithms

As explained previously, We used Gradient Descent algorithm to find the minimum loss for our functions. But Gradient Descent is not the only algorithm used for this purpose. In this part, we will introduce another type of optimization algorithm which most commonly used for optimization problems special in Deep Learning which named *ADAM Optimization Algorithm*.

ADAM Optimization Algorithm

ADAM stands for adaptive moment estimation. It is an adaptive learning rate optimization algorithm designed for training deep neural networks. It published in 2014 at ICLR 2015 [20]. It is one of the most well-designed algorithms for deep learning and proven to work well across a wide range of deep learning architectures. The Adam optimization algorithm is taking Stochastic Gradient Descent with momentum (it used a moving average of the gradient instead of gradient itself) and RMS prop (it uses the squared gradients to scale the learning rate) and putting them together. It derives its name from adaptive moment estimation, and the reason it's called that because Adam uses estimations First is β_1 is computing the mean of the derivatives as the first moment. Second, β_2 used to compute the exponentially weighted average of the squares and that's called the second moment. We can show the algorithm pseudocode in Algorithm 1. The choosing Hyperparameter can be as follow

1. α is the learning parameter and it needs to be tuned based on the problem type.
2. β_1 the default choice is 0.9. So, this is the moving averages in momentum term for (dw,db).
3. β_2 the author of Adam paper recommend being 0.999 this is computing the moving average of dw^2 and db^2 squares.
4. ϵ the author of Adam paper recommended being 10^{-8}

Algorithm 1 ADAM Algorithm for Deep Learning Optimization.

Vdw = 0, Sdw = 0, Vdb = 0, Sdb = 0
for $t = 0$ to $num\_iterations$ **do** ▷ %: compute dw,db on mini-batches
$V_{dw} = \beta_1 V_{dw} + (1-\beta_1)dw, \quad V_{db} = \beta_1 V_{db} + (1-\beta_1)db$ ▷ %:Momentum Step
$S_{dw} = \beta_2 S_{dw} + (1-\beta_2)dw^2, \quad S_{db} = \beta_2 S_{db} + (1-\beta_2)db$ ▷ %:RMS prop Step)
$V_{dw}^{corrected} = \frac{V_{dw}}{1-\beta_1^t}, \quad V_{db}^{corrected} = \frac{V_{db}}{1-\beta_1^t}$
$S_{dw}^{corrected} = \frac{S_{dw}}{1-\beta_1^t}, \quad S_{db}^{corrected} = \frac{S_{db}}{1-\beta_1^t}$
$w := w - \alpha \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}} + \epsilon}, \quad b := b - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \epsilon}$
end for

2.2.5 Recurrent Neural Networks (RNNs)

Deep Neural Networks shows its ability to solve many problems. However, in some use cases, Naive Neural Network architecture cannot works or get the expected results. One of the famous example related to this issue in the NLP tasks when working on a text problem for example, If we say our Harry is the king and Elizabeth is the queen, and we need our model to understand from the sentence that, Harry is he and Elizabeth is she. Also, if this word appears again, we need the model to detect that Harry is a person. This type of problem has a dependency on the input text and how to get the output prediction based on the provided information from the input.

As explained previously, Most of the research in this area trying to simulate human brains. So, we will not find anyone every time trying to think about something start from scratch it always starts from another related point. Example, What is the human do if he tries to connect the information to generate the knowledge about something.

RNN shows its ability to work on sequence data and its related application problems such as natural language[21]. showed the effective of RNN on language modeling. There are many problems which based on this idea of dependency. For example,

- Time series anomaly detection.
- Speech recognition.
- Music Composition.
- Image captioning.
- Stock market prediction.
- Translation.

So, What are the problems in the Naive Neural Network architecture?

- Input and output length can be the different length in a different example.

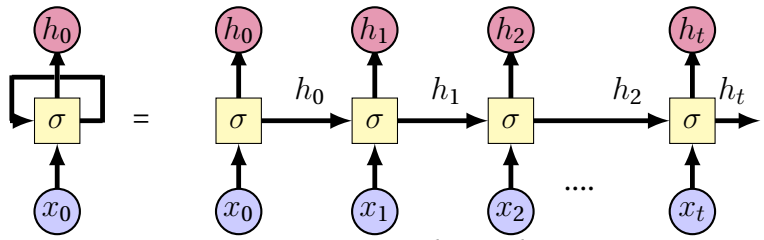


Figure 2.12: Recurrent Neural Networks Loops[5]

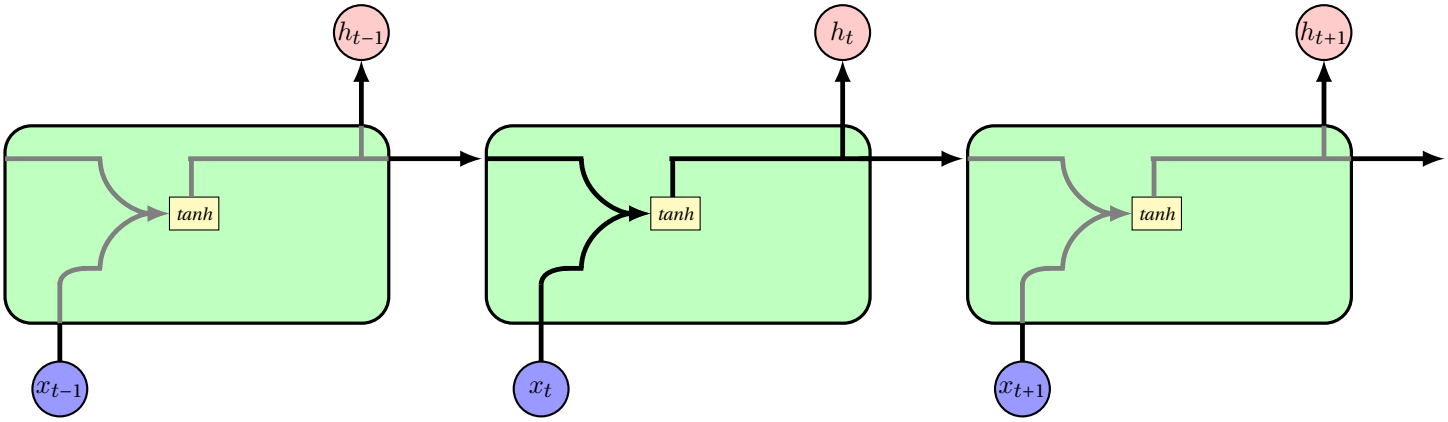


Figure 2.13: The repeating module in a standard RNN contains a single layer.[5]

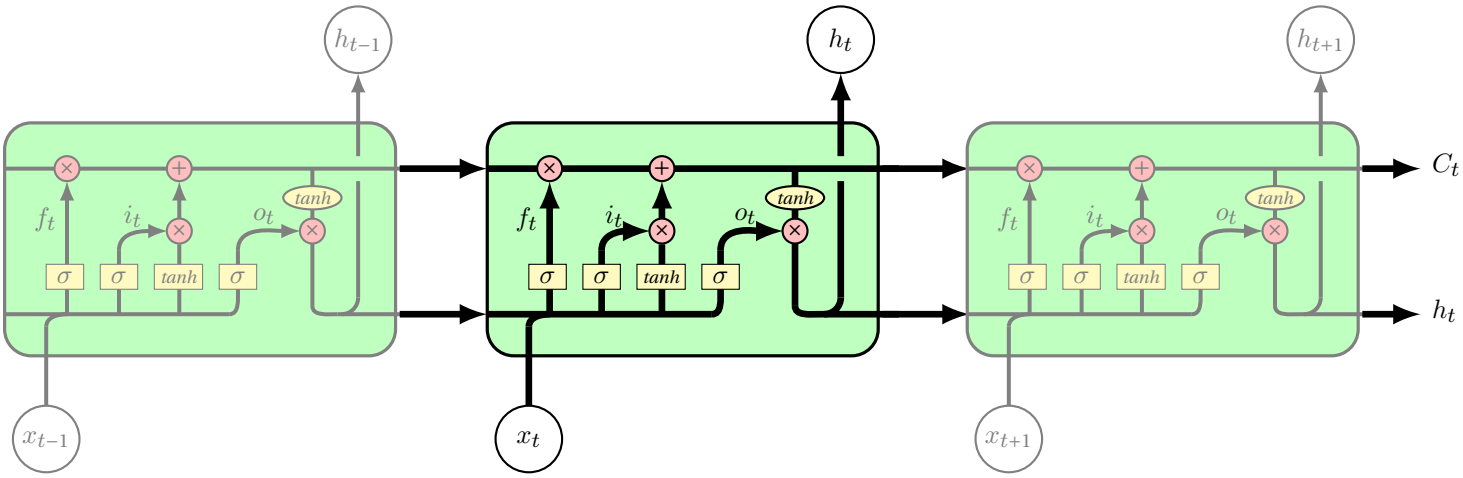


Figure 2.14: The repeating module in an LSTM contains four interacting layers.[5]

- The most important issue is that the Naive architecture cannot share features learned across different positions of text. In this case, we will lose the learned feature, and the lake of dependency, in this case, will affect the overall performance.

What is the new proposed architecture which can provide a way to share the features between the Network?

- First, Assume we have input features x_1, x_2, x_n in the old architecture we input all these features to the Neural Network but now we will input for example x_1 and take the output activation from $a^{<1>}$ to be a feature input with x_2 then take the output activation from $a^{<2>}$ as input to x_3 similar till x_n figure 2.12 shows an example. So, This new change will allow us to share the learned feature between the networks input data. Also, we can thing about it as multiple copies of the same network, each passing a message to a successor[5].
- Second, The feedfarward will compute for time t and then we will calculate the loss at step t. The final loss is the sum of loss at every step t (2.33) explains feedfarward Steps. The backpropagation also will be calculated though time at every step.

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.33a}$$
$$= g(W_a[a^{<t-1>}, x^{<t>}] + b_a) \tag{2.33b}$$
$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y) \tag{2.33c}$$
$$\ell^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -(y^{<t>}\log\hat{y}^{<t>} + (1-y^{<t>})\log(1-\hat{y}^{<t>})) \tag{2.33d}$$
$$\ell(\hat{y}, y) = \sim_{t=1}^{T_m} \ell^{<t>}(\hat{y}^{<t>}, y^{<t>}) \tag{2.33e}$$

2.2.5.1 Vanishing Gradient with RNNs

As we explained, RNN works on sequential data, and the idea is to predict new output not only based on the input data vector but also, other input vectors. Due to the recurrent structure in RNNs, it tends to suffer from long-term dependency to simplify this point let's have an example, the following sentence
*Waleed Yousef who is Associate Professor at Helwan University and teaching Data Science courses and its dependencies **was** got Ph.D. in Computer Engineering from GWU at 2006..*

In the previous example, to predict the word was is depending on long dependency to check if Waleed is singular or not to be consistent. Also, shows how some problems need the long-term dependencies handling.[Bengio et al.,1994][22] showed that Basic RNNs has a problem in long-term dependency. Another problem which may happen into basic Neural Networks is gradient exploding. One of the side-effects of gradient exploding is exponentially large gradient which causes our parameters to be so large. So, the Neural Networks parameters will have a server problem. Another fetal problem with Basic Neural Networks is overfitting problems [Zaremba et al., 2014][23].

So, to solve this learning problem [Hochreiter and Schmidhuber, 1997] introduced Long Short-Term Memory which helps to reduce the dependency problem using memory cell and forget gate.

2.2.6 Long Short Term Memory networks (LSTMs)

Long Short Term Memory networks – aka "LSTMs" – are a special type of RNN, capable of learning long-term dependencies. To solve the vanishing gradient problem for long-term dependencies, [Hochreiter and Schmidhuber, 1997][24] suggested new cell architecture for RNN by adding Long Short Term Memory which significantly reduced the long-term dependency problem using memory cell and forget gate.

LSTMs designed to help solving the long-term dependency problem and to hold information in memory for long periods of time. It also, use same RNNs sequential model but with adding some gating mechanism structure to every cell.

Both Basic RNNs and LSTM have the form of a chain of repeating modules of neural network. The main difference is the structure of the Networks.

In Basic RNNs it is very simple structure for every layer with simple output function 2.13. But in LSTMs it has four interacting layers Figure 2.14.

2.2.6.1 LSTM Gate Mechanism

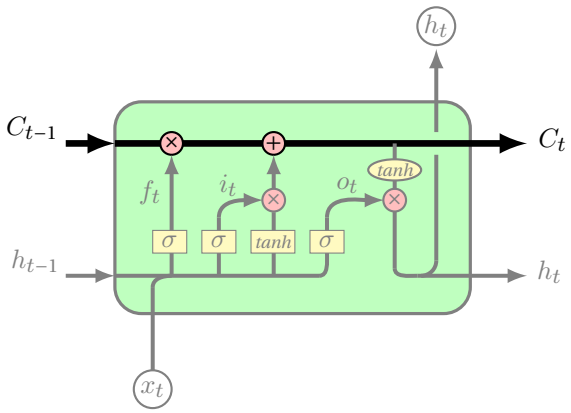
The main component of LSTM is the cell state; It allows the information to pass through along it unchanged. In figure 2.15(b) the top line show the information flow through the cell. The LSTM cell can add or remove information to the cell state using the Gating mechanism.

Gates's idea is a methodology to manage the way how and which information pass or not. It controls information flow through the cell. It has three of these gates. They are consist of a sigmoid neural network layer 2.15(a) and a pointwise multiplication operation 2.15(b).

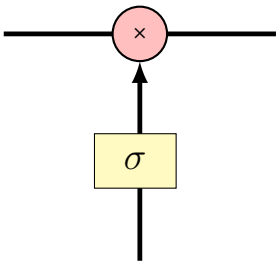
Sigmoid function output values between zero and one. If the value is one these means that everything should pass, while if the value is zero these means do not pass anything. So, the value output from the sigmoid function refers to the amount of each component should be passed.

2.2.6.2 How LSTM Works?

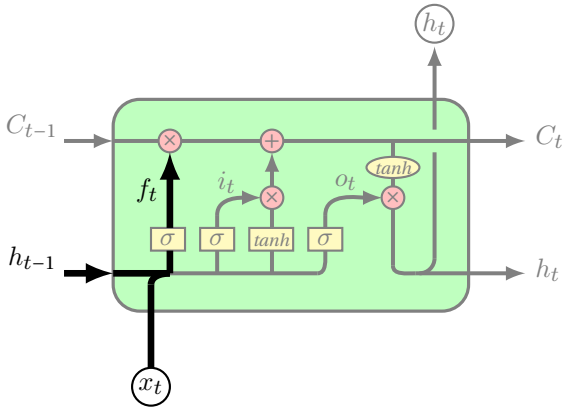
We have explained LSTM has three gates with some



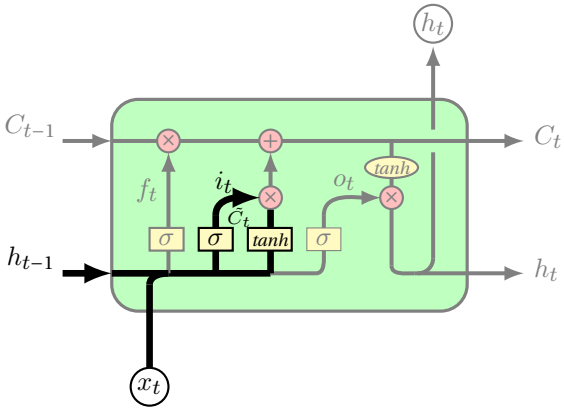
(a) Top line is the medium for information flow



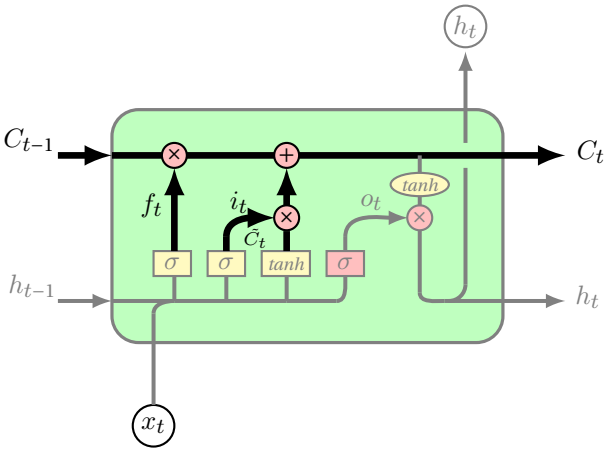
(b) Pointwise Multiplication Operation



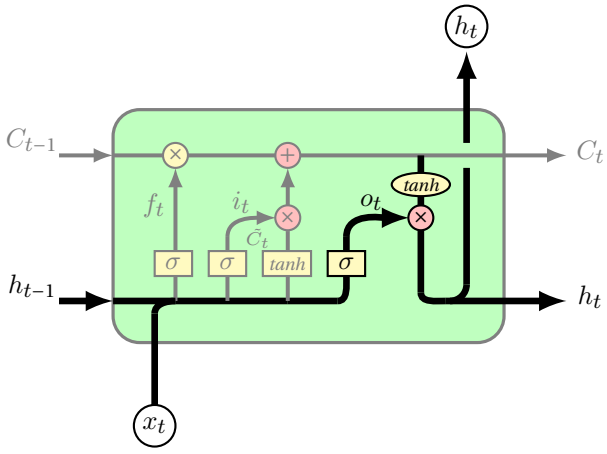
(c) LSTM sigmoid forget gate



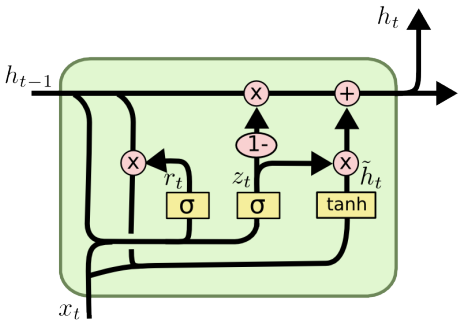
(d) LSTM Input gate



(e) Multiplication and Addition Operation in LSTM.



(f) LSTM output gate.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

(g) GRU cell architecture.

Figure 2.15: LSTM Gates and Configurations adapopted from [5].

- **Forget Gate Layer** a Sigmoid layer Figure 2.15(c) decides which information will be allowed to pass and which will not. It looks at h_{t-1} and x_t, and calculate the output from Sigmoid function between zero and one. As explained if one *everything should pass*, while if zero *do not pass anything*. The value zero or one depends on the value of the cell state if it includes a gender type and we need to predict the pronouns so, it will pass else it will ride of this state (2.34).

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_t) \tag{2.34}$$

- **Input Gate Layer** is a combination between *sigmoid layer* which works to decide which values we should be updated, and tanh *layer* creates a new vector of the new information $\tilde{C}_t$ which should be stored for the next state Figure 2.15(d). The previous combination controls the update state as shown in Equation (2.35). This layer used when we have new input information. For example, We have a new subject named Elizabeth we need to store it for the next input. The next step is the pointwise multiplication and addition operations.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.35a}$$
$$\widetilde{C_t} = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.35b}$$

- **Multiplication and Addition operations** This step is to apply the actions recommended by the previous gates as shown in Equation (2.36). This step is the actions applying the forget of the old information and add the new information, as we decided in the previous steps. Let's look into the upper line in Figure 2.15(e) there are two operations,

  1. Multiplication Operation: This operation to apply the forget gate step by multiplying the old state $\tilde{C}_{t-1}$ by the f_t.
  2. Addition Operation: This operation will add the output from the previous multiplication with the new input information scaled by how much we need to update each state value $i_t \times \tilde{C}_t$

$$C_t = f_t \circ c_{t-1} + i_t \circ \widetilde{C_t} \tag{2.36}$$

- **Output Gate** This gate is a combination of *sigmoid* layer and tanh layer. *Sigmoid* layer decides the information which should be output. Then the output of the *sigmoid* function will be multiplying with the output of the tanh layer of the cell state. This tanh will make the values between -1 and 1. The output of the multiplication of *sigmoid* and tanh will be the final output as show in Equation (2.37). In practice, this gate responsible for deciding which information should be the output. For example, if it saw a subject such as Elizabeth, it might want to output a verb to be relevant to her as a singular.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{2.37a}$$
$$h_t = o_t \circ \tanh(C_t) \tag{2.37b}$$

We have explained the normal LSTM. Also, we need to mention that there are much research proposed different modifications of the normal LSTM type. We will not explain all the types, but we will give a small overview of one of these modifications named Gated Recurrent Unit (GRU) in the next part.

2.2.6.3 Gated Recurrent Units (GRUs)

In RNN Gated recurrent units (GRUs) are a gating mechanism, introduced in 2014 by Kyunghyun Cho et al. [25]. It works to overcome the problem for long-term dependencies. It also aimed to solve the vanishing gradient problem from Basic RNNs.It proposed a new architecture Figure 2.15(g) similar than the LSTM but with some major variants as below,

- It combines the forget gate and input gates into a single gate named "update gate and reset gate."
- The GRU unit controls the flow of information without having to use a memory unit. It just exposes the full hidden content without any control.
- It also merges the cell state and hidden state.

The result of this modifications is GRUs are simpler and easier for modifications in the design. GRUs trains faster and in some case, it performs better than LSTMs on less training data mainly in language modeling. However, LSTMs has some benefits over GRUs in case longer sequences than GRUs in tasks requiring modeling long-distance relations.

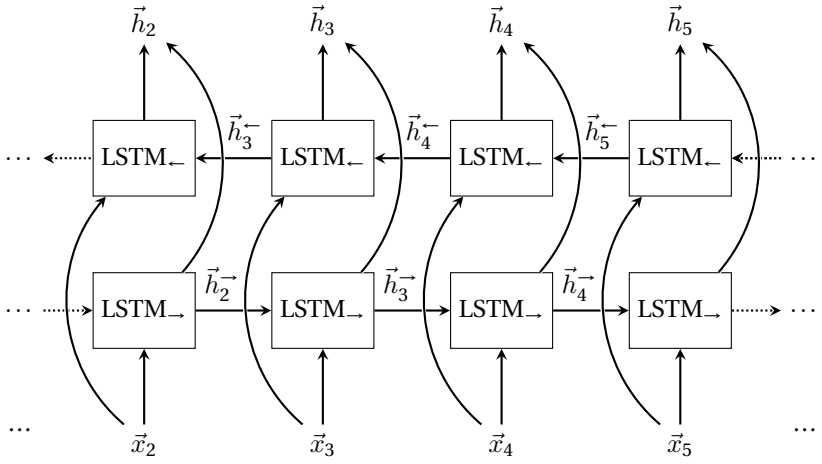


Figure 2.16: Bidirectional long short-term memory [4]

Label	Actual Number	Actual Spam (Positive)	Actual Normal (Negative)
Predicted as Spam (Positive)	200	160	40
Predicted as Normal (Negative)	300	10	290

Table 2.5: Spam vs Normal Email Classifier Example.

2.2.6.4 BI-LSTM

BI-LSTM two LSTMs stacked on top of each other, It used to solve some problem where the information needs to be considered in both directions for LSTM. As the normal LSTM is working from left to right, the BI-LSTM adds the other directions into the learning information. Let's take a motivation example regarding why we need BI-LSTM?

- *Harry is the king, and he will travel next week.*
- *The new book which makes the big sale named Harry Poter.*

Harry in the first example refers to a person, however, in the second example refers to the book. So, if we are working left to right, we will not get the type of word Harry in the second example.

The architecture in BI-LSTM is similar to what we discussed previous regarding Uni-LSTM. We can mention here that BI-LSTM is very slow compared to LSTM, and it needs much time in the training phase, but for example, As we will see later in our research it is impressive regarding the results and the effect in the language problems.

Figure 2.16 represents a recurrent neural network consisting of several LSTM blocks, processing the input sequence simultaneously forwards and backwards (to exploit both directions of temporal dependence)

2.2.7 Machine Learning Model Assessment

As explained previously, Machine learning cycle starting by Data preparation, Feature extraction, Model training and Model assessment 1.1. In this section we will explain the meaning by model assessment. Also, We will discuss different techniques for model assessment.

In Supervised machine learning problems we have two types *Classification and Regression.* Classification the output is discrete variables, for example, Spam vs Normal Email detection. In Regression, The output is a continues variable for example, Predict Housing Price. Each type of problem has its methods or performance evaluation matrix. In this section, we will focus on Classification problem as our problem is a meter classifier application.

How can we measure the model performance? The good is good when the difference between the predicted value and the actual is small and it is not overfitting on development dataset.

There are lots of ways to measure the classification performance Before we explain every method we will give a simple example to allow us to understand the output of every method. Let's assume we have a binary classifier which detects Spam vs Normal Emails. We will explain by example the definition for every type base on the example data in Table 2.5

2.2.7.1 Accuracy

Accuracy: It measures the corrected prediction over the dataset. It calculated using the ratio between the corrected predicted sample from the test data over the total test data sample. Ex: in Table 2.5 *(the total positive predicted as positive + the total negative predicted as negative)/total number of test data* (2.38) which means 90%. There is an issue in Accuracy as a measurement it doesn't give us any sense of the results with respect what is the actual performance of the positive which calculated as positive and vice-versus. So, one measurement which gives us some sense is the Precision and Recall we will discuss it in the next part.

$$\frac{\text{the total positive predicted as positive + the total negative predicted as negative}}{\text{total number of test data}} = \frac{160+290}{500} = 0.9 \tag{2.38}$$

2.2.7.2 Precision and Recall

Precision and Recall are two measurements which answers questions related to our model performance, Some application considers one of them and others required both or a combination. Before we explain it, We should prepare a data table named Confusion Matrix similar to the example in 2.5.

- Precision (also called positive predictive value) is used to answer the question, **Based on the test data how many items predicted correctly from the test data sample**. It can be calculated from Equation (2.39).
- Recall (also known as sensitivity) answer another question, **Based on the test data how many from the truly predicted items are truly predicted**. We can calculate it using Equation (2.40).

$$precision = \frac{\text{true positives}}{\text{true positives + false positives}} = \frac{160}{160+40} = 0.8 \tag{2.39}$$

$$recall = \frac{\text{true positives}}{\text{true positives + false negatives}} = \frac{160}{160+10} = 0.941 \tag{2.40}$$

We need to highlight some application could require a focus in precision more than recall and vice-versus. So, We need to choose the right measure based on our problem.

2.2.7.3 F_1 Score

In statistical analysis of binary classification, the F1 score is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the accuracy score. The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0 [26]. We can calculate it using Equation (2.41).

$$F_1 = 2 \times \frac{precision * recall}{precision + recall} \tag{2.41}$$

2.2.7.4 Per-Class Accuracy

Per class, accuracy is one of the important accuracy measures when we have multi-class. There is a difference between each accuracy calculation regarding the dataset classes distribution. Most of the performance measurement calculate the average accuracy per-class but the difference is some of them take into consideration the size of each class and some don't take the size. So, in case we have imbalanced dataset we should use the type which considers the class size for the accuracy of calculations. In case the data is imbalanced the results could hide information about how the model is performing. For example, assuming the model working perfectly with a class which has the most amount of our data and performing badly in the other class the accuracy can give us results high. However, if we use the average per-class, it will give us a more clear vision of how the model is performing regarding the dataset size for each class. Below is common types used in most of the framework to calculate accuracy per-class. Note: The below naming is followed the same as *Sklearn* Library.

- Weighted: Calculates the F1 score for each class independently but when it adds them together uses a weight that depends on the number of true labels of each class (2.42).

$$F_{1class1} \times W_1 + F_{2class2} \times W_2 + F_{3class3} \times W_3 \tag{2.42}$$

- Micro: It uses the global number of TP, FN, FP and calculates the F1 directly (2.43). It doesn't take in consideration the size for each class.

$$F_{1class1+class2+class3} \tag{2.43}$$

- Macro: It calculates the F1 separated by class but not using weights for the aggregation (2.44) which results in a bigger penalization when the model does not perform well with the minority classes.

$$F_{1class1} + F_{2class2} + F_{3class3} \tag{2.44}$$

2.3 Literature Review

Poetry meter classification and detection have not addressed as learning problem or similar our way for solving this problem. In this literature, we can see that they treated the problem mainly as deterministic problem. They restricted by some static conditions and not able to build a scientific approach to satisfy the problem.

2.3.1 Deterministic (Algorithmic) Approach

[27] present the most related work to our topic, classifying Arabic poetry according to their *meters*. However, they have not addressed it as a *learning problem*; they have designed a deterministic five-step *algorithm* for analyzing and detecting meters. The first step and the most important is to have the input text carrying full diacritics; this means that every single letter must carry a diacritic, explicitly. The next step is converting input text into *Arud writing* using *if-else* like rules. *Arud writing* is a pronounced version of writing; where only pronounced sounds written. Then metrical *scansion* rules applied to the *Arud writing*, which leaves the input text as a sequence of zeros and ones. After that they defined each group of zeros and ones as a *tafa'il*, so now we have a sequence of *tafa'il*. Finally, the input text classified to the closest meter to the *tafa'il* sequence. 82.2% is the classification accuracy on a relatively small sample, only 417 verse.

[28] has taken a similar approach to the previous work, but it replaced the *if-else* by *regular expressions* templates. This approach formalized the *scansion*s, *Arud* based on lingual rules related to pronounced and silent rules, which is directly related to *harakat* as *context-free grammar*. Only 75% from 128 verses were correctly classified.

[29] have taken a similar approach but worked on detecting and analyzing the *arud* meters in Ottoman Language. They convert the text into a lingual form in which the meters appear. Their First Step, Converting Ottoman text transliterate to Latin transcription alphabet (LTA). After that, they feed the text to the algorithm which uses a database containing all Ottoman meters to compare the detected meter extracted from LTA to the closest meter found in the database which saved the meters.

Both [27] and [28] have common problems,

1. **The size of the test data** which cannot measure the accuracy for any algorithms they have constructed because it is a very small dataset. Also, a 75% total accuracy of 128 verses is even worse.
2. **The step converting verses into ones and zeros patternso** are probabilistic; it also depends on the meaning, which is a source of randomness. Then treating such a problem as a deterministic problem will not satisfy the case study. It results in many limitations like obligating verses to have full diacritics on every single letter before conducting the classification. This is also the case with [29] work, for their algorithm to work, the text must be transliterated into LTA.

We can summarize the results difference between our work and the previous work in Table 2.6

Ref.	Accuracy	Test Size
[28]	75%	128
[27]	82.2%	417
This article	96.38%	150,000

Table 2.6: Overall accuracy of this article compared to literature.

Chapter 3

DESIGN DATASET AND EXPERIMENTS

In this chapter, we will discuss the Dataset Design, Training and Experiments done in this project. The Dataset Design steps started from acquisition and encoding including the essential pre-processing steps, and the justification for their need. Pre-processing steps are data extraction, data cleansing, data format, data encoding techniques used. Also, it contains comparisons between the three techniques used. The training phase started by exploring what the ratio of Training, Testing, and Validation is. Choosing the correct percentage of training dataset compared to the Testing and Validation in Deep learning differs from normal machine learning structure, and it affects the model performance. The results and Discussion phase It explains the results of all the 192 experiments on our dataset. Also, How we assess our model design.

3.1 Dataset Design

In this section, We introduces the Dataset acquisition and encoding including the essential pre-processing steps, and the justification for their need. Pre-processing steps are data extraction, data cleansing, data format, and data encoding techniques used. Also, it contains comparisons between the three techniques used.

The collection of the dataset was one of the most laborious tasks in this project. There were criteria we were searching to find. These criteria are as follows,

- **Datasets availability:** There are old Arabic references which have a lot of Poems but not all these books were not available in a PDF or a Web pages format, and it was hard to find it.
- **The Poem with diacritics:** There are resources which have Arabic Poems, but it is much harder to find same with diacritics.
- **The amount of the dataset:** To have a successful project with good results we need a massive amount of data. From the previous work, We did not find this amount of data. The maximum number found was 1.5k. However, We were searching for around 1.5M record of classified poetry.
- **Cleansing of this data:** There was a limitation for the datasets which we can consider it, or we can scrap it due to the limitation for the APIs or the ready datasets in this context.

To meet the above criteria and overcome it, We applied following,

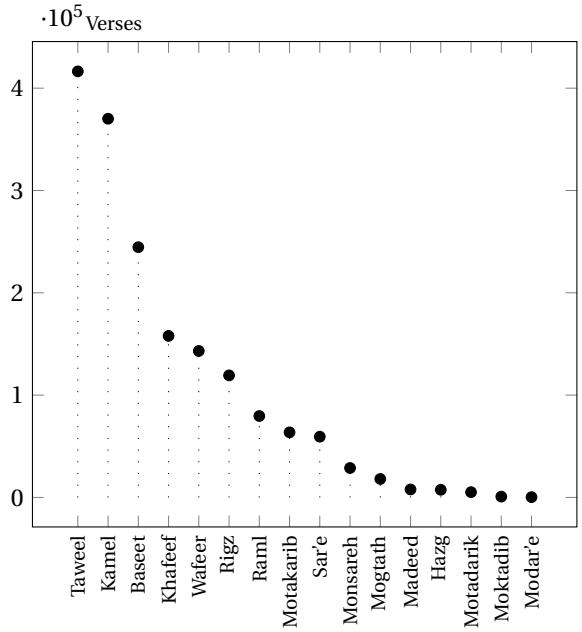
- **Datasets availability:** We have scrapped the Arabic datasets from two big poetry websites: الديوان [30], الموسوعة الشعرية [31]. Both merged into one large dataset, and we open sourced it online [32].
- **The Poem with diacritics:** We tried to get the most verses with the available diacritics, but the diacritics states are not consistent, So, a verse can be fully diacritics, Semi diacritics or without diacritics.
- **The amount of the dataset:** The total number of verses is 1,862,046 poetic verses; each verse labeled by its meter (class), the poet who wrote it, and the age which it was written. There are 22 meters, 3701 poets and 11 ages; and they are Pre Islamic, Islamic, Umayyad, Mamluk, Abbasid, Ayyubid, Ottoman, Andalusian, the era between Umayyad and Abbasid, Fatimid and modern. We are only interested in the 16 classic meters which attributed to Al-Farahidi, and they are the majority of the dataset with a total number of 1,722,321 verses. Figure 3.1 shows the distribution of the verses per meter.
- **Cleansing of this data:** Dataset was not cleaned enough for usage in this research, but we have applied cleansing rules explained in details in Data Preparation and Cleansing section 3.1.2. We also open sourced all the code scripts used in our online repository [33].

3.1.1 Data Scraping

To scrap the data from the website: الديوان [30],ends up into such a problem just reduce your problem to the most smallest one. That means: First: Check if any ”keywords” is set, if used. Then: Use your whole preamble and print the complete bibliography. If this ends up in the same error, your problem might be in your preamble. -> Reduction of preamble, until you get your bib printed. Adding slowly parts back to the preamble, until the error occurs again. That might show you, what lead to the warning.

الموسوعة الشعرية [31], We used custom Python scripts for each websites to get the verses details. The script created with simple usage to pass the link we need to scrap. We will show two examples from both websites.

1. The First example, If we need to scrap a meter from الديوان the website, for example Al-Tawil `https://www.aldiwan.net/poem.html?Word=%C7%E1%D8%E6%ED%E1&Find=meaning`, We will pass this link to the script and the output file name. The script will start scraping and save the output in a CSV format. We can get the output similar than the output in table 3.1
2. Second Example, If we need to scrap the same meter from الموسوعة الشعرية the website for example Al-Raml `https://poetry.dctabudhabi.ae/#/diwan/poem/126971`, We will pass this link to the script and the output file name. The script will start scraping and save the output in a CSV format. We can get the output similar than the output in table 3.2 We scrapped all the available datasets on both websites and merged them based on the common columns. Then we started the Data preparation tasks. We need to mention that, Not all diacritics was correctly available on all the websites. Also, We did not work to generate the diacritics for those datasets. So, we depended on whatever available without changing the data all the next sections is related to correction, preparation, and cleansing of the current datasets.



(a) *Arabic Dataset*

Figure 3.1: Arabic dataset Meter per class percentage ordered descendingly on x axis vs. corresponding meter name on y axis all class in the left of the red line (less than 1% assume to be trimmed in some experiments).

البيت	الشطر الأيسر	الشطر الأيمن	البحر	الشاعر
رَجا شافع نسج المودّة بيننا ولا خيرَ في ودّ يكون بشافع	ولا خيرَ في ودّ يكون بشافع	رَجا شافع نسج المودّة بيننا	الطويل	ابن نباته المصري

Table 3.1: Aldiwan scraping output example

3.1.2 Data Preparation and Cleansing

Data preparation and cleansing tasks divided into multi-stages.

- Merge all scrapped datasets into one CSV file with a selection of the common columns in each file.
- Remove the duplicates rows from the files in case we have any joined rows between both websites.
- Filter the datasets on the 16 meters required as some data belonged to other non-famous or not original meters.
- Remove many unnecessary white spaces which were useless.
- Remove non-Arabic characters and the other web symbols.
- Fix diacritics mistakes, such as the existence of two consecutive harakat, we have only kept one and have removed the other.
- Remove any *harakah* comes after a white space, it removed as it is useless.
- We factored *Shadaa* 1 to its original format explained in this example 2.1 previously.
- We also factored *Tanween* 2 to its original format explained in this example 2.2 previously.[1]

We need to highlight that the last two points are not a handcrafted feature. It is a factorization for the letter to its original format. This factorization will affect the size of the data in the memory and the letter representation in the vector. We will explain this part in details in the next chapter about encoding mechanism and the impact of the encoding type in the model training time and performance.

3.1.3 Data Encoding

As we explained, We have collected the dataset and cleaned the data from any quality issues. The next step is to change the data representation to be ready for model training. This change of the data structure named *Data Encoding*.

[1] *We ignored the factorization of Alef-Mad* آ *in our data preparation and transformation which can save more memory and shorten our encoding vectors*

#	البيت	الشطر الأيمن	الشطر الأيسر	البحر	القافية	الديوان	الشاعر	العصر
1	من يرد مورد حب ظمأ بالشوق يزدد	ظمأ بالشوق يزدد	من يرد مورد حب	الرمل	د	الديوان الرئيسي	يعقوب الحاج جعفر التبريزي	الحديث

Table 3.2: Al-Mosoaa Elshearyaa scraping output example

3.1.3.1 Encoding in English

- **Work embedding Encoding in English** The concept of data encoding was first introduced by [Bengio et al., 2003] [34]. They used an embedding lookup table as a reference and map every word to this lookup. They used the resulting dense vectors as input for language modeling. There are many works to improve the word embedding one of them [Collobert et al., 2011] [35] proposed improvement of word embedding task and proved the versatility of word embedding in many NLP tasks. Another work proposed by [Mikolov et al., 2013 [36]; Jeffrey Pennington et al., 2014 [37]] shows the maturity of word embedding and is currently the most used encoding technique in the neural network based natural language processing.
- **Character Level Encoding in English** All the previous work focused on word embedding encoding, but in our research problem here we do not work on word level we focus into character level encoding as input feature to the model. There is a good deal of research based on the character level encoding [Kim et al., 2015] [38] used character level embedding to construct word level representations to work on out of vocabulary problem. [Chiu and Nichols, 2015] [39] also used character embeddings with a convolutional neural network for named entity recognition.[Lee, Jinhyuk et al.,2017] [40] used character embeddings for the personal name classification using Recurrent Neural Networks.

3.1.3.2 Character Level Encoding in Arabic

Working on Arabic language embedding based on the character level did not take much attention from the research community. [Potdar et al.,2017] [41] has done a comparative study on six encoding techniques. We are interested in the comparison of one-hot and binary. They have used Artificial Neural Network for evaluating cars based on seven ordered qualitative features. The accuracy of the model was the same in both encoding one-hot and binary. [Agirrezabal et al.,2017][42] shows that representations of data learned from character-based neural models are more informative than the ones from hand-crafted features.

In this research, We will make a comparative study of different encoding techniques between binary and one-hot. Also, we provide some new encoding method specific for Arabic letters, and we will see the effect of this on our problem. We will show the efficiency of every technique based on performing model training and model running time performance.

Generally, a character will be represented as an n vector. Consequently, a verse would be an $n \times p$ matrix, where n is the character representation length and p is the verse's length, n varies from one encoding to another, we have used One-Hot and Binary encoding techniques and proposed a new encoding, the **Two-Hot** encoding.

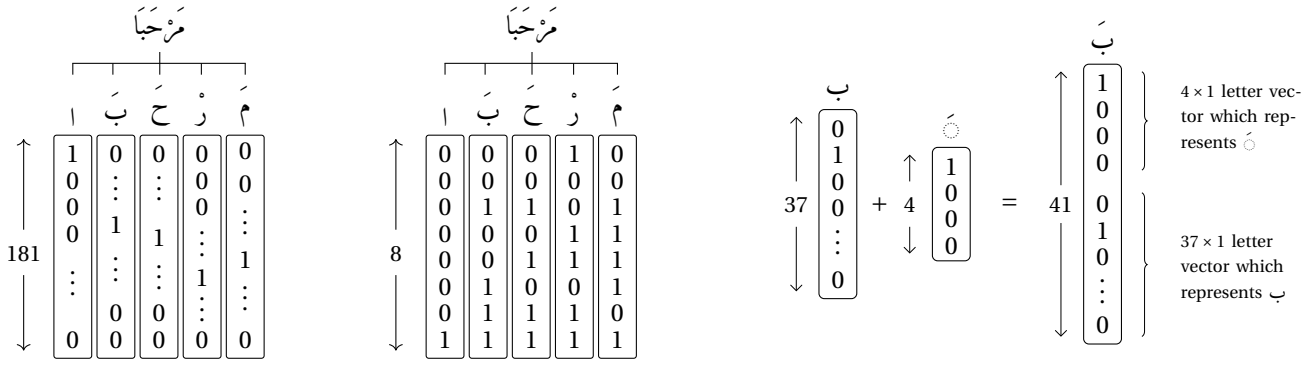
Arabic letters have a feature related to the diacritics; To explain this feature we will take an example based on *One-Hot* encoding. This feature is related to how we will treat the character with a diacritic. Arabic letters are 36 + white space as a letter. So, the total is 37. Any letter represented as a vector 37×1. Let's take an example a work such as مرحبا having 5 letters encoded as a 37×5 matrix. If it came with diacritics such as مَرْحَبًا and we need to represent the letters as One-Hot encoding we will consider every letter and diacritics as a separate letter. So, it will be 5 character and 4 diacritics. The vector shape will be 41×9.

One of the main reason we need to care about the encoding is the *RNN* training. If we have a different number of time steps in *RNN* cell and the input vector dimensions are different based on the input, It will have a standard architecture for the model and to be able to train both the work with diacritics and without diacritics to show the effect of the model learning on the same architecture.

To achieve the model architecture unification, we proposed three different encoding systems: *one-hot*, *binary*, and the novel encoding system developed in this project *two-hot*. The three of them explained in the next three subsections.

One-Hot encoding In this encoding system, We assume the letter with the diacritic as one unit. So, for example, د represented as letter differs than (دَ, دِ, دُ, دْ). Now every letter is represented 5 times one without diacritic and four times with different diacritics combinations 36×5 besides the white-space character. So, the total is 181×1. From now forward, We have 181-characters Arabic alphabet represent the One-Hot encoding, and according to it, we will encode verses. (Figure 3.2).

We need to mention that, One-Hot encoding technique is of the famous techniques in encoding problem. We will not compare the encoding technique in these sections. However, We will discuss it in details in model results. Also, The implementations of the One-Hot is trivial. But we need to focus on here about the size for every letter which is 181×1 which means if we have a verse with 82 characters it will results up with a matrix 181×82 which is very big to be in memory.



(a) *One-Hot* (b) *Binary* (c) *Two-Hot*

Figure 3.2: Different encoding mechanisms

Binary Encoding The idea is to represent a letter with an $n \times 1$ vector which contains a unique combination of ones and zeros. $n = \lceil \log_2 \rceil l$ where l is the alphabet length, and n is the sufficient number of digits to represent l unique binary combinations. For example a phrase like this مَرْحَبَا, it has 5 characters, figure 3.2 shows how it is encoded as a 8×5 matrix, which saves 22.6 times memory than the *one-hot* and reduces the model input dimensions, significantly. But on the other hand, the input letter share some features between them due to the binary representation as it is shown in figure 3.2.

Two-Hot encoding This is an intermediate technique which takes the advantages of the previous two encoding techniques. In which we encode the letter and its diacritic separately as two *One-Hot* vectors, this way the letter is encoded as 37×1 *One-Hot* vector and the diacritic is encoded as 4×1 *One-Hot* vector, then both vectors stacked to form one 41×1 vector (Figure 3.2).

By this way, we reduced the vector dimension from 181 to 41 and also minimizes the number of shared features between vectors to the maximum one at each vector.

3.2 Model Training

In a normal machine learning project, we used to split the dataset as around 60% as training, 30% testing and 10% as a validation. However, In The Deep Learning, the amount of data profoundly affects the model performance. So, the more data fed as training, the more performance results the model can achieve on test data (Assuming we did the regularization and the needed generalization on the model training phase). Also, The main reason to change the split size is due to the size of the dataset we used to work on for example 1% of 1M sample is 10k which is big enough for such experiments way. However, if we work on 10k sample, we need around 30% 3k sample to have confidence in our model. So, it depends on the problem and size of the dataset.

In this research, We worked on *Poem Comprehensive Dataset (PCD)*[32]. We are interested in the 16 classic meters which attributed to Al-Farahidi. These meters comprise the majority of the dataset with a total number of 1,722,321 verses. Figure 3.1 shows an ordered bar chart based on the number of verses per meter. We trained all our models based on 80% which is around 1,377,856 verse. Our testing data(development) is 10% around 172,232 verse, and our Validation data around 172,232 verse.

We can show training phase designed as a Data representation configurations and an RNN configuration. The number of experiments is the cross product of both data representation and RNN configuration, for example, If we have 12 data representation and 12 RNN configurations the total number of experiments will be 144. We need to highlight that,

- Data representation feature is a general feature applied on the dataset not a Hand-Crafted features more details on 3.2.1.
- There is an effect of the Data representation feature due to Arabic language pronunciation and some features provide more information than others.
- RNN configurations are the parameters related to the Network model development training, and it used after many of experiment to find and tune the best configurations. These means we did more than the number of experiment written in this research but we only publish the best results overall. It will explain more details in sec 3.2.2.
- The number verses used on testing and validation is a significant number 344,464 verse which confirms that the model tested on all types of verses and have confidence in the results.

3.2.1 Parameters of Data Representation

Arabic language parameters have types Diacritics, Trimming, and Encoding. Every type has its effect on the data and the performance on model learning rate. We will explain each one in details in the next subsections.

3.2.1.1 Diacritics

Arabic dataset has the verse with diacritics. We can feed the network the characters with diacritics and without diacritics. With diacritics, it will be much easier for the network to learn since it provides more information on the pronunciation. Moreover, It provides more information related to the vowel and consonant sounds in the letters. However, as we discussed in Data Encoding Chapter 3.1.3, Both With and Without diacritics has the same length in input vector size.

3.2.1.2 Trimming Small Classes

Arabic poem dataset, as stated in Figure 3.1, is unbalanced. So, As part of our research, we make the dataset representation as a Full dataset and Trimmed dataset. This way allows us to study the effect of this unbalance. Also, We not only explore the impact of the unbalanced dataset But also, We have applied a technique to solve this issue 3.2.2.1. The trimmed classes are five classes which have less than 1% of the total dataset. We presented these classes as all the classes on the left side of the horizontal red line in Figure 3.1. So, The total classes after trimming are 11 classes and the full with all 16 meters presented.

3.2.1.3 Encoding Techniques

As explained previously, There are three different encoding methods 3.1.3. Although all carry the same information, It expected that Every encoding has its behaviors as below,

- **Running Time**: It was expected the running time would differ from one encoding type compared to others. This information is important if someone has an experiment with limited time and need to get the results as fast as it can.

#	Diacritic	Encoding Types			Trimming
1	With diacritic	One-hot	Two-hot	Binary	Full
2	Without diacritic	One-hot	Two-hot	Binary	Trimmed

Table 3.3: Data Representation Combination Matrix

- **Required Resources**: It was expected to have different resources consumption from one encoding typed compared to others. This information is important if someone has an experiment with limited resources. • Learning Rate: Some encoding will learn faster than others (Note: Learning Rate not the overall performance) for example, one encoding can achieve 80% on training performance after four epoch, but another one can reach the same percentage after 20% epoch.
- **Learning Rate**: The final performance percentage which can be achieved with every encoding technique (Note: the best can be the worst in learning rate or the method which take much time). So, the researcher who will use this encoding should decide which one will be used based on the criteria needed.
- **Overall Performance**: The final performance percentage which can be achieved with every encoding technique (Note: the best can be the worst in learning rate or the method which take much time). So, the researcher who will use this encoding should decide which one will be used based on the criteria needed.

3.2.1.4 Data Representation Matrix

The data representation matrix is the cross product of the Diacritics 2, Data Encoding 3, and Trimming 2 total 12 combinations table 3.3 shows this matrix combination. Example (With diacritic + One-hot + Full)

3.2.2 Parameters of Network Configuration

As explained previously, Recurrent Neural Networks (RNN) showed the ability to solve language model problems in Section 2.2.5. We used RNNs with Long Short Term Memory (LSTM) 2.2.6 as the main architecture for our experiments. We also used BI-LSTM discussed in Section 2.2.6.4 as an alternative way to test the effect of BI-Directional LSTM to check the learning patterns with the two directions.

We also thought using BI-LSTM will support the model to learn the Tafa'il for every class as it can be combined the sound of music from both ways. We will explain our argument and the effects on the results in Section 3.4. In RNNs network configuration parameters, there are four parameters:

- **Cell Type**: We used LSTM and BI-LSTM.
- **Layers**: We tried many numbers of layers and we found the best number based on our problem is 4 and 7 layers.
- **Cell Unit Size**: We also tried many numbers but the best results achieved from 50 and 82.
- **Weighting Model**: As showed in Figure 3.1 classes is unbalanced. So, as an alternative to remove the small classes we tried to keep all classes, but with weighting the loss function to account for the relative class size. We introduced a new weighting function explain in next sub-section 3.2.2.1 to help work on all the dataset. So, we will have two combinations, One with weighting loss and one without weighting loss.

The total number of combination is 16 which is 4 parameters each one has 2 types. So, the total will be $2^4 = 16$; and hence, there are 16 different network configurations to run on each of the 12 data representations above. This results in $16 \times 12 = 192$ different experiments (or models). Hence, there are 96 different network configurations to run on each of the 2 data representations above. This results in $96 \times 2 = 192$ different experiments (or models), whose accuracy are presented on the y-axis of Figure 3.3. For all the 192 experiments networks are trained using dropout of 0.2, batch size is of 2048, with Adam optimizer, and 10% for each of validation and testing sets.

3.2.2.1 Working on Unbalanced data using Weighted Loss

One of the important problem we worked to solve it during our research unbalance dataset issue. We worked to overcome the unbalanced dataset and to not make our model suffer from this issue. We so introduced a Weighting function 3.1 where nc is the sample size of class c, c = 1,2,...C, and C is the total number of classes.

$$w_c = \left(\frac{\frac{1}{n_c}}{\sum_{c'} \frac{1}{n_{c'}}} \right) \tag{3.1}$$

The idea is to increase the loss for the small classes as the number of verses of small classes is small so the output will be bigger than the loss in case the class has a lot of verses. To have a clear explanation we can show equation 3.2 [2] as an example the biggest class will have a smaller loss compared to the smallest class. We divided on constant which is the sum of classes density.

$$w_c = \frac{\frac{1}{288}}{\sum \frac{1}{416428} + \frac{1}{370116} \cdots + \frac{1}{288}} \tag{3.2a}$$

$$= \frac{\frac{1}{288}}{0.00535} = 0.03 \tag{3.2b}$$

$$= \frac{\frac{1}{416428}}{0.00535} = 0.0004 \tag{3.2c}$$

[2] The equation numbers are rounded for simplicity.

3.3 Experiments

3.3.1 Hardware

We used a Dell Precision T7600 Workstation to conduct our experiments with Intel Xeon E5-2650 32x 2.8GHz CPU, 32GB RAM, 1 NVIDIA GeForce GTX 1080 ti GPU[3], Hard desk SSD 256; and with Ubuntu OS, x86_64 Linux 16.04 LTS. We need to highlight that[4]

3.3.2 Software

During our Model development we used the following software and libraries,

- Python 3.7: *We used it as main programming language.*
- Tensorflow: *We used it as Deep learning backend framework*
- Keras: *We used it as High level framework on top of the backend*
- Pyarabic: *We used it in data pre-processing and cleansing.*
- pandas: *We used it in data pre-processing and splitting.*
- sklearn: *We used it to encode the classes using Label-Encoder and for model assessment phase.*
- pickle: *We used it to save the encoder and the model as serialized pickle object.*
- h5py: *We used it to save the encoded dataset matrix in h5 format.*

3.3.3 Implementation Outline

1. Data Reading and Cleansing:
    - We set the random seed and *Numpy* seed in the code to be reproducible.
    - We used PyArabic [43] to trim and strip some dummy letters [5] and working on Arabic letters.
    - We clean the data from any dummy char using *Pandas and Numby* libraries.
    - We get the max Bayt length for padding [6].
    - We removed the dummy letters or wrong diacritics letters.
    - We factorize *Shadaa and Tanween* to its original letters as explained in Chapter 2.
    - We split the data to be Full *Including all the Meters* and Eliminated *Removed the meters which have less than 1% of the total data.*
2. Data Encoding:
    - We encoded the Meters label using label encoder in *Sklearn* and we generated to output label encoder Full and Eliminated.
    - We save the encoder serialized in the desk for later usage.
    - We used *Pickle* python library to save the serialized object with *H5 format* in *h5py* library.
    - We used three ways of encoding *One-Hot, Binary and Two-Hot* all of them developed using custom python functions.
    - We save the data for each encoding with two combinations *Full/Eliminated and With/Without Tashkeel.*
    - We saved them as *h5* format in the desk.

[3] *GPU ASUS ROG STRIX GeForce GTX 1080 Ti Assassin's Creed Origins Edition AC-ORIGINS-ROG-STRIX-GTX1080TI, Memory Type: GDDR5X, Connectors: DisplayPort Output, DVI Output: HDMI Standard Output, Chipset/GPU Manufacturer: NVIDIA, Brand: ASUS NVIDIA GeForce GTX 1080 Ti, Compatible Port/Slot: PCI Express 3.0, Memory Size: 11GB 352-Bit GDDR5X, Core Clock 1594 MHz (OC Mode), 1569 MHz (Gaming Mode (Default)), Boost Clock 1708 MHz (OC Mode), 1683 MHz (Gaming Mode (Default)), 1 x DL-DVI-D 2 x HDMI 2.0 2 x DisplayPort 1.4, 3584 CUDA Cores more details is available in the following website link* https://www.asus.com/us/Graphics-Cards/AC-ORIGINS-ROG-STRIX-GTX1080TI/

[4] *we found a major impact using SSD hard desk when data reading. Beside the effect of the GPU for Deep Learning experiments we utilized the memory and the processors to prepare the batches for the input model*

[5] The data has some dummy letters named Tatweel for example شعـــر is similar than شعر so, we removed this letters as it doesn't have any meaning.

[6] Padding used zeros which is not a selected Arabic Char representations

3. Model Training:

    - We used *Keras with Tensorflow* backend as Deep Learning Framework.
    - We created custom function for handle the Data combinations *Full/Eliminated and With/Without Tashkeel* with the Network combinations *Layers, Units and cell type (Bi-LSTM or LSTM)*.
    - We faced a performance issue while training LSTM for Recurrent Neural Networks as we have a huge amount of data with the current known behaviour of RNN as it is recurrent. So, We did some techniques to reduce our RNN training time by
        - Save the data cleaned and encoded in the hard desk and read it direct encoded as explained in the previous steps.
        - Read the data with batches and utilized the hardware to train the dataset in batch parallel.
        - We used Nvidia Cuda optimized LSTM cell which highly reduced our training time up to 6x speedup.
        - We used a parallel GPU in our experiments to test the affect we used it by utilize a Keras utils *multi_gpu_model option*.
    - We saved all the model output for later validations.

4. Results and Validations

    - We calculate the confusion matrix on the testing data to calculate the results and do the validations.
    - We calculate the accuracy from the confusion matrix which was similar than the accuracy generated from *Tensorflow* on the testing data.
    - We used weighted accuracy to check the accuracy for Per-Class.

3.4 Results

In this section we will explain the results of all the 192 experiments on our dataset. We measure the results using the overall accuracy Then we measure the performance accuracy of the model per class (meter). We will start by present the results for every combinations and then discuss our findings related to the topic.

As we explained, In Section 3 we have a set of combinations we need to explore it. So, most of our results will combine a combination and show the results of these combinations. Let's explore it as below,

1. We have three data representation ***Binary, One-hot, and Two-Hot*** we will represent it as ***BinE, 1D, 0T*** respectively.
2. We have two types of model loss functions ***Weighting loss and no Weighting loss*** we will represent it as ***(1 and 0)*** respectively.
3. The Number of layers is represented as ***nL***, for example, 7 layers are 7L.
4. The Number of cell units is represented as ***nU***, for example, 82 unit is 82U.

So, If we need to explain a set of combination we can write (4L, 82U, 0) which means 4 layers, 82 units, and no weighted loss function. Also, we will provide many figures every figure will explain specific result perspective.

3.4.1 Overall Accuracy

We present the overall accuracy Score in Figure 3.3 of the 16 neural networks configurations and at each of the 12 data representations (y- and x-axis respectively). The x-axis is divided into 4 strips corresponding to the 4 combinations of trimming and diacritic parameters. Then, each strip includes the 3 different encoding values. Each point on the figure represents the overall accuracy score of one of the 192 experiments; (some values are too small, and hence omitted from the figure). To explain the figure, we take as an example the most-left vertical list of points that represent the 16 experiments of the full (no trimming), diacritics, and binary encoding dataset representation. For each rug plot, the highest (Bi)LSTM accuracies are labeled differently as circle and square respectively; and the network configuration of both of them is listed at the top of the rug plot.

To explain the figure, we take as an example the most-left vertical rug plot, which corresponds to (0T, 1D, BinE) data representation. The accuracies of the best (Bi)LSTM are 0.9025 and 0.7978 respectively. The configuration of the former is (7L, 82U, 0W). Among all the 192 experiments, the highest accuracy is 0.9638 and is possessed by (4L, 82U, 0W) network configuration on (1T, 0D, BinE) data representation.

3.4.2 Data Representation Effects

In this section we will explain the effect of the 12 data representation technique we explained it previous.

1. **Trimming Effect:** The effect of trimming (remove the small classes from the training cycle) can be observed if we fix the other two parameters, diacritic and encoding. The score with trimming is consistently higher than that with no trimming. E.g., by looking at the two unshaded strips, the score at (1T, 0D, TwoE) is 0.9629, while that at (0T, 0D, TwoE) is 0.9411. The only exception, with a very little difference, is (1T, 1D, BinE) vs. (0T, 1D, BinE). We need to highlight that is logic to have this effect as the training will have fewer classes with a huge amount of data for these classes.
2. **Diacritics Effect**
   - *Without Trimming:* The effect of diacritics is obvious only with no trimming (the two left strips of the figure), where, for each encoding, the accuracy score is higher for diacritics than no diacritics.
   - *With Trimming:* The diacritics don't have except for the *one-hot* encoding but other encoding doesn't have an effect on the model performance. This result is inconsistent with what is anticipated from the effect of diacritics. We think that this result is an artifact due to the small number of network configurations.
3. **Encoding Effect:** The effect of encoding is clear; by looking at each individual strip, *overall accuracy score is consistently highest for two-hot then one-hot then binary—the* only exception is (1T, 0D, BinE) that performs better than the other two encoding. It seems that two-hot encoding makes it easier for networks to capture the patterns in data. However; we anticipate that there is a particular network architecture for each encoding that can capture the same pattern with yielding the same score.

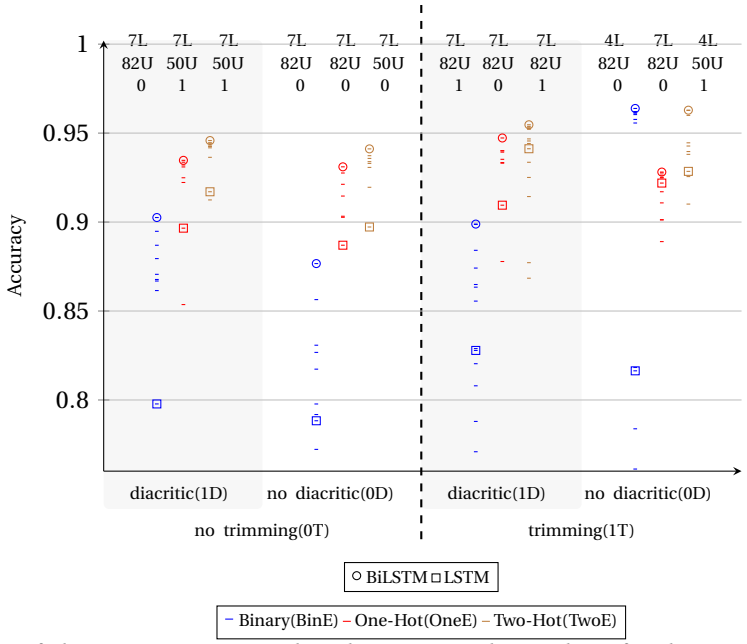


Figure 3.3: Overall accuracy of the 192 experiments plotted as 2 vertical rug plots (for the 12 different data representations: $\{Trimming, NoTrimming\} \times \{Diacritics, NoDiacritics\} \times \{OneE, BinE, TwoE\}$), each represents 16 exp. (for the 16 different network configurations: $\{7L, 4L\} \times \{82U, 50U\} \times \{0W, 1W\} \times \{LSTM, BiLSTM\}$). For each rug plot the best model of each of the four cell types —(Bi)LSTM labeled differently. Consistently, BiLSTM was the winner, and its network configuration parameters are listed at the top of each rug plot.

3.4.3 Network Configurations Effects

This section is to comment on the effect of the network configurations parameters.

- **Cell Type**: It is clear that BI-LSTM (large circle) is the highest accuracy score for each data representation. It is always higher than the highest score of the LSTM model (large square). This is what we expected the more complex architecture, the more results we can achieve. But we need to mention that the BI-LSTM is slower than LSTM in overall running time for all experiments, and it also consumes much more resources than LSTM cell.
- **Layers Number:** As we explained in Section 2.2 The idea behind the deep neural network come from the multi-layers which make the network learn more details. So, the more complex network (more layer) the more results we can achieve. So, in our experiments, we can show that 7 layers achieved the highest scores more than the 4 layers. There is an exception for the trimming data without diacritics in (1T, 0D, BinE) and (1T, 0D, TwoE). The straightforward interpretation for that is the reduction in dataset size occurred by trimming and no diacritics, which required a less complex network. So, if we reduced the complexity of our problem the number of layers will not be effective.
- **Cell Units and Weighting Loss:** We can't figure out a consistent effect based on the number of cell units or the weighting loss. But we need to mention that the highest results achieved were using both the highest cell units 82 and the weighted loss.

3.4.4 Per-Class (Meter) Accuracy

In this section, we will explore the accuracy of each class. This is regarding how our model can detect every class separately. It similar the previous accuracy but per-class calculation. It is also useful to check it as it will show us how the model able to understand every class and what is the classes which our model not able to classify it.

Similar in the previous section, We have four combinations of *trimming and diacritic* we will investigate which models achieved the best results. We will take the best four models (the first three of them is the two-hot encoding, and the fourth is the binary encoding) from Figure 3.3 which is the overall accuracy and show the results of the per class accuracy for each one.

In Figure 3.4 we have the previous four models display the per-class accuracy. We ordered The class names based on their data size per class. It explained previously in Figure 3.1 with the same order. If we compare the results for the four models accuracy scores were around 95% in Figure 3.3 and the per-class accuracy, we will find only 6 classes (which have around 80% of the total datasets) are around this value. But there are significant drops for some classes which make the figures line has dropped in the results.

Figure 3.4 shows the relation between the model accuracy results per-class and the dataset size per class; However, This trend

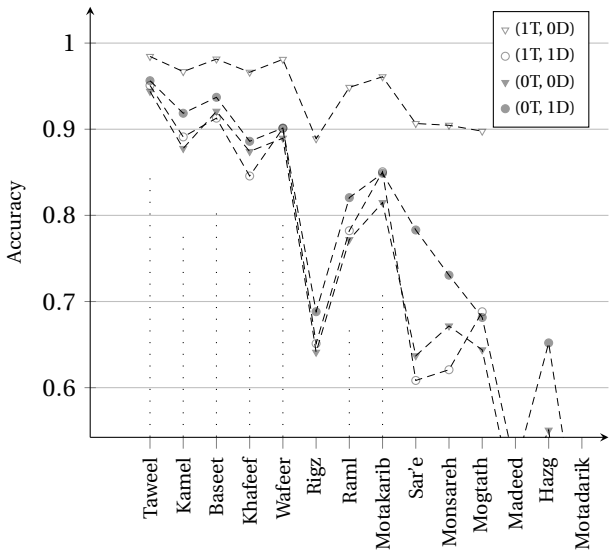


Figure 3.4: The per-class accuracy score for the best four models with combination of ({*Trimming*} × {*Diacritics*}); the x-axis is sort by the class size as in Figure 3.1. There is a descending trend with the class size, with the exception at *Rigz* meter.

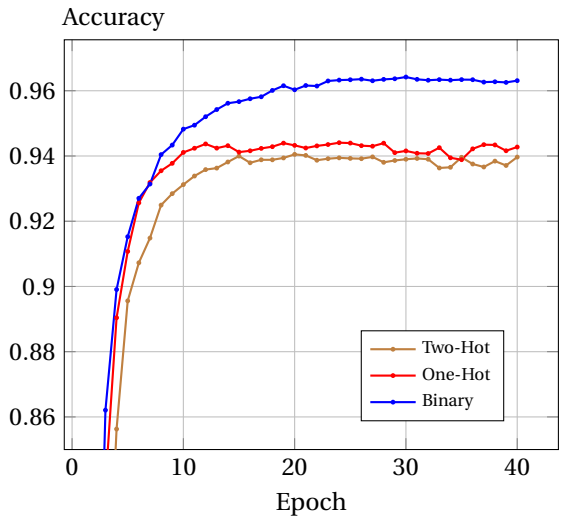


Figure 3.5: Encoding effect on Learning rate with the best model (1T, 0D, 4L, 82U, 0W, BinE) and when using the two other encodings instead of BinE.

was expected to be fixed from the weighted loss which has an inconsistent effect of the weighting loss for all the models. This inconsistent effect shows we need to have a new design for has an function which can solve this trending issue. The overall accuracy can be increased after trimming but there will be a gap between the accuracy per class the size of the data per class as the dataset is not balanced. Moreover, We can repeat this experiment again with enforcing all classes to have an equal size so, we can show the accuracy without in data unbalance issue. We will elaborate more in Section 3.5.

3.4.5 Encoding Effect

As explained in Section 3.1.3 the difference between data encoding types. In this section, we will explore the effects of Data Encoding with respect to the *Accuracy, Learning Rate and Memory Utilization* on the best model results (4L, 82U, 0W, 1T, 0D, BinE). During our experiments, we didn't find a consistent effect on the model encoding type and the model accuracy 3.5-a. However, most of the cases we found the accuracy of the two-hot is slightly better than binary and then one-hot.

Figure 3.5 shows the effect of encoding on learning rate which has no difference in convergence speed between the encoding types; However, We can found some encoding start learning faster than other between epochs[1:5] but overall they will converge with the same learning curve at the end.

Memory Utilization is not similar for each model encoding. We found that each encoding has it is own vector size representation Figure with different memory utilization which is based on the vector size, for example, the vector size of *One-hot* is $181 \times 8(bits)$ it

will output $1,448$ if we compare this encoding with *Two-hot* we will find $41 \times 8(bits)$ it will output 328. If we compare the previous two encoding with *Binary* we will find it is the lowest memory consumption $8 \times 8(bits)$ it will output 64. We can find that *Two-hot* is in the middle between the two encoding with respect to memory consumption and also it gives some more meaning for data encoding as explained before in Section 3.1.3

3.4.6 Comparison with Literature

As explained previous, One of the advantages in our research work is the very large dataset we have which allows us to have a good subset of testing. This provides us confidence regarding our results.

If we compared our work approach results the best model scored 0.9638 with the highest two in literature, We will find that our model results are significantly higher than the others as illustrated in Table 2.6. Moreover, Our approach is a learning approach, not a Hand-crafted algorithmic approach which gives our model more confidence to be mature enough for these types of problems as explained in Chapter 2.3.

3.5 Discussion

In this section, we need to discuss some points regarding our experiments and results approach. We will show some parts we think it needs more discussion or exploration.

3.5.1 Dataset Unbalanced

Our dataset was unbalanced which for sure affect the results we showed we have some significant drops in Per-class accuracy which most of them regarding the data size issue. We think we should have some further work regarding this point to reconstruct the experiments with balanced data, for example, 10k samples per class and check the results. Another approach could be to increase the size of the small classes to be at least 5% of the overall classes percentage this would enhance the learning accuracy of these classes.

3.5.2 Encoding Method

Although all the encoding methods which carry the same information should produce the same results in theory, In practice, Deep Neural Networks showed this is not the case. To explain the reason let's first explain how Neural Network work with different encoding mechanism?

The encoding method is a transformer function $\mathcal{T}$ this function transform a discrete input values X. We can denote to the values as a transformed feature $\mathcal{T}(X)$ the output of this transformer method. The output $\mathcal{T}(X)$ of this transformer in the new encoding space will be input to the Neural Network model. The model should be able to "decode" this type of encoding. Since the lossless encoding is invertible, it is clear for any two functions and any two encodings that $\eta_1(\mathcal{T}_1(X)) = \left(\eta_1 \cdot \mathcal{T}_1 \cdot \mathcal{T}_2^{-1}\right)(\mathcal{T}_2(X))$. This means that if the network η_1 is the most accurate network which can "decode" the encoding function (transformer) $\mathcal{T}_1$ this network η_1 is not a general network which can understand any encoding function. Also, to design this network requires a very complex architecture. So, if we have another encoding function $\mathcal{T}_2$ and we tried to use the same network for the $\mathcal{T}_2$ requires designing another network $\eta_2 = \eta_1 \cdot \mathcal{T}_1 \cdot \mathcal{T}_2^{-1}$. However, this network may be of complicated architecture to "decode" the complicated pattern of $\mathcal{T}_2(X)$.

In general, Any encoding function $\mathcal{T}$ require a special network η to get the correct decoding (learning) for the dataset. So, our comparison between the encoding methods in the same Neural Networks architecture not accurate as each one required different network design. But all of them will reach the same results but with a different time or can be a small difference due to the not accurate network architecture. Moreover, Our work illustrated clearly the effect of the encoding methods and compared between each other, We think the *Two-Hot* encoding is the more suitable method to work with character level problems. It is the middle approach between the *One-Hot* which needs a huge amount of memory and the *Binary* which loss some meaning in Arabic language diacritics effect.

3.5.3 Weighting Loss Function

Our weighting loss functions don't solve the small classes issues (regardless the best model accuracy achieved with weighting loss but this is not a consistent result). The weighting loss function needs to be redesigned to solve this issue with the combination of learning rate and the batch size.

3.5.4 Neural Network configurations

During our work, we show the effect of different network configurations on the model learning and accuracy. We did a lot of experiments to find the best development architecture to make our experiments run faster and be able to do a lot of experiments. In the beginning, Out experiments took around 1.5 hours. Second, we proposed the multi-batch training to utilize parallel processing and prepare the data faster to the model as our data was huge. Then we use enhanced *Cuda* LSTM cell which allows reducing our experiments time overall to be around 7-9 min per epoch based on the architecture of the network.

We also showed the effect of network layers on Learning and accuracy results. So, If we have done more experiments with more deep layers and more complex architecture it can reach more language knowledge and build a more complex model which will enhance both the Per-class accuracy and the overall accuracy.

3.5.5 Model Assessment

In our work, we proposed the overall accuracy score as the model assessment method for the results. But we need to highlight that the overall model accuracy produced from the Deep Neural Networks was very close to the overall accuracy score calculated from the confusion matrix and in some experiments it was almost the same. We also, tried different statistical ways, for example, F_1 score to assess our model and we find it will be the same as the model results.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

This thesis aimed to train Recurrent Neural Networks on Arabic Poem at character level to recognize their meters. This can be considered a step forward for language understanding, synthesis, and style recognition. This step aimed to understand how can Deep Learning models understand the Arabic Poem Meters and its rules and grammar. The datasets were crawled from several non-technical online sources; then cleaned, structured, and published to a repository that is made publicly available for scientific research. To the best of our knowledge, using Machine Learning (ML) and Deep Neural Networks (DNN) in particular for learning poem meters and phonetic style from a written text, along with the availability of such a dataset, is new to literature. We described different encoding mechanism each has its own structure and understanding mechanism. We Also experiment different RNN configurations including (number of layers, cell units, cell types,..) with different language settings (with Diacritics and without Diacritics) We tried to use different ways to handle the imbalanced dataset which shows the different accuracy and the effect of each one. The classification accuracy obtained on our Arabic Poem dataset was remarkable, especially if compared to that obtained from the deterministic and human-derived rule-based algorithms available in the literature.

4.2 Future Work

In this section, We will mention some future work which can be built based on this research. We will split the future work into two parts, One related to this idea and how can we enhance it. Second, related to the new research area which can be built on the dataset we have.

- Enhancement on the current work
    1. Enhance the classification results to be the same as the human expert. We have many areas of enhancement. First, Enhance the network configurations with more layers with combinations of cell units and batch size. Second, There is an open area to solve the accuracy drops in the Per-class performance issue in the small classes using a new design of weighting loss function. Third, It can increase the dataset for the small classes which will affect the learning and understanding of their patterns.
    2. This problem can be treated as unsupervised learning which will be a different approach to the problem-solving.
- Build new work based on the dataset
    1. Use the current datasets to classify the poetry meaning as this paper did not work for this idea.
    2. Generate a new poem from learning the current classes and patterns.
    3. Analyze the historical impact on the Poem and the Poetry for example for a specific period if the poetry affected by this period, or there are patterns of writing between the Poetry or not.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[3] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-generation Machine Intelligence Algorithms*. O'Reilly Media, 2017. [Online]. Available: https://books.google.pl/books?id=EZFfrgEACAAJ

[4] "Collection of latex tikz figures." [Online]. Available: https://github.com/PetarV-/TikZ

[5] Colah, "Understanding Lstm Networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] G. F. Simons, "The 20th Edition of Ethnologue," 2017. [Online]. Available: https://www.ethnologue.com/ethnoblog/gary-simons/welcome-21st-edition

[7] "الدّيْوَانُ," 2013. [Online]. Available: https://www.aldiwan.net

[8] M. Al-Metari, "القواعد العروضية وأحكام القافية العربية , محمد بن فلاح المطيري."

[9] A.-K. A. tabrisi, الكافي في العروض والقوافي.

[10] A. Almuhareb, W. A. Almutairi, H. Al-Tuwaijri, A. Almubarak, and M. Khan, "Recognition of modern arabic poems," *JSW*, vol. 10, pp. 454–464, 2015.

[11] "Verse (poetry) wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Verse_(poetry)

[12] D. R. Cox, "The regression analysis of binary sequences," pp. 213–230, 2005.

[13] "Convex function wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Convex_function#cite_note-1

[14] W. Rudin, *Principles of Mathematical Analysis*. McGraw-Hill Higher Education, 1976.

[15] "Wolfram alpha." [Online]. Available: http://mathworld.wolfram.com/ConvexFunction.html

[16] "Concave function wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Concave_function

[17] M. R. D. G. Richard, *The Secret Life of the Brain*. Joseph Henry Press, 2001.

[18] "Overfitting, wikipedia page." [Online]. Available: https://en.wikipedia.org/wiki/Overfitting

[19] "An overview of regularization techniques in deep learning." [Online]. Available: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[21] T. Mikolov, M. Karafi, L. Burget, J. ernock, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, vol. 2010, no. 9. International Speech Communication Association, 2010, pp. 1045–1048. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=9362

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, March 1994.

[23] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[25] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014. [Online]. Available: http://arxiv.org/abs/1409.1259

[26] "F1 score wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/F1_scorex

[27] B. Abuata and A. Al-Omari, "A rule-based algorithm for the detection of arud meter in classical arabic poetry," *International Arab Journal of Information Technology*, vol. 15, 12 2017.

[28] M. Alnagdawi, H. Rashaideh, and A. Aburumman, "Finding arabic poem meter using context free grammar," *J. of Commun. & Comput. Eng.*, vol. 3, no. 1, pp. 52–59, 03 2013.

[29] A. Kurt and M. Kara, "An Algorithm for the Detection and Analysis of Arud Meter in Diwan poetry," pp. 948–963, 2012.

[30] "الدِّيْوَانُ," 2013. [Online]. Available: https://www.aldiwan.net

[31] "المَوسُوعَةُ الشِعْرية," 2016. [Online]. Available: https://poetry.dctabudhabi.ae

[32] W. A. Yousef, O. M. Ibrahime, T. M. Madbouly, M. A. Mahmoud, A. H. El-Kassas, A. O. Hassan, and A. R. Albohy, "Arabic Poem Comprehensive Dataset," 2018. [Online]. Available: https://hci-lab.github.io/ArabicPoetry-1-Private/#APCD

[33] "Hcilab, arabic poetry," 2018. [Online]. Available: https://github.com/hci-lab/ArabicPoetry-1-Private/

[34] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

[35] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Nov. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078186

[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[37] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[38] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *CoRR*, vol. abs/1508.06615, 2015. [Online]. Available: http://arxiv.org/abs/1508.06615

[39] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *CoRR*, vol. abs/1511.08308, 2015. [Online]. Available: http://arxiv.org/abs/1511.08308

[40] J. Lee, H. Kim, M. Ko, D. Choi, J. Choi, and J. Kang, "Name nationality classification with recurrent neural networks," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2081–2087. [Online]. Available: https://doi.org/10.24963/ijcai.2017/289

[41] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," pp. 7–9, 2017.

[42] M. Agirrezabal, I. Alegria, and M. Hulden, "A Comparison of Feature-Based and Neural Scansion of Poetry," 2017.

[43] T. Zerrouki, "pyarabic, an arabic language library for python," 2010. [Online]. Available: https://pypi.python.org/pypi/pyarabic