Attention mechanism in neural network for early diagnosis in newborns using cry signals

by

Anisan GUNARATHINAM

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR A MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING
M. Sc. A

MONTREAL, 9th OF NOVEMBER 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS MASTER'S THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

M. Chakib Tadj, Thesis Supervisor
Department of Electrical Engineering, École de technologie supérieure

M. Tony Wong, President of the jury
Department of Electrical Engineering, École de technologie supérieure

M. Christian Gargour, Member of the jury
Department of Electrical Engineering, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

OCTOBER 21st 2021

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENTS

I want to thank everyone who contributed to this project and overcome all the obstacles I encountered while working on this master's thesis.

My special thanks go to my master's thesis supervisor, Mr. Chakib Tadj, to guide me and help me make the right decisions at the right moment. I could not have completed this thesis without him and his expertise.

I am grateful to everyone who worked on this project before me and helped develop a database that is filtered from all silence and annotated. This project would have taken longer to realize without it, since it made me focus my time on the neural network part for classification.

Thank you to the open-source Python community for all the inspiration and code that made my development easier.

My greatest thanks go to the nurses who take the time to record newborn's cries and fill their data with the help of experts. I do not think we give enough credits for them, but a neural network is nothing without its data and people who use their time and patience to annotate the data for an efficient supervised neural network.

Last but not least, I would like to thank my parents for their support during these years and for helping me achieve my desired level of education.

Attention mechanism in neural network for early diagnosis in newborns using cry signals

Anisan GUNARATHINAM

ABSTRACT

Humans use their voice to communicate, and it is often driven by instinct. On a larger scale, this same instinct can be applied to a newborn infant who tries to express itself, with obviously, his cry. According to past studies made on this subject, there is a strong correlation between an infant's cry and psychological condition and pathology that affects the latter. This study focuses on finding a correlation and identifying the newborn's condition using a neural network. This tool would help experts identify something that they could have potentially ignored and better diagnose a newborn according to the disease affecting it before it is too late. Whether it is in a developed country or a developing country, this solution requires no expensive diagnosis material.

Our study uses an Attention mechanism neural network that has known success in speech recognition and text translation in the last two years. The Attention layer learns to focus on different aspects of the input. The Transformer uses Encoder-Decoder architecture to summarize the entire input before outputting the results, which are strongly linked with each input sequence. The idea is to eliminate the dependency in the fixed-length input conducted in the previous studies. The Attention mechanism enhanced LSTM (Long short-term memory) is a recurrent neural network that inherits the self-attention layer from the Transformer.

Only the expirations sessions are extracted for each cry sample to generate the matrix of Mel-frequency cepstral coefficients (MFCC). These features are then fed into the neural network. Using the data at our disposition, we train the network and test it with new data to compare the performance with the classical LSTM default variant.

Several previous studies use the same dataset, so the results are compared to the same criteria to evaluate the performance of this variant for our purpose. The features and parameters are optimized with both variants to obtain a global view of the Attention mechanism for early diagnosis in newborns and conclude if this path can be taken or not.

Keywords: diagnosis in newborns, pathology classification, newborn cries, recurrent neural network, Transformer, Attention enhanced LSTM, Long-Short Term Memory, Multilayer Perceptron, Mel-frequency Cepstral Coefficient

Mécanisme d'Attention dans le réseau de neurones pour le diagnostic précoce chez les nouveau-nés en utilisant les signaux de cri

Anisan GUNARATHINAM

RÉSUMÉ

Les humains utilisent leur voix pour communiquer, et elle est souvent motivée par l'instinct. À plus grande échelle, ce même instinct peut s'appliquer à un nouveau-né qui tente de s'exprimer, avec évidemment, son cri. Selon des études antérieures réalisés sur ce sujet, il existe une forte corrélation entre le cri d'un nourrisson et l'état psychologique et la pathologie qui affecte ce dernier. Cette étude se concentre sur la recherche d'une corrélation et l'identification de l'état du nouveau-né à l'aide d'un réseau de neurones. Cet outil aiderait les experts à identifier quelque chose qu'ils auraient pu potentiellement ignorer et à mieux diagnostiquer un nouveau-né en fonction de la maladie qui l'affecte et ce, avant qu'il ne soit trop tard. Que ce soit dans un pays développé ou un pays en développement, cette solution ne nécessite aucun matériel de diagnostic coûteux.

Dans cette étude, nous utilisons le mécanisme d'Attention dans un réseau neuronal qui a connu un succès au cours des deux dernières années dans la reconnaissance vocale et la traduction de texte. La couche d'attention permet au réseau de se concentrer sur différents aspects de l'entrée. Le Transformer utilise l'architecture Encodeur-Decodeur pour résumer l'intégralité de l'entrée avant de sortir les résultats, qui sont fortement liés à chaque séquence d'entrée. L'idée est d'éliminer la dépendance de l'entrée de longueur fixe menée dans les études précédentes. Le mécanisme d'attention amélioré LSTM (Long short-term memory) est un réseau neuronal récurrent qui hérite la couche du Self-Attention du Transformer.

Pour chaque échantillon de cri, seules les sessions d'expiration sont extraites pour générer la matrice des coefficients cepstraux à fréquence Mel (MFCC). Ces paramètres sont ensuite alimentés dans le réseau neuronal. En utilisant les données à notre disposition, nous formons le réseau et le testons avec de nouvelles données pour comparer les performances avec la variante classique par défaut du LSTM.

Plusieurs études précédentes utilisent le même ensemble de données, donc les résultats sont comparés aux mêmes critères pour évaluer les performances de cette variante. Les fonctionnalités et les paramètres sont optimisés avec les deux variantes pour obtenir une vue globale du mécanisme Attention pour le diagnostic précoce chez le nouveau-né et conclure si cette voie peut être empruntée ou non.

Mots-clés : diagnostic chez le nouveau-né, classification de pathologie, cris de nouveau-né, réseau neuronal récurrent, LSTM avec attention, Long-Short Term Memory, Perceptron Multicouche, coefficient cépstral de fréquence de Mel

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS, INITIALS AND ACRONYMS

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BFCC	Bark Frequency Cepstral Coefficients
BPSO	Binary Particle Swarm Optimization
BPTT	Backpropagation Through Time
CEC	Constant Error Carrousels
CFS	Correlation-Based Feature Selection
CIA	Communication Interauriculaire
CIV	Communication Interventriculaire
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
CSV	Comma-Separated Values
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Dense Neural Network
DPS	Discriminant Power Score
DSP	Digital Signal Processor
DTW	Dynamic Time Warping
ELMO	Embeddings from Language Model
ETS	École de Technologie Supérieure
FFNN	Feed Forward Neural Network
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Models

GPT	Generative Pre-trained Transformer
GPU	Graphical Processing Units
GRNN	General Regression Neural Network
GRU	Gated Recurrent Unit
GSFM	Genetic Selection of a Fuzzy Model
HTAP	Hypertension de l'artère pulmonaire
IFSM	Iterative Forward Selection Method
KNN	K-Nearest Neighbor
LPC	Linear Prediction Coefficients
LPCC	Linear Predictive Cepstral Coefficients
LSTM	Long-Short Term Memory
LU	Logistic Regression
MCLNN	Masked Conditional Neural Network
MFCC	Mel-Frequency Cepstral Coefficients
MLP	Multi-Layer Perceptron
mLSTM	Multiplicative Long-Short Term Memory
NLP	Natural Language Processing
OMLSA	Optimally Modified Log-Spectral Amplitude Estimator
PCA	Principal Component Analysis
PNN	Probabilistic Neural Network
RBFN	Radial Basis Function Network
RDS	Respiratory Distress Syndrome
RNN	Recurrent Neural Network
SED	Sound Event Detection

STFT	Short Time Fourier Transform
SVM	Support-Vector Machine
TDNN	Time-Delay Neural Network
WAV	Windows Wave
ZCR	Zero-Crossing Rate

LIST OF SYMBOLS AND UNITS OF MEASUREMENT

s	Seconds
ms	Milliseconds
Hz	Hertz
kHz	kilohertz

INTRODUCTION

Context of the Research

Every creature on this planet has a way of communicating with others. Some use a very complex language to deliver their thoughts and emotions like humans, while others use basic sounds that contains all the necessary information to make them understandable. While for some, those are hidden information, they can be interpreted as a sentence with a whole meaning for others. It is only a matter of time and recognition before a pattern can be found within this information and be used to understand them. Baby cries are an excellent example of this statement. According to recent studies (Bğnicğ, Cucu, Buzo, Burileanu, & Burileanu, 2016), we believe there is a strong correlation between an infant's cry and its reason. Although this research does not focus on the infant's needs (hunger, pain, discomfort, diaper wet, etc.), another classification can be beneficial at a larger scale, one that can prevent infant death worldwide.

According to the World Health Organisation, there were 2.4 million newborn deaths in 2019, and most of them occurred during their first 28 days. Figure 0.1 shows that the numbers are concentrated on the first few days of birth. In fact, 47% of infant deaths happen on the first day the child opens his eyes, while 75% happen after a week.

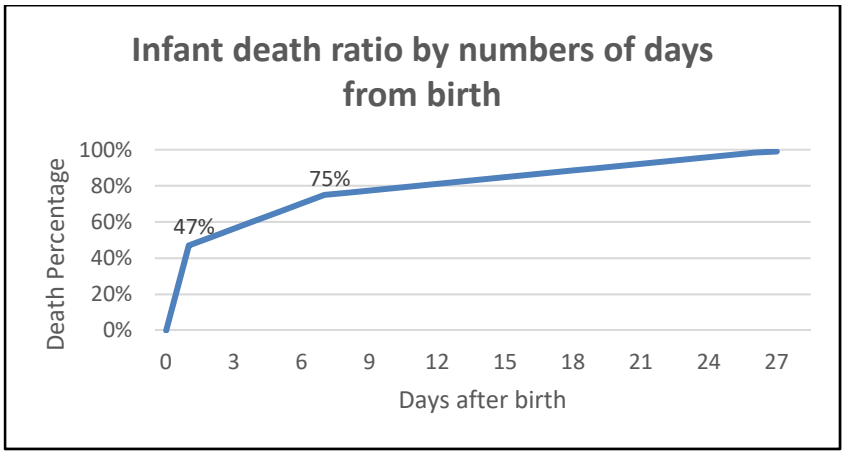


Figure 0.1 Newborn date ratio evolution by days

Among the reasons for death, failure to offer good healthcare during the first 28 days is the leading cause. A wrong diagnosis of the infant's psychological and physical state or an absence in detecting its pathology on time might lead to inadequate treatment and eventually cause its death. Also, the lack of expensive and accurate equipment is not helping the cause (World Health Organization, 2020). Figure 0.2 demonstrates that the highest infant mortality rate is located in developing countries such as India and countries in the African continent. These data can be obtained publicly through the World Health Organization website. As mentioned earlier, these countries do not have the materials to efficiently diagnose a disease affecting the child during its early days.

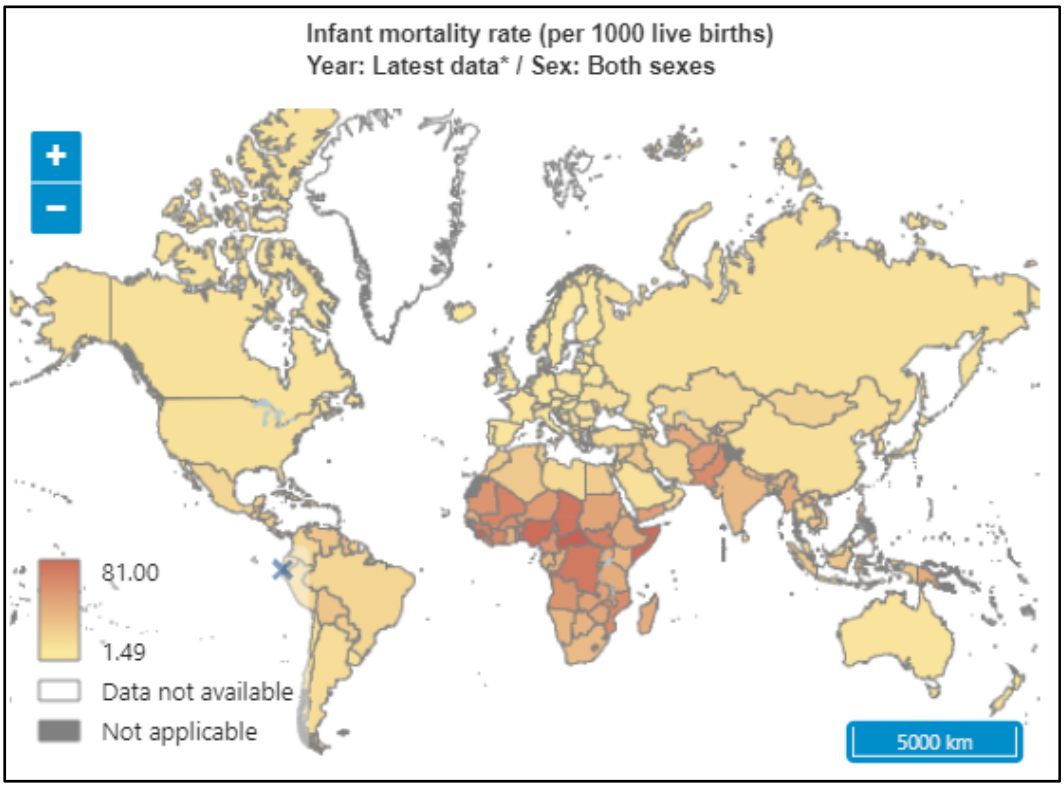


Figure 0.2 Infant mortality rate per 1000 newborns per country

Statement of the research problem

Since the vast majority of infant deaths occur in developing countries, it wouldn't make sense to implement a solution that requires expensive technology and equipment because these countries wouldn't afford them anyway. The idea behind this research is to come up with a product that can be affordable, accurate and does not require any special equipment to diagnose the newborn's health condition. We use something already present and produced by the baby naturally; its cry.

The infant's cry is its way of expressing itself when facing discomforts such as hunger or pain. There are also other types, such as pleasure cries. Moreover, the acoustic properties of the cries give away more information than anticipated (Zeifman & St James-Roberts, 2017). In fact, features such as the fundamental frequency and the duration can determine the type of pathology affecting the child. Studies show that certain conditions like Down's syndrome, brain damage and hyperbilirubinemia can be diagnosed using only features present in the infant's cry (Zeifman & St James-Roberts, 2017). Autism can also be detected using this method, according to recent studies.

Few questions are raised from the thought:

1. What about other common and uncommon pathologies?
2. Is there also a pattern in the acoustic properties that can be used for early diagnosis?
3. How accurately can we tell if a baby has a particular pathology or not?
4. Could the experts use this method to support their hypothesis?

In this work, we combine acoustic properties with a fast-evolving technology that is taking over the course of the development in the field of automation and computers, the artificial intelligence, also known as the neural network. We focus on two types, the Transformer network and the Attention enhanced LSTM (Long-Short Term Memory). Both had great success in the history, Transformer being the latest one using an Encoder-Decoder architecture.

Objective and methodology

The main objective of this research is to offer a binary classification among four diseases to detect them in the early stage of the infant's birth and hopefully treat them better for a higher probability of survivability. By limiting the research to four conditions only, we focus on comparing neural network performance in this specific field of study rather than detecting as many health conditions as possible. We want to observe if we can get a reasonable accuracy so that the system may be used worldwide with a stable confidence level. By comparing the network's performance, we might be able to determine if it could be worth taking this path for further studies or not.

The initial problem that is studied is: Can the state-of-art Transformer network, known to have great success in natural language processing (NLP), outperform the classical LSTM, which is a recurrent neural network (RNN)?

This project is not an entirely new study. In fact, several studies were performed on this case by other authors worldwide and from École de Technologie Supérieure (ETS). Among them, we have identified few studies that helped us refine the problematic:

1. Using acoustic features such as gliding fundamental frequency over the expiration and irregularity in the resonance frequency (Kheddache & Tadj, 2019).
2. Study of rhythm and melody pattern in infants with a pathological condition (Salehian Matikolaie & Tadj, 2020).
3. Qualitative study of the cry's spectrogram (Kheddache, 2014).
4. Implementation of the Gaussian Mixture Models (GMMs) (Farsaie Alaie, 2015).
5. Neural networks to classify the infant's condition, such as LSTM (Chang & Li, 2016), CNN (Lavner, Cohen, Ruinskiy, & Ijzerman, 2016), (Anders, Hlawitschka, & Fuchs, 2020), Masked Conditional Neural Network (MCLNN) (Medhat, Chesmore, & Robinson, 2020) and Transformer network (Vaswani et al., 2017).
6. Using statistical learning methods, like the SVM (Support-vector machine) (Salehian Matikolaie & Tadj, 2020).

To deliver this research, the following diagram in figure 0.3 describes the methodology that has been used. Data preprocessing is a big part of the research. There were already the necessary codes in Matlab and other software for features extraction from the audio files. Still, our focus was to make a system that is 100% dependant on the same programming language and processing architecture. This way, all the steps from A to Z could be done on the same equipment, like the Raspberry Pi, without an external resource. The main steps of this equipment would be:

1. Record the audio data for a few seconds.
2. Filter the unnecessary background noises and silences.
3. Extract the features that will be learned by the neural network.
4. According to the expiration sessions, classify the infant's pathology and refine the accuracy with more expirations coming in from the child.
5. Use the recorded data and an expert's point of view to train the neural network and reduce its errors.

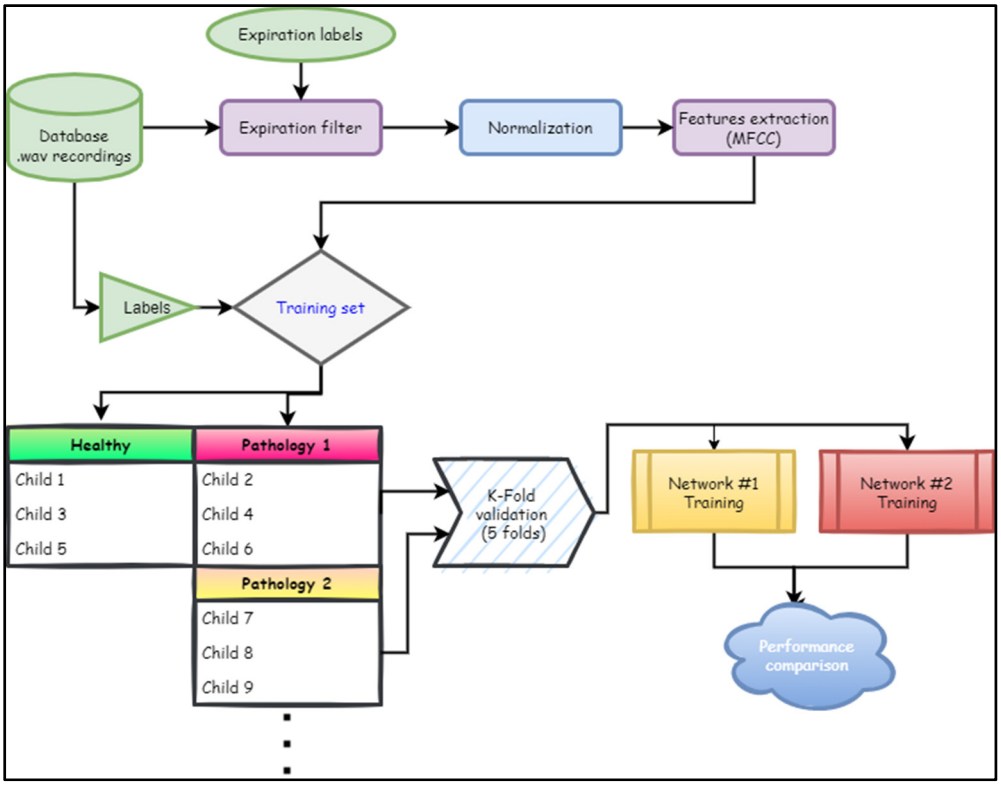


Figure 0.3 Flowchart of the methodology is this research

Organization of the thesis

The following thesis is organized in this manner:

Chapter 1 is a review of state of art in the field of signal processing and classification of audio sounds. We describe the contribution of other authors to this problematic and which problems that they have solved so far. Our goal is to make a path that has not been explored yet. With the help of the obtained results, we determine if this path can be followed for further development.

Chapter 2 is a detailed description of the essential theories to understand the foundations of this research. The chapter explains the features that are extracted, a summary of the algorithms used for data extraction, the mathematics behind the studied neural networks and the parameters of the network training process.

Chapter 3 contains all the information on the experimentation that was done in this thesis. The database details and the chosen data for this study are mentioned, as well as the pre-treatment work done in order to feed the extracted features to the classifier model. Also, the methods used to describe the comparison of the networks are marked out.

Chapter 4 presents the final results obtained throughout this research. The neural networks are compared based on their performance, and furthermore, the results are discussed in detail. The hypothesis of the thesis is, therefore, be affirmed or refuted.

The final part is the conclusion and the recommendations for future research that might interest other authors in pursuing this path.

CHAPITRE 1

PATHOLOGY CLASSIFICATION BASED ON INFANT CRY

Analysis and pattern detection on specific pathology based on the infant's cry is not a recent field of interest. In fact, studies starting from the 1970s observed the spectrum effects of infant cries having a particular disease. For example, Lind, Vuorenkoski, Rosberg, Partanen, and Wasz-Höckert (1970) analyzed the spectrographic of 30 infants cry with Down Syndrome and 120 normal infants to conclude that infants with the disease present higher stuttering, a longer cry duration, a lower pitch, and a lower pitch flat melody.

This method can be helpful, but as more diseases are identified, some of the main features are also observed on several different pathologies. For example, longer cry duration can be associated with numerous pathologies, making it impossible to classify the cries efficiently. Instead of using manual classification from experts based on these patterns, researchers started using different classifiers with a multitude of features on datasets since the year 2000.

Training experts to recognize different pathologies is time-consuming and resource-consuming compared to what machines can do nowadays. Mukhopadhyay et al. (2013) conducted a study where they trained a group of people to recognize cries in a database. The accuracy of the human ear is only 33.09% compared to 80.56% obtained from a machine on the same database.

In this chapter, we discuss the previous research that has been conducted in the field of infant's cry pathology classification using classifiers and their contribution.

1.1 Steps of infant cry classification

In every single research on this topic and several others, we find five main steps for effectively classifying an infant cry. Each of those steps is crucial and highly impacts the final performance of the classifier. Presented in this section are the data acquisition, the pre-processing, the features extraction, the feature selection and the classification.

1.1.1 Data acquisition

As required by any classification or regression system, the first step is to acquire valuable data to train the machine. In our case, the data is the recordings of the infant's cry gathered by medical staff. Furthermore, an expert like a nurse or a doctor would have to label the data by filling at least the “reason of cry” or “known pathology” field for each recording. In the case of unsupervised learning, labels are not required.

1.1.1.1 Challenges of data acquisition

The main challenge in our study is that there is a minimal number of samples and databases available to the public, and it is not easy to acquire them from other researchers (Jeyaraman et al., 2018). Classifiers require an extensive database to be able to be trained efficiently. Healthy infant samples are the most common, while infant cries with specific pathology are very rare to find. Also, the medical information of an infant is sensitive and cannot be obtained easily. There are several procedures to be done by the medical team and the parents.

Another challenge is the environment in which the data is recorded. The hospital has many random noises and interferences, which, added to the cry sounds, confuse the classifier, and falsify the prediction. Ferretti, Severini, Principi, Cenci, and Squartini (2018) experiment concluded that synthetic samples produce better accuracy than real-time samples using a convolutional neural network (CNN). Synthetic data means purposely adding random noises on top of clean recordings to simulate a noisy environment.

1.1.1.2 Databases

Databases can be found online or from authors studying this case. Manikanta, Soman, and Manikandan (2019) used an online database in their study of different classifiers like the multi-class support vector machines (MC-SVM) and the one-dimensional convolutional neural network (1D-CNN). They obtained a performance of 98% and above. However, the online

database is limited in samples, and they are not always well labelled and accurate (Chunyan Ji, Mudiyanselage, Gao, & Pan, 2021).

One famous database is the Baby Chillanto database (Reyes-Galaviz, Tirado, & Reyes-Garcia, 2004), from the National Institute of Astrophysics and Optical Electronics in Mexico. However, there are only five types of cry signal in their database: asphyxia, deaf, hungry, normal and pain. One-second segments are automatically extracted in their database.

Chunyan Ji et al. (2021) reviewed many databases used by authors working on this issue.

1.1.1.3 Data augmentation

In our field mainly, the number of samples is minimal. Few reasons include that it requires time and dedication to record the cries and label them correctly. One technique is data augmentation, which consists of increasing the data size by adding more samples that are almost copies of existing samples. Using this method, slight variations of a sample with the same class are produced, causing the model to be more robust to noise.

Felipe et al. (2019) used several techniques to increase the number of samples. They used features from spectrograms and features from the audio signal directly to train their SVM classifier that determines if an infant suffers from pain or not. They modified some factors like noise and tonality variation for each feature category to create a more extensive dataset. However, these methods did not increase the accuracy.

1.1.2 Pre-processing

The second step is the pre-processing of the data. This part is essential to optimize the performance of the classifier. In the previous section, we discussed the noisy environment where the cries are recorded. Once the data has been gathered, it is impossible to go back and reduce the amount of noise. There are a few techniques to clean the data from unuseful sounds and feed the machine only with the most valuable data for classification.

1.1.2.1 Noise filtering

Filtering the signal with a high-pass filter helps to remove the low frequencies noise in the recording since an infant's cry starts at around 250 Hz – 300 Hz (Jam & Sadjedi, 2009). The fundamental frequency ranges from 250 Hz to 600 Hz, while the resonant frequencies are typically around 1100 Hz and 3300 Hz (Lavner et al., 2016).

With a sampling frequency of 44100Hz, up to 22050 Hz frequencies can be detected, but most of the range does not contain valuable data since there is no information beyond 4000 Hz or even 6000 Hz. Authors often use 4kHz as their maximum frequency for features extraction.

Ferretti et al. (2018) used a system of 8 circular microphones to reduce the noise using a filter-and-sum beamformer and focus on the source of the noise. To reduce the residual diffuse noise in the pre-treatment phase, they used an Optimally Modified Log-Spectral Amplitude Estimator (OMLSA) as a filter.

1.1.2.2 Noise and silence removal

More profound treatment is to remove the silence and noise frames from the sound. Abou-Abbas, Alaei, and Tadj (2015) separated the segments of the recordings and manually labelled them. Segments containing noises, silences or a mix of cry and noise are identified correctly. Only the sections that are labelled as "expirations" and "inspirations" are considered in future research (Salehian Matikolaie & Tadj, 2020). However, this process is very time-consuming as it either requires a sophisticated system or manually separating and labelling the segments.

Affendi and Yusoff (2019) worked on an anomalous sound event detection (SED) system that can trigger an action when the convolutional recurrent neural network (CRNN) with LSTM detects a specific sound. They obtained a performance of 93.1%. This neural network can eliminate the silences and other noise during the recording phase directly. However, this system has not been tested on expirations that only last a few seconds and occur several times during a crying session.

1.1.3 Features extraction

Features extraction allows the collection of the discriminative features in the recordings. These features are then fed into the classifier for decision-making. Directly feeding the samples of the recordings results in a deplorable network as the samples alone contain no discriminative information that the machine can focus on. There are two main categories of audio features: the time domain and the frequency domain (Chunyan Ji et al., 2021).

1.1.3.1 Time-domain features

Features extracted from the time domain samples are easier to extract since few calculations are needed to obtain them. A few examples include zero-crossing rate (ZCR), amplitude-based features, energy-based features and rhythm-based features. While these features can offer acceptable performance in the field, they are not discriminative enough to accurately classify the sound when recorded in a natural environment containing noises (Chunyan Ji et al., 2021). It is the reason they are rarely used alone.

1.1.3.2 Frequency domain features

The frequency-domain gathers the most valuable information in a sound signal. Researchers use this domain along with the variation of the frequencies over time to construct a whole new set of features. The most common is the Mel Frequency Cepstral Coefficients (MFCC), like the one Liu, Li, and Kuo (2018) used to classify the reasons for an infant's cry. They tested the linear predictive cepstral coefficients (LPCC) and the Bark Frequency Cepstral Coefficients (BFCC) to compare the performance. They concluded that BFCC offers a higher performance.

Although the MFCC features yield the best overall performance, combining other frequency domain features to increase the results is possible. Salehian Matikolaie and Tadj (2020) combined MFCC, tilt and rhythm features to classify infant pathology in expiration and inspiration sessions. They obtained an increase of more than 3% in terms of accuracy.

Kheddache and Tadj (2019) used several features in the frequency domain to classify infants' cries by pathology. Their main idea captures the rapid increase or decrease of the fundamental frequency. Harmonics of the fundamental frequency and the resonant frequencies, as well as the dysregulation, are used. Compared to MFCC alone, with an accuracy of 60% on full-term infants, they obtained 67% with all these features combined for binary classification. Multi classes (five) obtains an accuracy of 82% compared to 72% with MFCC alone.

1.1.3.3 Possible audio features

Many other features can be used for classification. Figure 1.1 lists feature based on their category, obtained from (Chunyan Ji et al., 2021).

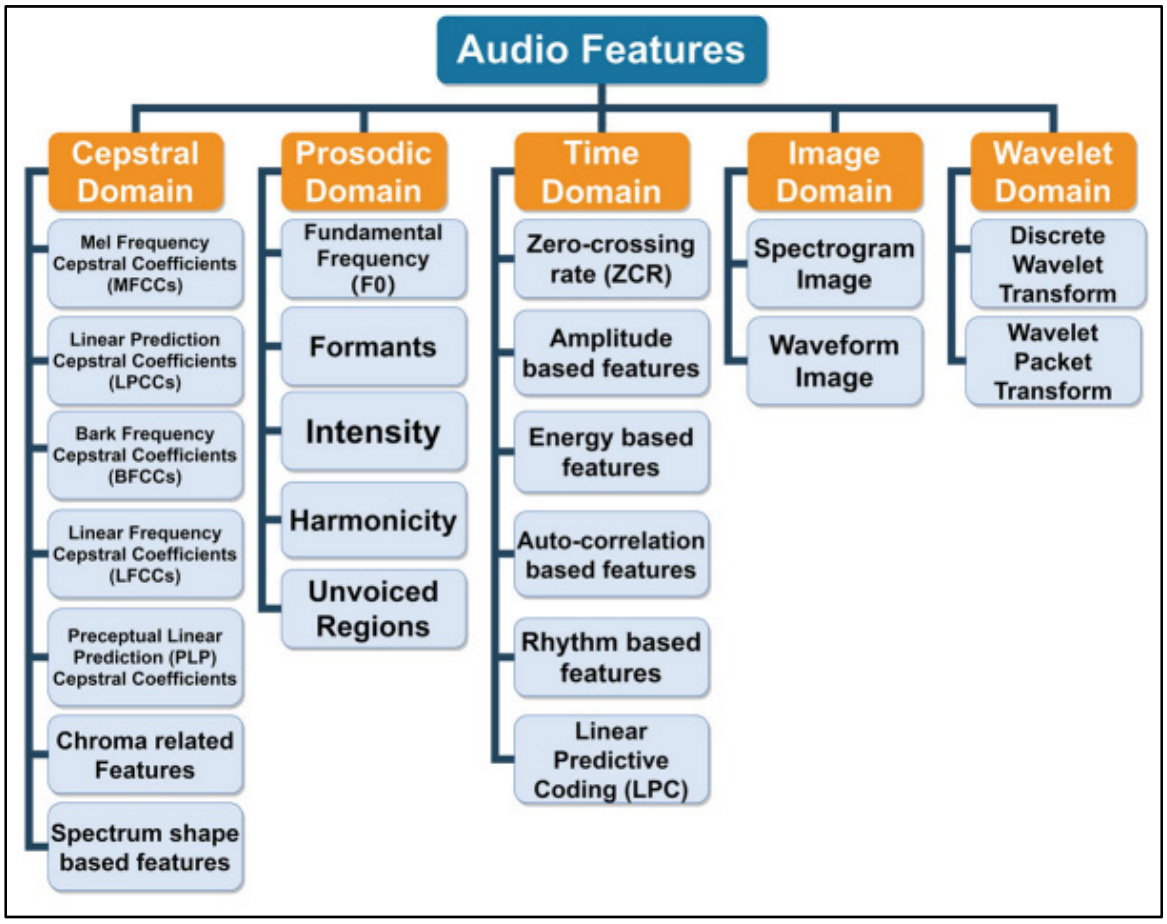


Figure 1.1 Example of audio features that may be used for classification
Taken from (Chunyan Ji et al., 2021)

1.1.4 Feature selection

This step is ignored sometimes, but it is precious as it can significantly reduce the training time and use less processing power (Jeyaraman et al., 2018). According to this author, the Principal Component Analysis (PCA) technique provides an improved performance. It reduces the number of features that are fed to the classifier by using an algorithm. However, PCA calculation requires computational power and time to calculate the most important features that are fed to the classifier network. The main goal of this phase is to reduce dimensionality by finding the discriminants and a correlation in the features.

Yoshifumi, Kentarou, and Tomomasa (2011) proposed an iterative forward selection method (IFSM) inspired by the cross-validation method. By ranking all the discriminants using the discriminant power score (DPS), they concluded that using high ranking scores offers a better performance from the machine learning system. Also, IFSM offers a relatively big difference in accuracy compared to its default non-iterative variant.

Zabidi, Mansor, Lee, Yassin, and Sahak (2011) used a multilayer perceptron (MLP) to classify infant's cries between healthy and infants with asphyxia. They observed that a Binary Particle Swarm Optimization (BPSO) technique, which selects the most valuable MFCC for the artificial neural network, successfully increased the accuracy of the classifier.

Wahid, Saad, and M. (2016) tested five different feature selection methods to reduce the number of inputs on their extracted MFCC and LPCC for infant cry classification. They have obtained their best performance with an accuracy of 93.43% using the Correlation-Based Feature Selection (CFS) and the Radial Basis Function Network (RBFN). These techniques were used on MFCC, delta MFCC and delta-delta MFCC.

As we can observe in the research mentioned above and several others, feature selection filters the irrelevant information from the extracted features and reduces the computational time and power and increases the machine’s overall performance.

1.1.5 Classification

After the most crucial features have been extracted, the last step is to send them into the classifier. If the previous steps can potentially increase the machine's performance, this step doubles the weight since it is crucial to choose one classifier adapted to the type of data we are working with. There are two main types of classifiers used in machine learning: traditional machine learning classifiers and neural network models.

1.1.5.1 Tradition machine learning classifiers

Support Vector Machine (SVM) is one of the most famous learning algorithms for data classification and regression analysis. It learns a possible boundary using a kernel equation from the received data in classification. SVM can support binary classification or multiclass (Badreldine, Elbeheiry, Haroon, ElShehaby, & Marzook, 2018). This classifier works well with high dimensionality data and does not require a massive amount of processing power. Chang, Hsiao, and Chen (2015) extracted four features out of fifteen and classified three kinds of cries using an incremental SVM. This technique consists of adding more and more data with each step to the training network. They obtained a remarkable increase of 18% accuracy compared to the original SVM.

K-Nearest Neighbor (KNN) is another method consisting of associating each sample to the closest neighbour class. The neighbour is updated as more data comes in. The value of k determines the number of neighbours from which to choose (Chunyan Ji et al., 2021).

Gaussian Mixture Model (GMM) is composed of several Gaussian distributions, also known as a normal distribution, representing each cluster (Farsaie Alaie, 2015). Each sample in the dataset belongs to a distribution. The mean vector and the variance matrix for each distribution are learned using the Expectation-Maximization (EM) technique. Sharma, Gupta, and Gupta (2019) demonstrated that GMM produced the best results compared to K-means clustering and hierarchical clustering. The overlapping is also kept minimal, even though the GMM model does not offer the optimal performance with a limited amount of data.

A fuzzy classifier is a similar concept which regroups fuzzy data for classification. Fuzzy data must be calculated first using probabilistic samples using the extracted features. Rosales-Pérez et al. (2015) used the Genetic Selection of a Fuzzy Model (GSFM) algorithm on MFCC to identify pathological cry. They have concluded that their optimized GSFM significantly improves the accuracy.

The classifiers can work well on supervised and unsupervised networks. Supervised training requires labels on the data, while unsupervised training learns to regroup similar data together and create classes depending on the similarity of the incoming inputs. The clustering methods such as K-means, GMM and fuzzy classifier are mainly used during unsupervised training.

1.1.5.2 Neural network models

Feed Forward Neural Network (FFNN) used alone serves mainly as a reference model in nowadays applications (Dewi, Prasasti, & Irawan, 2019). MLP model is a type of FFNN, and it usually scores less than state-of-art neural networks in almost every field. Variants were used by Saraswathy, Hariharan, Yaacob, and Khairunizam (2012) like the Probabilistic Neural Network (PNN), General Regression Neural Network (GRNN) and the Time-Delay Neural Network (TDNN). They obtained an accuracy of above 97% to classify asphyxia pathology. However, the FFNN had a second chance with the state-of-art Transformer model, the one we are studying in this research. It is placed above the Attention layer and contains most of the network's parameters.

Convolutional Neural Network (CNN) network is mainly used in computer vision for images analysis. Since images are 2-dimensional, this network utilizes matrix convolutions followed by a pooling layer to reduce the size of the data. Images or sequential images constituting a video can quickly grow large in space and computational requirements. Manikanta et al. (2019) succeeded in using a one-dimensional CNN (1D-CNN) to detect a baby cry sound in an indoor environment, including air-conditioners, fan, speech, and music sounds. Their network performed better than the SVM and FFNN networks.

Recurrent Neural Network (RNN) is best used with sequential data since the internal states of the cells preserve the information in the previous inputs of the sequence. They are primarily used for time series applications such as market prediction, speech recognition and natural language processing (NLP). Maghfira, Basaruddin, and Krisnadhi (2020) used a combination of CNN and RNN on the Dunstan Baby Language dataset. The CNN learns discriminative features in the spectrogram obtained using a Short Time Fourier Transform (STFT), which is then flattened to be fed in an RNN to gather the temporal information in the extracted values. Figure 1.2, which is obtained from their work, presents the model that they used in their work. They have obtained an accuracy of 94.97% using binary cross-entropy for two classes.

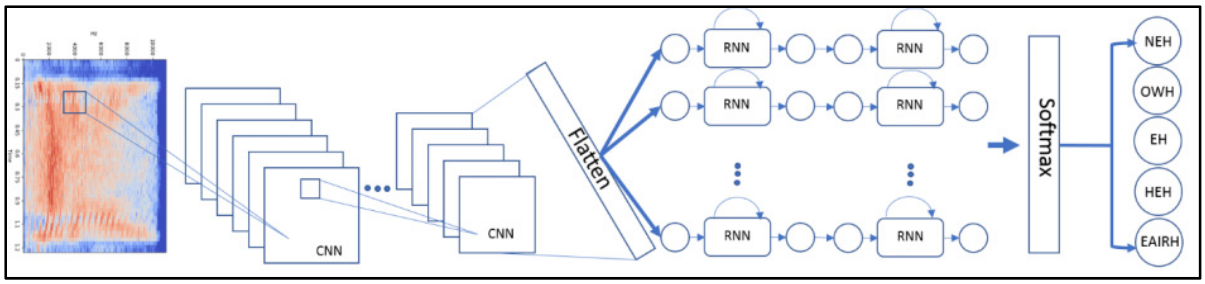


Figure 1.2 CNN-RNN model
Taken from Nadia Maghfira et al. (2020)

Fuzzy Neural Network is another type using fuzzy logic. Molaeezadeh, Salarian, and Moradi (2012) used a type-2 fuzzy pattern matching for classifying hunger and pain cries from healthy full-term infants. Using MFCC as features, they obtained a higher accuracy than SVM and Logistic Regression (LR) classifiers.

Overall, models using neural network obtains better performance than traditional machine learning classifiers, as shown by many types of research in this section compared to the SVM classifier. The latter compensates for its weak performance by its simplicity and the required amount of processing power. SVM, K-means clustering and GMM machine classifiers offer a lower training time, search time and classification time (Chunyan Ji et al., 2021). If the hardware allows and the number of training data is adequate, choosing a neural network as the classifier is the best option while keeping the SVM a reference for comparing the performance.

1.2 Identity of the infants

Several factors influence the results on pathological classification, such as the age of the infant, the gestational age, race, body mass index, etc. The performance can significantly vary if the chosen population is not carefully filtered first.

1.2.1 Age and gestational age

Jeyaraman et al. (2018), Affendi and Yusoff (2019), Kheddache and Tadj (2019), Salehian Matikolaie and Tadj (2020) mention that pathological detection in infant cries works best with infants under 2 months old. From this age, the vocal cords evolve and alter the main features used to classify. In our research, we focus on the same objectives. However, some research does not limit to very young babies. Anders et al. (2020) studied infants as old as 9 months. They use a CNN network to classify the type of sound produced by the infant, such as crying, fussing, babbling, laughing and vegetative vocalizations.

Salehian Matikolaie and Tadj (2020) use the data of only children with a gestational age of over 37.2 weeks. Their work uses the same database as we are using in this research. Preterm infants do not have the same traits in the recordings as full terms. Mixing the two types can create confusion for the neural network classifier as it tries to find a pattern and similarities with cries belonging to the same pathology.

1.2.2 Gender

The gender of the infant has a recognizable impact on the characteristics of the data. In fact, the fundamental frequency of a male and a female is different even though they have the same pathology (Jeyaraman et al., 2018). The range of the frequencies in which the cry is situated varies greatly. Furthermore, the melody of the sound, characterized by the frequencies composing the studied sound, are features that modify the nature of the input data. The author's responsibility is to gather an equal amount of data from both genders for all the pathologies to avoid classification problems.

1.2.3 Body mass index

The body mass index does not have as much impact as gender or gestational age. However, the weight and height of the infant modify the structure of the vocal tracts producing the cry (Jeyaraman et al., 2018). To this date, there is no research focusing on evaluating the performance of a higher and lower body mass index infant's pathology or cause classification. We believe it is due to the requirements being too strict on the few available databases for this problematic.

1.2.4 Other factors

According to Jeyaraman et al. (2018) review, other factors' influence on the classification problem is negligible, such as the birth of the child and its race. The difference is not significant enough to modify the accuracy of the machine classification.

There are hardware differences that might alter the results, such as the microphone recording quality and the amount of environmental noise, as discussed in section 1.1.2. The sampling rate and the resolution are other factors to be considered during the data-gathering phase.

1.3 Applications of infant cry classifications

Our research focuses on pathological classification, but there are other applications for which the cry of the infant can be used. In this section, three main applications are described: pathology classification, cry reason classification and cry sound detection.

1.3.1 Pathological classification

Among several pathologies studied in infants such as respiratory distress syndrome, hyperbilirubinemia, sepsis, down syndrome and hypothyroidism, asphyxia is a famous one among the authors. The Baby Chillanto dataset is mainly used for this classification in recent years. Here are some of the research done using different other databases:

- Rosales-Pérez et al. (2015) studied the asphyxia pathology using a fuzzy model classifier.
- Farsaie Alaie, Abou-Abbas, and Tadj (2016) studied several health conditions such as heart problems, neurological disorders, respiratory diseases, blood abnormalities, etc.
- Moharir, Sachin, Nagaraj, Samiksha, and Rao (2017) used GoogleNet and AlexNet classifiers and obtained a performance of 94%.
- C. Ji, Xiao, Basodi, and Pan (2019) managed to obtain an accuracy of 96.74% on the Baby Chillanto dataset using a Feed-Forward Neural Network (FFNN).

1.3.2 Cry reason classification

Many research focuses on the reason for the cry since there is a higher number of healthier children's data. The challenge, however, is the lack of certitude in the labelling. In a hospital environment, nurses and experts can discover the pathology of the infant with the proper diagnostic at any time and even modify the labelling later on in the dataset. It is not usually the case for the cry reason since there is a place for many interpretations. There is no valid method other than experimental to determine the reason why the baby was crying, hence biasing the initial database itself. The main challenge is that there is a lack of standard public datasets, and the accuracy is still low nowadays (Chunyan Ji et al., 2021). Some of the research in the past decades are:

- Yamamoto, Yoshitomi, Tabuse, Kushida, and Asada (2013) used a self-recorded database to classify discomfort, hunger and sleep. They used a 32-dimensional FFT and reduced the dimensionality using PCA. Using the nearest neighbour algorithm, an accuracy of 62.1% was obtained.
- Bano and RaviKumar (2015) worked on a self-recorded database using the K-nearest neighbour (KNN) algorithm to classify discomfort, sleep, burp need, discomfort and hungry cries. Considering the self-recorded database, they obtained 86% accuracy.
- Chang and Li (2016) classified hunger, pain and sleep in the National Taiwan University database using the CNN model to obtain an accuracy of 78.5%.

- Liu et al. (2018) categorized drawing attention cry, diaper change cry and hungry cry. They used the nearest neighbour approach and neural network. Their best accuracy using several features such as MFCC and LPCC is 76.4%.
- Felipe et al. (2019) used the iCOPEaudio database to predict if an infant is experiencing pain or not (binary classification). They used the spectrogram as features and the SVM classifier to obtain an accuracy of 71.68%.
- Maghfira et al. (2020) worked on the Dunstan Baby Database using the spectrogram of the sound and CNN-RNN network to classify pain, hunger, discomfort, burp need and belly pain. They obtained a performance of 94.97%.

On average, we observe that the accuracy for cry reason classification is lower compared to pathology classification.

1.3.3 Cry detection

Cry detection is the most straightforward application among the ones mentioned in this research. It is a binary classification aiming to detect an infant's cry sound in an environment containing several other noises. This type of application can be closer to a product used by parents to monitor their baby's activity instead of doing a 24h surveillance. In general, the researchers obtain very high performance for cry detection in the following papers:

- Lavner et al. (2016) demonstrated that using a CNN classifier yields higher performance than a low-complexity logistic regression classifier in detecting baby cry among domestics sounds such as parents talking and door opening. For babies ranging from 0 to 6 months, they obtained an accuracy of 95%.
- Ferretti et al. (2018) compare real and synthetic data for detecting infant cry in a noisy environment. Several noises such as hospital machines “beeping” and speech were tested. Synthetic data obtained a performance of 92.92% with a CNN classifier.
- Gu, Shen, and Xu (2018) used a different approach to classify infant cry sound. They extract linear prediction coefficient (LPC) directly from a digital signal processor (DSP) in

real-time. By applying the dynamic time warping (DTW) algorithm, they obtained an accuracy as high as 97.1%.

- Manikanta et al. (2019) classified self-recorded audio in homes to work on samples containing a mix of sounds during the infant`s crying phase, such as the fan or air conditioner. An accuracy of 86% was obtained.
- Affendi and Yusoff (2019) used CNN, RNN and LSTM networks to detect sound anomalies. They did not target infant cry specifically, but their system can be adapted to learn a sound type among others with an accuracy of around 93%.

Even if the complexity of this application is lower, the accuracy must compensate. If taking the example of a real-world application, it is crucial that the system must detect the infant's cry and alert the parents as soon as possible. If the device fails to do so many times, its trust will be lost, and therefore it has no real utility. However, in a domestic application, even if the system doesn't detect a cry on the first expiration session, chances are it will recognize with two or three expiration samples.

In summary, to efficiently classify infant pathology, the five main steps are:

1. Data acquisition: Gather the raw data from the source.
2. Pre-processing: Filter out the non-essential information.
3. Features extraction: Extract the most valuable deterministic factors in the data.
4. Feature selection: Reduce dimensionality and choose the best features that are fed into the classifier.
5. Classification: Use a traditional classifier such as the SVM or a neural network to train and classify the different cries successfully.

CHAPITRE 2

DEFINITION OF THE CONCEPTS

In this chapter, we define all the necessary concepts and theories that are essential in understanding the process of neural network classification. Starting from the pre-treatment, we characterize the features, Mels-frequency cepstrum (MFCC), followed by the neural network with all its facades, including the training process and hyperparameters optimization.

2.1 Pre-treatment on the database

2.1.1 Stereo to Mono

There is a high probability that a recorded file is encoded as stereo, meaning that the sound is optimized to be heard on the left and right speakers or headphones. The idea behind this is to add a more natural experience and spatial localization of the sound to the user. However, for signal processing, we do not need that duplicated information. It only occupies more space on the storage device without bringing new information.

The stereo to mono conversion process can be done either on the hardware side or the software side. On the hardware side, it implies regrouping both channels using a resistor. It is, however, easier on the software side as it does not require physical components. The following formula succeeds in converting to mono using the average of both channels:

$$Mono(i) = \frac{Left(i) + Right(i)}{2} \quad for\ i = 1, 2, 3, \dots N \tag{2.1}$$

In this equation, N corresponds to the number of samples in the recording, Left and Right are the left and right channel data, respectively.

2.1.2 Audio normalization

Sometimes, a recording's global energy might be very high, for example, if the microphone is too close to the sound source. The opposite effect can also be observed. The idea behind normalization is to adjust the sound level of the entire recording so that the energy of all recordings is similar. As adaptable as the neural network might be, helping it with normalization can improve its performance (Geller, 2019).

In this research, we used the method called standardization, in which the data is centred around the mean value with a unit standard deviation. It follows this equation:

$$X' = \frac{X - \mu}{\sigma} \tag{2.2}$$

μ is the mean of the samples in the recording and σ is the standard deviation.

The second method, which we call normalization, uses this equation:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.3}$$

X_{max} and X_{min} are the maximum and minimum values of the samples, respectively.

While the normalization rearranges the values between 0 and 1, standardization shifts the values between the standard deviation of the data while maintaining a mean value around 0. In standardization, the values can be negative, just like the audio samples. There is no clear winner between those two methods for audio classification (Bhandari, 2020), even though standardization is said to work better on data that follows a Gaussian distribution. Experimenting with these two methods can declare the winner for a specific task.

Figure 2.1 shows the effect of both these methods on a dataset that has an uneven energy level.

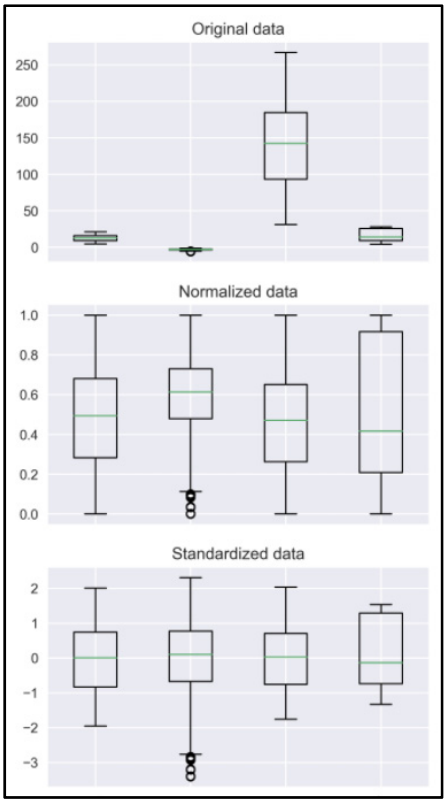


Figure 2.1 Normalization vs Standardization on four audio samples
Taken from Bhandari (2020)

2.2 Mel-frequency cepstrum coefficients (MFCC)

2.2.1 Coefficients

In the field of speech recognition, the Mel-frequency cepstrum coefficients (MFCC) are widely known to offer state-of-art performances when used as one of the features of a classification or translation work (Chauhan & Desai, 2014). From a higher number of data extracted in the frequency domain from the time domain, there is a possibility of reducing 512 samples to only 13 coefficients for a timeframe without losing critical information in the audio file. We accomplish this by focusing on the lower frequencies and extracting the human ear perceived spectrum from the frequency domain. The benefits are observed better when working with a classifier network, whose training time can be exponentially reduced when using coefficients instead of the entire spectrum as features.

2.2.2 Time domain, frequency domain and time-frequency domain

Time-domain: Raw audio signals are encoded in the time domain. There is a sample every few micro seconds depending on the sampling frequency rate, and a voltage represents the amplitude at every moment. There is not much information that can be observed in this domain. A little noise can greatly alternate the shape of the signal (Jeyaraman et al., 2018).

Frequency domain: With a Fast Fourier Transform (FFT), the signal can be converted in the frequency domain. It contains a lot more information, and a human can visualize it. On a frame of samples, we get to know the energy of each frequency present in the signal. It is also easier to detect anomalies in the signal and filter them efficiently.

However, since this domain is like a "screenshot" of a few samples (for example, 1024 samples), there is a loss of information from the neighbouring samples. On the other hand, if the FFT is performed on the whole signal, which may contain thousands of samples, we obtain every single frequency on the audio. This is not helping since we do not know when these frequencies appeared in the audio. There is no temporal localization of the sound. As an instance, a noise of 1 kHz might be present on the end of an audio file, which is not essential for the study, but we will find a magnitude of that frequency in the analysis.

Time-frequency domain: By combining both domains, we create a complementary domain that inherits both advantages and drops the disadvantages. This domain is the one nowadays researchers use to handle audio sounds and speech recognition (Affendi & Yusoff, 2019). The idea is to perform an FFT on slices of the audio signal and combine all the results in a 2-dimensional array, with or without overlapping the samples. The obtained array is called a spectrogram and can be interpreted as an image. This domain contains much more information about the signal and the varying fundamental frequencies throughout the time. A study uses this gliding effect as a feature for the neural network (Kheddache & Tadj, 2019).

2.2.3 Mel scale

Psychophysical studies show that human perception of different frequencies in the sound doesn't follow a linear scale like it should (O'Shaughnessy, 1987). In fact, it follows a subjective and experimental scale called the Mel scale (Morgan, 2002).

Human ears are more efficient at working and discriminating lower frequencies than higher ones. The Mel scale emphasizes the lower bands by giving more resolution to that area compared to high frequencies.

Several studies have been conducted to determine the exact proportions of the scale. Still, since it is subjective, only experimental constants were able to define the pitch that we actually hear. The perceived frequency can be obtained using the following equation:

$$f_{mel} = 2595\ log_{10}(1 + \frac{f}{700}) \tag{2.4}$$

f_{mel} is the perceived pitch frequency in Mels unit and f is the real frequency of the sound.

2.2.4 Filter banks

A series of triangular filter banks are applied on the Mel scale in order to focus on the most critical frequency band to the human ear. The result of the combination shows a vague approximation of the perceived spectrum. As mentioned above, a higher resolution in the lower frequency is needed.

The filter bank has an equidistant peak in frequencies below 1000 Hz, and each subsequent filter has its peak at 1.1 times the bandwidth of the previous filter (Sangeetha, Hariprasad, & Subhiksha, 2021). This factor gives an exponential variation of the focus in the higher frequencies compared to the lower frequencies, more precisely under 1 kHz. Figure 2.2 shows the exponential spacing between the peaks of the triangular bank filters from 0 Hz to 4000 Hz.

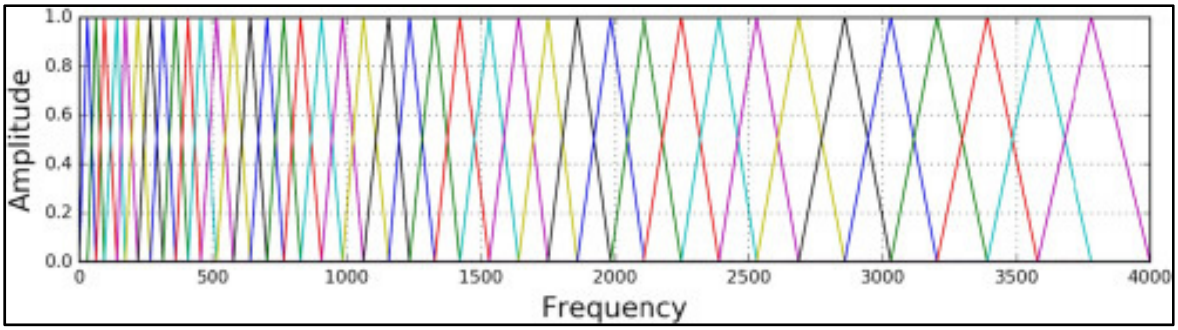


Figure 2.2 Triangular filter banks used with the Mel scale
Taken from Sangeetha et al. (2021, chapter 8)

2.2.5 MFCC calculation

The following steps, illustrated in figure 2.5, show the extraction process of the MFCC from an audio file containing voice or sound (Muda, Begam, & Elamvazuthi, 2010).

1. **Pre-emphasis:** This filter emphasizes high frequencies. Microphones are not efficient at high frequencies and create a low energy distorted signal. A pre-emphasis filter with the following equation can help soften this phenomenon:

$$y(n) = x(n) - 0.95\,x(n-1) \tag{2.5}$$

   y is the output while x is the discrete input. A 0.95 coefficient is used here.

2. **Framing:** Segmenting the audio samples into smaller batches. This step is essential to obtain the spectrogram of the audio signal. Standard frames range from 20 to 40 msec, with 256 samples depending on the sampling frequency (Muda et al., 2010).

3. **Windowing:** Adding a window function that is not rectangular smoothens the edges of the audio signal and adds more weight to the middle of the frame (Kelly & Gobl, 2011). Since we cut the audio in constant length frames in the previous step, we have discontinuities that significantly affect the results of a Fourier transform since the edges don't start and end with the value of 0. A typical window function used in the speech recognition area is the

Hamming window, which is obtained with the following equation (Muda et al., 2010) and represented in figure 2.3 right below:

$$w(n) = 0.54 - 0.46\ cos\left(\frac{2\pi n}{N-1}\right) \mid 0 \leq n \leq N-1 \tag{2.6}$$

w is the window function; n is the sample and N the total number of samples in each frame.

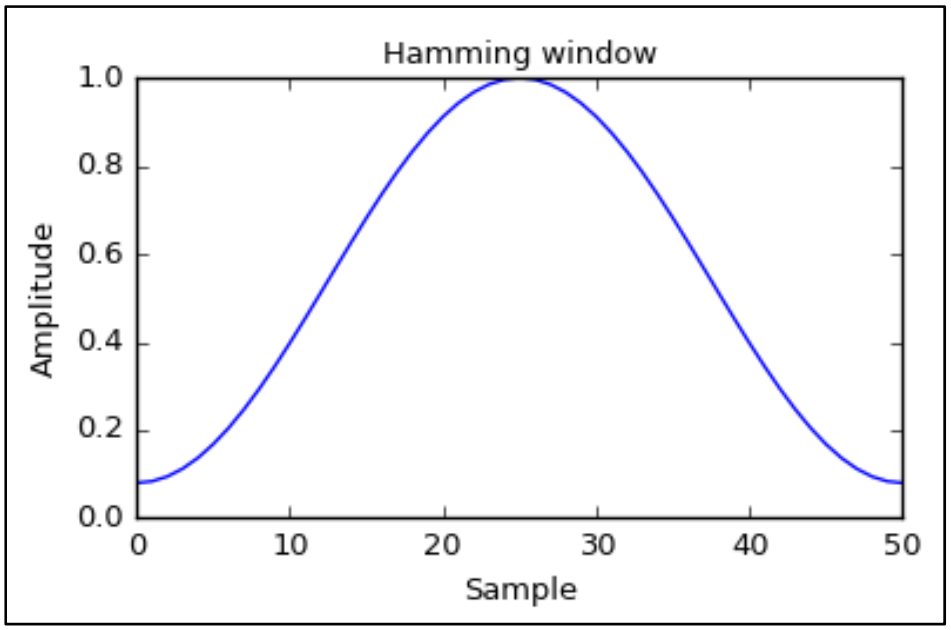


Figure 2.3 Function of a Hamming window (obtained with the Scipy library)

The hamming window function is then multiplied with every sample in the frames. We can observe that the middle sample keeps its full amplitude while the ones on the edges lose most of their energy. On the other side, the Hann window has a window where the edges amplitude value is 0. However, the hamming window generally yields better results in speech processing (Kelly & Gobl, 2011).

4. **Fast Fourier Transform:** The Fourier transform algorithm converts data from the time domain to data in the frequency domain. Since the data is digital, the Discrete Fourier Transform (DFT) is applied to do so using the following equation:

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-2j\pi kn}{N}} \tag{2.7}$$

Where x is the data array of time-domain samples, N is the total number of samples.

However, the computation power requirement for this formula is $O(N^2)$, which could result in a very long processing time, exponentially growing with the number of samples. A faster algorithm has been discovered that exploits the full potential of computer architecture, the Fast Fourier Transform (FFT). Using the FFT, the computation requirement is now reduced to $O(N \log_2 N)$ (Maklin, 2019).

To compare, it would take the DFT 31.2 years to compute the Fourier Transform with a sample size of $N=10^9$ with 1ns per operation, while it would only take 30 seconds using the FFT with the same criteria. Figure 2.4 illustrates the required time to compute the Fourier Transform with both algorithms.

One disadvantage of the FFT is that it requires a number of samples that is an exact power of 2. The exponent determines the number of steps in the Fourier algorithm.

One solution to respect the criteria is to cut the number of samples accordingly during the framing step (step #2). For example, instead of having 1000 samples per frame, we would have $2^{10} = 1024$ samples. Another solution is to pad the data with zeros, but this method removes some accuracy in the transformation as those are not the actual values.

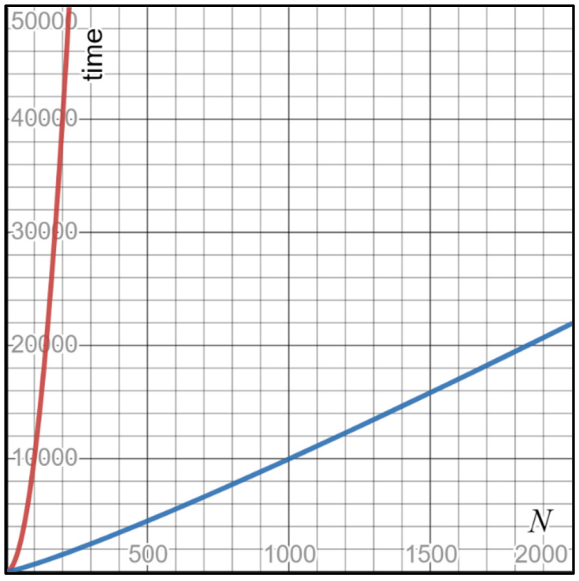


Figure 2.4 Estimated computational time for the DFT (in red) and the FFT (in blue) based on the number of samples N in the data

5. **Mel scale and filter bank:** Apply the triangular filter bank on the FFT data to obtain the equivalent of the Mel frequency in the Mel spectrum, based on sections 2.2.3 and 2.2.4.

6. **Discrete Cosine Transform:** By calculating the discrete cosine transform on the Mel spectrum, we obtain what we call the Mel Frequency Cepstrum Coefficient (MFCC). These coefficients, also called acoustic vectors or cepstral features (Muda et al., 2010), are the features that are fed into a classifier.

   A common choice for the number of coefficients is 13, with 1 coefficient representing the energy of the frame and 12 coefficients representing the velocity features. The number might look weak compared to the amount of information that was initially present in the database, but these features summarize pretty well the quantity of data in the recording (Chauhan & Desai, 2014). We may call it quality over quantities.

   Figure 2.5 illustrates the main steps of the extraction of the MFCC from the recording samples.

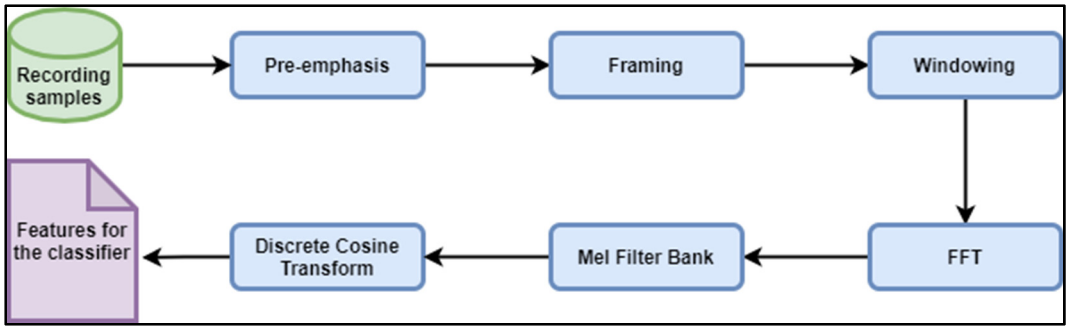


Figure 2.5 Bloc diagram of the MFCC extraction process

2.3 Neural network

In this section, we focus on learning the basics of neural networks, more precisely on the two models that are studied in this research: Long-short term memory (LSTM) and the Transformer architecture. The LSTM network is a variant of the recurrent neural network (RNN), hence why its understanding is essential to comprehend the variant and what problem the latter solves. The multilayer perceptron (MLP) is also studied for comparison purposes only. All the aspects around the model training and optimization are covered in the sections below.

2.3.1 Multi-Layer Perceptron (MLP)

A multilayer perceptron is one of the most common neural networks that has been deployed for all kinds of tasks. The perceptron's predecessor had issues dealing with data that is not linearly separable, like the XOR gate (Ahire, 2020). By adding more layers, it was possible to create a non-linear equation that could solve complex problems.

2.3.1.1 MLP layers

The MLP has three types of layers (Rožanc & Mernik, 2021), illustrated in figure 2.6:

1. **Input layer:** The data is inputted directly to this layer. There is no equation or any calculations as the raw features are inputted into this layer.

2. **Hidden layer:** Contains one (perceptron) or more layers (MLP) that are the body of the architecture. An arbitrary number of neurons on each layer is connected to every neuron in the next or previous layer. Until a certain limitation, adding more layers increase the non-linearity of a classification problem.

3. **Output layer:** The results are obtained at the output layer. For a classification task, the output predicts a probability of appurtenance for each class using the softmax function, which we cover in the sections below. For the prediction task, the output layer gives a value based on the current inputs.

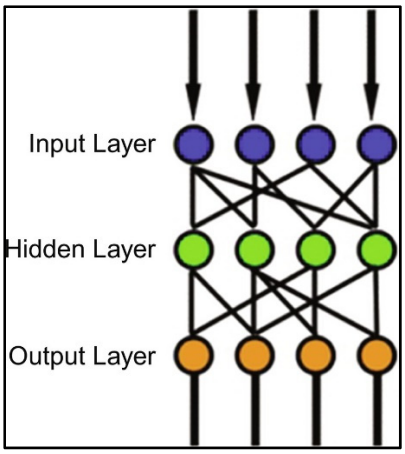


Figure 2.6 Architecture of a Multilayer Perceptron (MLP)
Taken from Rožanc& Mernik (2021)

2.3.1.2 MLP Neuron

The structure of a neuron is illustrated in figure 2.7. The output value of each neuron is the result of the following equation (Menzies, Kocagüneli, Minku, Peters, & Turhan, 2015):

$$y = \sum_{n=1}^{N} f(x_n w_n + b) \tag{2.8}$$

N is the number of links to that specific neuron, x is the input, w is the weight of the link, f is the activation function and b is an arbitrary bias. Replace x with the output of the neuron of the previous layer if the current neuron is in the middle layer.

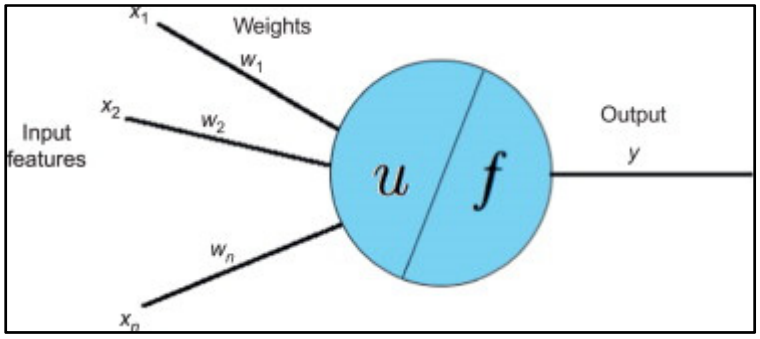


Figure 2.7 Overview of a single neuron property in the MLP
Taken from Menzies et al. (2015)

2.3.1.3 MLP Activation function

Each neuron in the hidden layer consists of a mathematical function, which is called the activation function. Choosing the proper activation function is crucial to the network's performance as it improves significantly (Apicella, Donnarumma, Isgrò, & Prevete, 2021).

Activation functions ensure that the output value of each neuron is contained within a specific range. This way, we limit the summation result from the equation getting bigger and bigger until it is outside the range of computer capabilities. This problem is called the vanishing gradient (Menzies et al., 2015), which is explained in the sections below.

According to Apicella et al. (2021) review, networks using rectified-based activation functions offer improved performance in several studies, such as Glorot, Bordes, and Bengio (2011). ReLU activation is quicker to compute and is better at fighting the vanishing gradient problem (Bengio, Simard, & Frasconi, 1994). The following figures (figure 2.8 and figure 2.9) show examples of classical and rectifier-based activation functions, respectively.

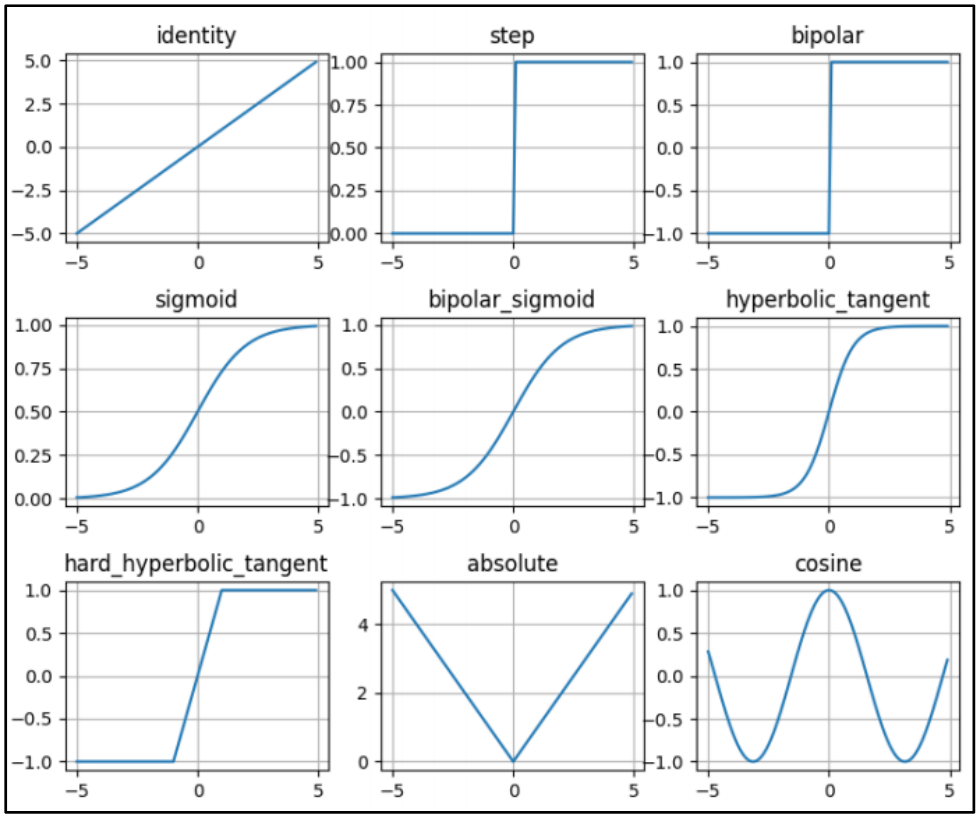


Figure 2.8 Examples of classic activation functions
Taken from Apicella et al. (2021)

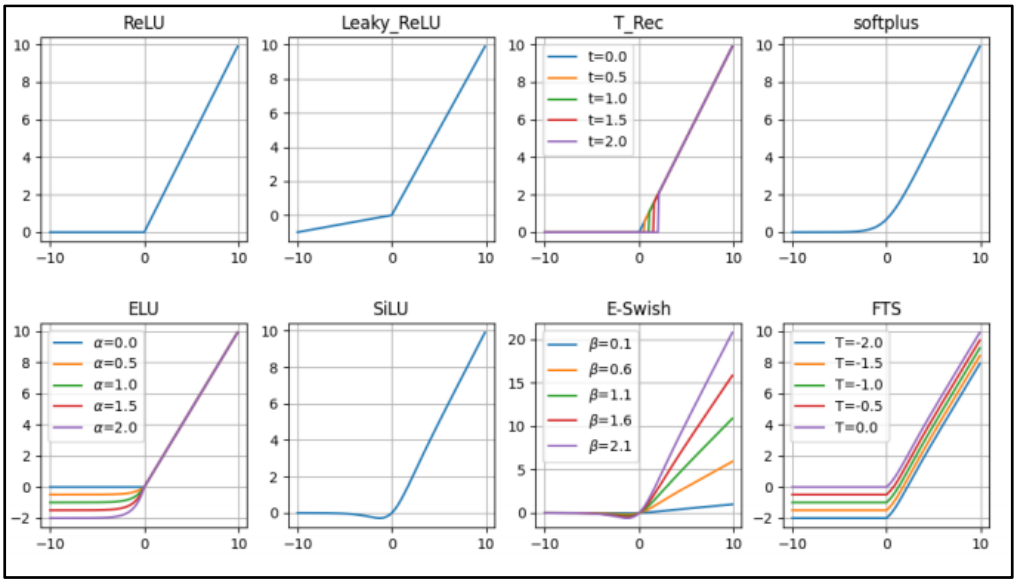


Figure 2.9 Examples of rectified-based activation functions
Taken from Apicella et al. (2021)

2.3.1.4 MLP Training and Loss function

To train an MLP, a backpropagation of the error is used. The ultimate goal is to minimize the loss function of the network by updating the weights between neurons, also known as the parameters of the neural network (Keim, 2019). The formula to calculate the loss function is:

$$E = \frac{1}{2}\sum_{i}(t_i - o_i)^2 \tag{2.9}$$

E is the total error, t is the target output, o is the calculated output and i is the range of all nodes.

By using the gradient descent algorithm, it is possible to head towards the minimum of the function. Figure 2.10 shows this process with three gradient points on the function.

The weights are updated in the nodes using this formula:

$$w = w_{old} + \alpha\frac{\partial E}{\partial w} \tag{2.10}$$

w is the weight value, α is the learning rate (typically 0.1), E is the total error obtained from equation 2.9 above.

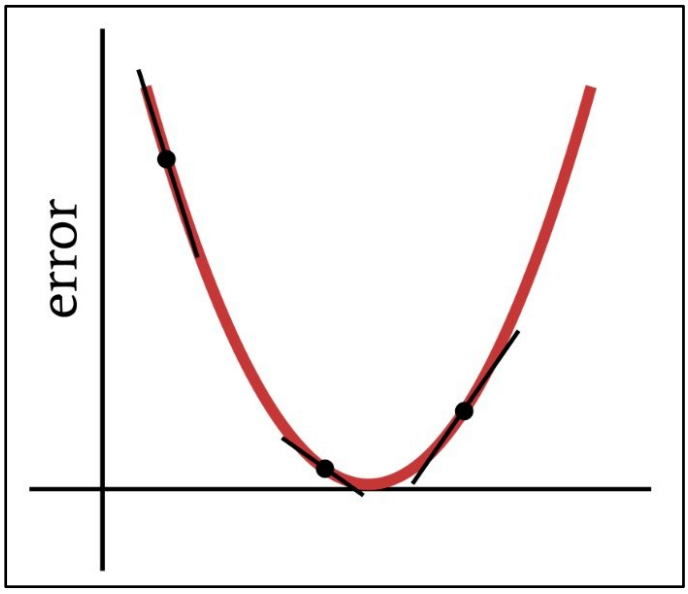


Figure 2.10 Gradient descend algorithm to minimize the error
Taken from Keim (2019)

2.3.2 Recurrent Neural Network (RNN)

The recurrent neural network (RNN) uses a different approach which is more dynamic than the MLP. In fact, there is a circular connection between the neurons from different layers (Staudemeyer & Morris, 2019). A popular architecture is called the Elman network. The particularity about the RNN network is that not only the output can be extracted from each neuron, but also a context vector, which is a piece of information generated from the current input and the previous inputs. Every cell generates this information. It is what creates this memory effect of time in an RNN. Figure 2.11 shows the general architecture.

The memory effect of the cells makes the RNN very popular in natural language processing (Graves, Mohamed, & Hinton, 2013) and market value forecasting. In both these areas, the output data of a network depends on a timed event in which the chronological order of the input is very important. For example, in the first field, a word in a sentence can have different meanings depending on where it is and which other words it is preceded by. In the second field, the variation of the market price in a specific order is the main feature that describes the financial state of the market. The MLP Network does not have this dependency on the time of the event and could easily confound two different data that have the same inputs at different positions.

2.3.2.1 RNN neuron

The structure of the RNN neuron is very simple, illustrated in figure 2.12. The current input and the previous context vector, also called the previous hidden state, are concatenated. It is then passed through an activation function, like the Tanh activation, before being outputted as the new hidden state to the next neuron (Phi, 2018). See section 2.3.1.3 for more details about the activation function of a network cell. This context vector now contains information about all the previous inputs in time and the current input.

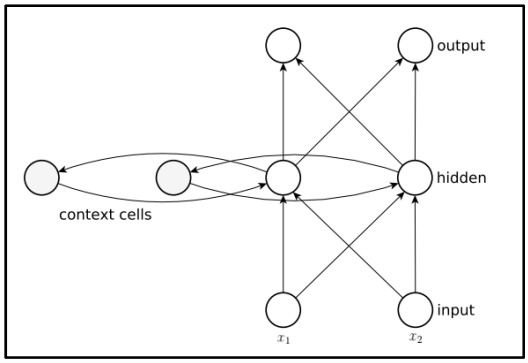


Figure 2.11 Simple architecture of a RNN network
Taken from Staudemeyer & Moris (2019)

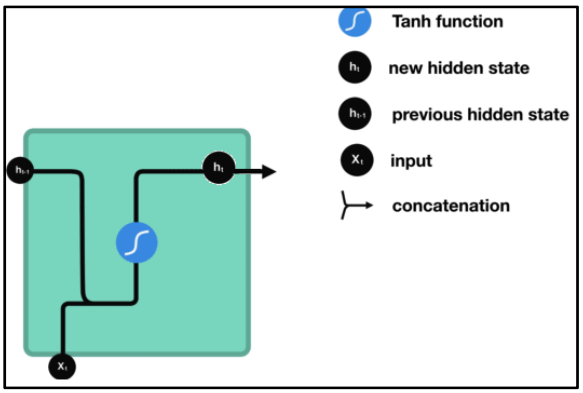


Figure 2.12 Process of each RNN neuron
Taken from Phi (2018)

2.3.2.2 RNN training

To train an RNN, we primarily use the Backpropagation Through Time (BPTT) technique (Staudemeyer & Morris, 2019). The network is unfolded and trained similarly to the MLP in section 2.3.1.4. After the training, the network is unfolded in time. The weights values are optimized to obtain the minimal value using the error backpropagation within a certain time step, called an epoch. They are updated with the sum of its deltas over all time steps, using this equation:

$$w_{new} = -\eta \frac{\delta E_{total}(t', t)}{\delta w_{old}} \tag{2.11}$$

w is the weight, E is the error within a time delta and η is the learning rate.

2.3.2.3 RNN vanishing gradient problem

When the error signal is back-propagated, there is a chance that over many time steps, it may vanish or blow up (Staudemeyer & Morris, 2019). In fact, if the derivate of the equation above outputs a value greater than 1, with several steps, it may lead to an immense value that computers can't process. If the derivate outputs a value below 1, then the value would exponentially decrease until it vanishes, especially if the number of time steps is greater than 10 (Staudemeyer & Morris, 2019). This phenomenon is what we call the vanishing gradient problem. If that happens, the training time is abnormally long, or it might not work at all. This issue is what caused the development of Long-Short Term Memory cells (LSTM).

2.3.3 Long-Short Term Memory (LSTM)

Hochreiter and Schmidhuber (1997) proposed a method called Long-Short Term Memory (LSTM) to overcome the problem he mentioned in 1991. This new method is believed to work with over 1000 time steps without any problem by using what he called "Constant Error Carrousels" (CEC). Hence being used for a long time, the LSTM has known several variants over history. The most famous one remains the vanilla LSTM (Van Houdt, Mosquera, & Nápoles, 2020). This research focuses on this default variant as the other variants proved to be more performant on very specific applications and databases. Applications of the LSTM network in different domains include (Staudemeyer & Morris, 2019):

1. Speech recognition.
2. Handwriting recognition.
3. Machine translation.
4. Image processing.
5. Other areas such as: protection structure prediction, music generation, network security, optimisation problems, etc.

2.3.3.1 Constant Error Carousel (CEC)

To have a constant error flow and ensure that it doesn't vanish, the function that multiplies the weight matrix must be linear, and its activation must remain constant over time. This can be accomplished using the identity function and setting W[u,u] = 1.0 (Staudemeyer & Morris, 2019). It also helps build long-term dependencies and exploit long-range context (Graves et al., 2013). The CEC is also connected to the other units, so the weighted input and outputs must be considered. The exact weight is used for storing and ignoring inputs, hence creating a conflict. To avoid it, the CEC must be used with gates.

2.3.3.2 LSTM input gate, output gate and forget gate

LSTM cells act as a memory and contain information from the very beginning time step. The context is carried all the way to the end of the sequence, hence storing relevant information on long-term dependencies. Since the amount of information is growing over the time-steps, we need to filter some of them to keep the data transfer efficient from one cell to another. This is where the gates come in useful. As the name suggests, they learn to identify which information is relevant to keep or forget during training. Each cell is responsible for transmitting the most valued data to the next cell and filtering some of the information. Two values are updated in the LSTM cell: the cell state (also called the context vector) and the hidden state. The cell state, absent in the classical RNN, is the memory of the LSTM cell, while the hidden state is the amount of information kept from the current input and transferred to the next cell. Both vectors are transferred in the network.

LSTM uses three gates: input gate, output gate, and forget gate to regulate the flow of information. Each of these gates uses a sigmoid activation function (Phi, 2018).

Forget gate: The forget gate comes first and decides the amount of information that should be forgotten. The current input and the previous hidden state are passed through a sigmoid activation function, obtaining a result between 0 and 1. The closer we are to 0, the more information is forgotten at this state.

Input gate: This gate serves to update the cell state. The current input and the previous hidden state go through a sigmoid function to determine the amount of relevant information. The same data also goes through a Tanh activation. Both outputs are multiplied by each other, with the sigmoid one deciding the weight of the information that is kept from the Tanh output. The resulting value is the output of the input gate.

The new context vector can now be calculated using the following procedure:

1. Pointwise multiplication of the previous context vector and the output of the forget gate.
2. Pointwise addition of the #1 output and the output of the input gate.
3. Output of #2 is now the new context vector, or cell state, which is transferred to the next LSTM cell.

Output gate: The output gate determines the new hidden state. This hidden state can be used to predict a value. The combination of the cell state and the current input is carried over to the next cell.

The new hidden state is calculated using the following steps:

1. The previous hidden state and the current input go through another sigmoid function.
2. The newly calculated cell state is passed through a Tanh function.
3. Outputs from #1 and #2 are multiplied by each other.
4. The result from #3 is the new hidden state for the next LSTM cell.

In summary, the forget gate is to filter out the unimportant information from the past cell. The input gate ensures to chose only relevant information from the current input. The output gate calculates the new hidden state of the cell. The new hidden state contains information about all the previous inputs and the current input in the correct chronological order. Figure 2.13 illustrates the process inside the LSTM cell. X is the inputs in different time steps and h is the hidden state.

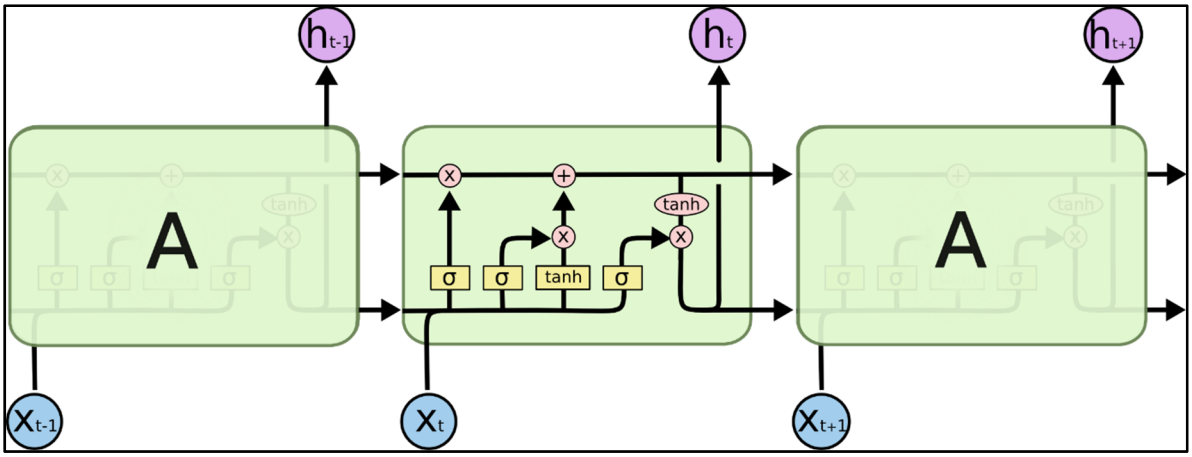


Figure 2.13 The arithmetic operations inside an LSTM cell
Taken from Github (2015)

2.3.3.3 Bidirectional LSTM

Sometimes it is crucial to consider future inputs on an input sequence (Graves et al., 2013). Bidirectional LSTM contains two separate hidden layers, one for data flowing from the beginning to the end and another one for data flowing from the end to the beginning. Those two hidden layers are fed forward to the same output. We obtain a better dependency of longer sequences using this method. However, since the operation is done back and forward, the training time is also doubled. In return, we exploit future data in the current timeline.

2.3.3.4 LSTM variants

Throughout history, several LSTM variants have been developed by many researchers to resolve a specific problem. Not many modifications have known success, but few actually worked pretty well for different tasks and offered state-of-art performances compared to the original version. Among them, there is the Peephole Connection, the Gated Recurrent Unit (GRU), the Multiplicative LSTM (mLSTM) and the LSTM with Attention. Although those variants are not used in this research, it is essential to consider their contribution to the default variant. The LSTM with Attention eventually led to the second type of studied neural network in this thesis, the Transformer model.

LSTM Peephole Connections: In section 2.3.3.2, we observed no connection between the previous state cell and the newly produced hidden cell. The gates do not consider the context vector of themselves. These new LSTM cells provide this connection to use the cell current memory in the calculations before replacing it with a new one and learning to time and count (Gers & Schmidhuber, 2000). Figure 2.14 shows the newly added connections.

Gated Recurrent Unit: The Gated Recurrent Unit (GRU) is a simpler version of the LSTM. They present a simpler version for tasks such as sequence to sequence training and provide a faster training time than the traditional LSTM. The GRU does so by combining the input gate and the forget gate into a single gate (Weiss, Goldberg, & Yahav, 2018). Also, the cell state and the hidden state have been combined into a single vector. Due to this limitation, the GRU also proves to be less powerful and offers lower performance than the default LSTM variant (Weiss et al., 2018). It is, however, the price to pay for a quicker training speed. Figure 2.15 illustrates the architecture of the GRU model.

Multiplicative LSTM: This variant has been quite popular since it was introduced in 2016 by Krause et al. It has known success in natural language processing. It has proved to outperform Stanford Sentiment Treebank using dramatically less data (Radford, Sutskever, Józefowicz, Clark, & Brockman, 2017). It is a complicated variant but performs well on unsupervised training with minimal fine-tuning.

LSTM with Attention: LSTM with Attention mechanism offers state-of-art performance in the sequence modelling area. It includes the Attention mechanism, same as the famous paper "Attention is All you Need" (Vaswani et al., 2017). Several works, including Wu et al. (2016), used this method to obtain state-of-the-art performance in the field of neural machine translation. Variants like BERT, ELMO, GPT-2 use this architecture (Exxact, 2019).

As suggests its name, the Attention mechanism is the network's ability to focus on some aspects of the data. For the LSTM network, the focus is on the hidden states in the LSTM cells.

Section 2.3.4.2 contains all the information about Attention in a network, as it is the core of the Transformer network.

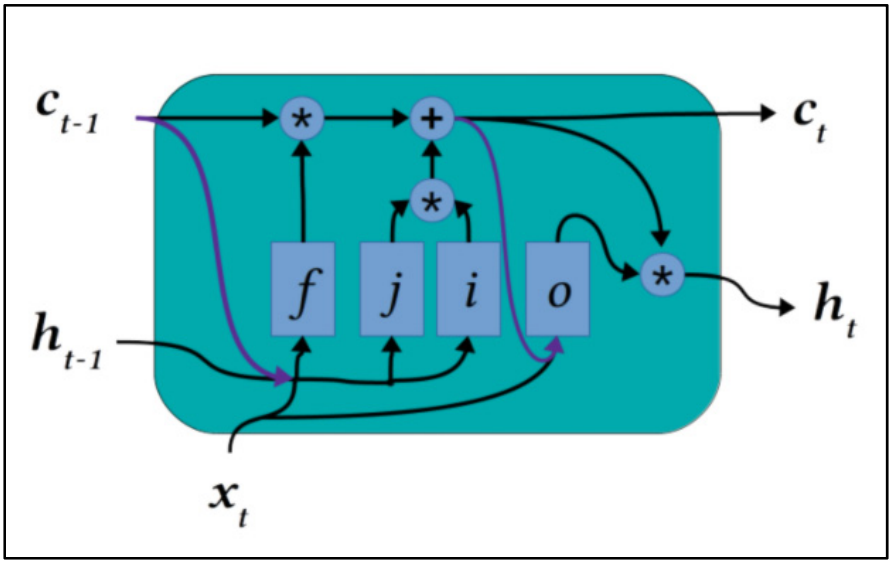


Figure 2.14 LSTM Peephole Connections. Dark purple arrows are the added connections compared to the classical LSTM
Taken from Exxact (2019)

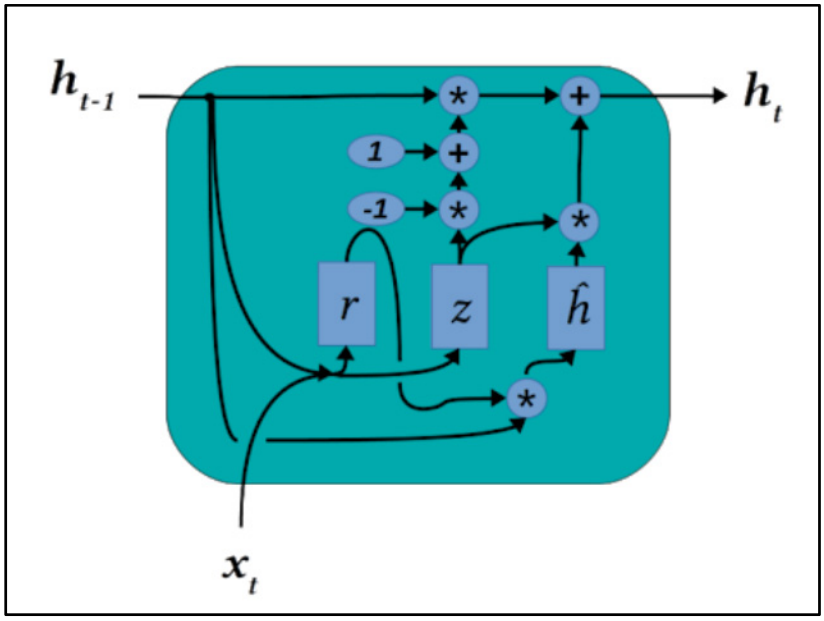


Figure 2.15 Architecture of the Gated Recurrent Unit (GRU)
Taken from Exxact (2019)

2.3.4 Transformer Network

The Transformer model was introduced in 2016 with the paper “Attention is all you need” by Google research team (Vaswani et al., 2017). The core idea behind this is to have an architecture that is not only dependant on recurrences like the RNN or LSTM. By removing the dependence on sequential computation, we can achieve parallelization during the training phase, hence exploiting modern Graphical Processing Units (GPU) hardware power to its full potential and significantly reducing the training time. To achieve this, the Transformer network sorely relies on the Attention mechanism in its encoder-decoder architecture.

Like its predecessor, the RNN, the Transformer works well on data that follows a certain chronological order. It has been known to offer improved performance overall on natural language processing (NLP) tasks such as speech recognition, language modelling, machine translation and question answering (Uszkoreit, 2017). The network outperforms any RNN or convolution models on language translation, English to German and English to French.

2.3.4.1 Encoder-decoder architecture

The Encoder-decoder is not unique to Transformer. It has been used for other tasks such as text simplification using LSTM cells in the layers (Wang, Chen, Amaral, & Qiang, 2016). The encoder takes a variable-length input vector, like texts containing a different number of words to translate, and output a fixed-length vector to the decoder (Cho et al., 2014). The fixed-length vector contains a summary of all the inputs with their corresponding position. The decoder receives this summary vector and uses it to output a variable-length vector, using its own output and all the information from the fixed-length vector. It is especially relevant to variable-length input and output, but a fixed-length vector can also be used if the database contains the same amount of words or any form of input. On the decoder part, each step's output is reinjected into the decoder to predict the next word or data. However, the fixed-length vector the decoder received at each time step stays the same.

2.3.4.2 Attention mechanism

The Attention mechanism was created to replace the vanishing gradient problem in the RNN structure in sequence modelling (Adaloglou, 2020). When the number of time steps grows, the information contained in the first time steps is forgotten over time. As mentioned by (Adaloglou, 2020), « The core idea is that the context vector z should have access to all parts of the input sequence instead of just the last one ». As the name suggests, particular attention is given to the input sequence as the network learns where to focus on the input data.

The process used in Vaswani et al. (2017) work is called the Scaled Dot-Product Attention, which is illustrated in figure 2.16. The following equation represents it:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

Q, K and V are matrixes, d_k is the dimension of the input vector.
Q contains the vector representation of one word, while K (keys) and V (values) are vectors representing the entire input sequence.

By dividing the dot products by the square root of the dimension, we ensure that if the value of the dimension is substantial, we don't end up with very small gradients after the softmax function. On top of the dot-product, the additive attention can be beneficial, but the latter is slower and less space-efficient in practice (Vaswani et al., 2017).

The attention mechanism adds to the learning process a method to focus on specific traits of the input sequence. With the help of several input data, the network learns how to recognize those traits and focus on them during the training session and during the classification session (Adaloglou, 2020). In the image processing domain, it is easier to visualize those traits. It can be the contour of an object, the contrast of the image, the brightness, the sharpness, the warmth, etc. Depending on what the mechanism learns to focus on, it can also be a specific object to detect on an image. Once the learning process begins, the more correlated data the network

receives, the more it can refine its knowledge. However, it is not always simple to determine those traits that the network will focus on. For instance, in natural language processing (NLP), it is uncertain which are the characteristics of the attention mechanism, and it greatly varies on the nature of the sound present in the audio file.

While the network learns to focus on specific traits of the input, it is also nearly impossible to extract that information from the attention layer. In fact, after the training is done, one cannot know what traits the neural network focused on during the training, making the comprehension of this process more complicated. In this research, the most probable characteristics are the pattern present in the sound frequencies and the variation of the frequencies over time, which can be observed with the MFCC.

One noticeable downside of the Transformer network and its Attention mechanism is the amount of required data in order to successfully train the classifier (Wu et al., 2016), especially when there are multiple heads of Attention to train, which will be covered in the next section.

Two types of Attention exist, self-Attention and regular Attention (Uszkoreit, 2017). The Attention has access to all the other layer's data and activations. In contrast, the self-Attention can only access the inputs of the same layer, making it more effective when looking at the similarities of the words in the same input sequence. The Attention allows creating the link between the encoder and the decoder, since it knows the dependencies between several different inputs, like different sentences. Both mechanisms are helpful during the process of creating the neural network. In certain applications, no decoder is required if the output is fixed-sized and unique. In this case, only the self-Attention mechanism is used.

2.3.4.3 Multi-Head Attention

Multi-Head Attention enables the computation of the Attention mechanism multiple times in parallel, making higher usage of the processing unit (Vaswani et al., 2017). By doing so, each head can focus on a particular aspect of the input and globally pay attention to different facades

of the incoming data. For example, few heads can focus on longer-term dependencies while others on shorter-term dependencies. In the end, all the results are concatenated into the final output vector.

Each of the following heads will learn to focus on a specific trait of the input. To determine the number of heads to maximize the performance is another issue because it is always uncertain how many different traits the network can learn to focus on. It is why in most studies, the number of heads is an optimized hyperparameter, no matter the nature of the input data. In fact, the accuracy can greatly vary if this number is poorly chosen.

Figure 2.16 illustrates the main steps inside an Attention mechanism, as well as the multi-head architecture.

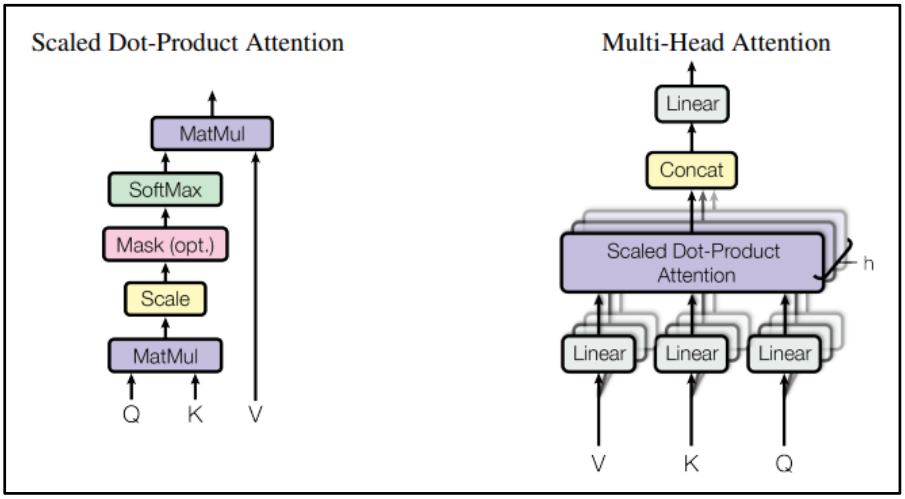


Figure 2.16 Steps of a Scaled Dot-Product Attention (left) and the usage of multiple head Attention in parallel (right)
Taken from Vaswani et al. (2017)

2.3.4.4 Input Embedding

Since the Transformer has been developed for natural language processing, the inputs are obviously not numbers but words. Each word in the vocabulary has a numerical value associated with it. For each value, we create a vector containing initially random values called

"linguistic features". The vector can be represented in a multidimensional space. In the original work from Vaswani et al. (2017), the author uses an embedding vector of 512 dimensions. As the model goes through the learning phase, these values are updated. Two words with similar linguistic features tend to get closer and closer in the multidimensional space as different inputs, sentences perhaps, are fed into the network.

2.3.4.5 Positional Embeddings

In the LSTM architecture, the order of the inputs is kept since it is processed sequentially. On the Transformer model, the treatment is parallel. The positional information of the data is now lost. It is problematic since the order in which a word appears in a sentence can completely modify the context of the idea that is being expressed. To resolve this issue, a positional encoding is required to keep that crucial piece of information.

The author uses sine and cosine functions of different frequencies into a matrix with the same dimension as the input embedding layer and sums those matrices to form the new input for the model. This method works because each value in the encoding is unique for each time step. Also, the distance between two time-steps is consistent for variable-length input. The positional encoding values are obtained using the following equation:

$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{\frac{2i}{dim}}}\right) \tag{2.13}$$

$$PE(pos, 2i+1) = cos\left(\frac{pos}{10000^{\frac{2i}{dim}}}\right)$$

pos represents the position of the embedding, i is the dimension and dim represents the length of the input embedding.

The following figure 2.17 represents an example of the different values of the resulting positional embedding containing 128 dimensions for the embedded input and a maximum length of 50 words. Each value is set to be unique on both axes.

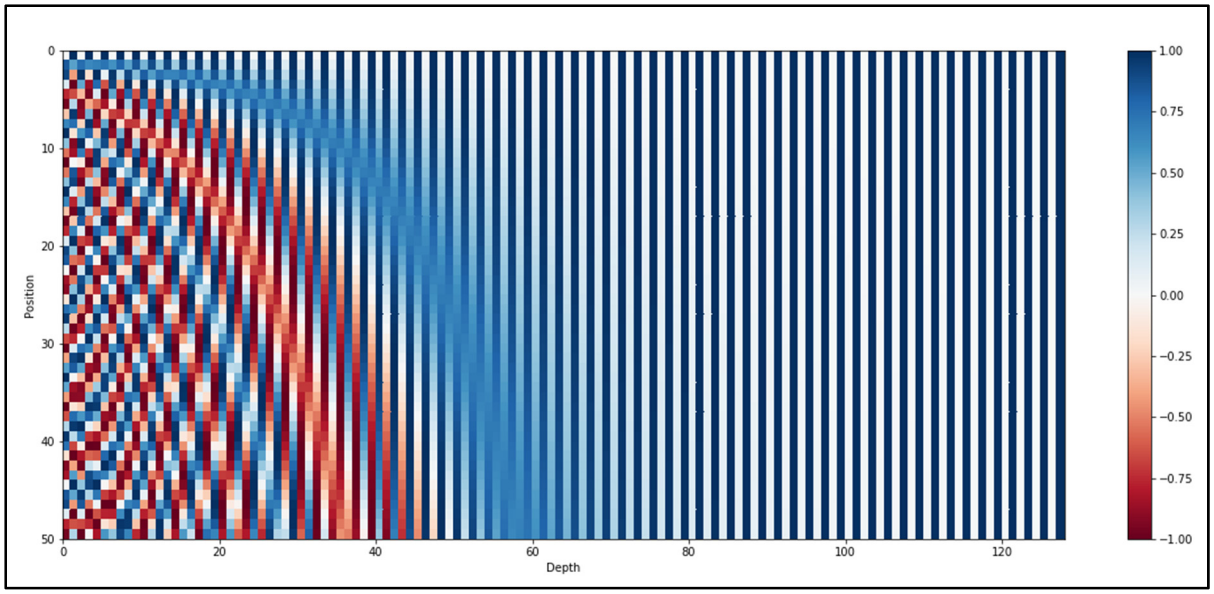


Figure 2.17 Resulting positional embedding using sine and cosine function

2.3.4.6 Transformer model architecture

By combining all the parts in the previous sections, we obtain the global Transformer model, which consists of layers of encoders and decoders. Although it is an arbitrary number, Vaswani et al. (2017) decided to go with 6 identical layers for each.

Both the encoder and the decoder contains very similar blocs. The decoder receives the same type of information as the encoder. In applications where the output is fixed length, the decoder is completely removed.

In the encoder, we have:

1. **Multi-Head Attention layer:** Several heads focus on different input facades and give special attention to different features in the data.
2. **Feedforward network:** A fully connected network that contains most of the trainable parameters of the Transformer. See section 2.3.1 for more information.
3. **Add and Normalization layer:** Residual connection for both previous layers followed by a normalization layer.

In the decoder, we have:

1. **Masked Multi-Head Attention layer:** The same algorithm as the Attention mechanism. Since the Transformer network is supposed to work in parallelization, the decoder layer can see the entire expected output. For example, in language translation, the first word that is outputted from the network should not see the subsequent words and bias the result during the training phase. For this reason, when the first word is being predicted, the entire output is masked. When the second word is outputted, only the first output is not masked, and when the third word is outputted, only the first and second output is not masked, and so on. Now the training is more natural as the model is learning the new language.
2. **Multi-Head Attention layer:** The query and key (Q and K from Scaled Dot-Product Attention equation) of this layer comes from the last encoder layer and it contains the summary of the entire input sequence. The value (V) is extracted from the masked layer.
3. **Feedforward network.**
4. **Add and Normalization layer.**

Finally, the decoder predicts the word using a linear layer, which serves as a classifier and has the length of the number of possible classes. In the field of language translation, it would be the number of words in the vocabulary. The output is then passed through a softmax layer, calculating the probability for each class. Ideally, the highest probability class would be the final output of the network. Figure 2.18 shows the architecture of the Transformer model that has been described in the previous sections. It is also the model that was used in the research, with a few adjustments.

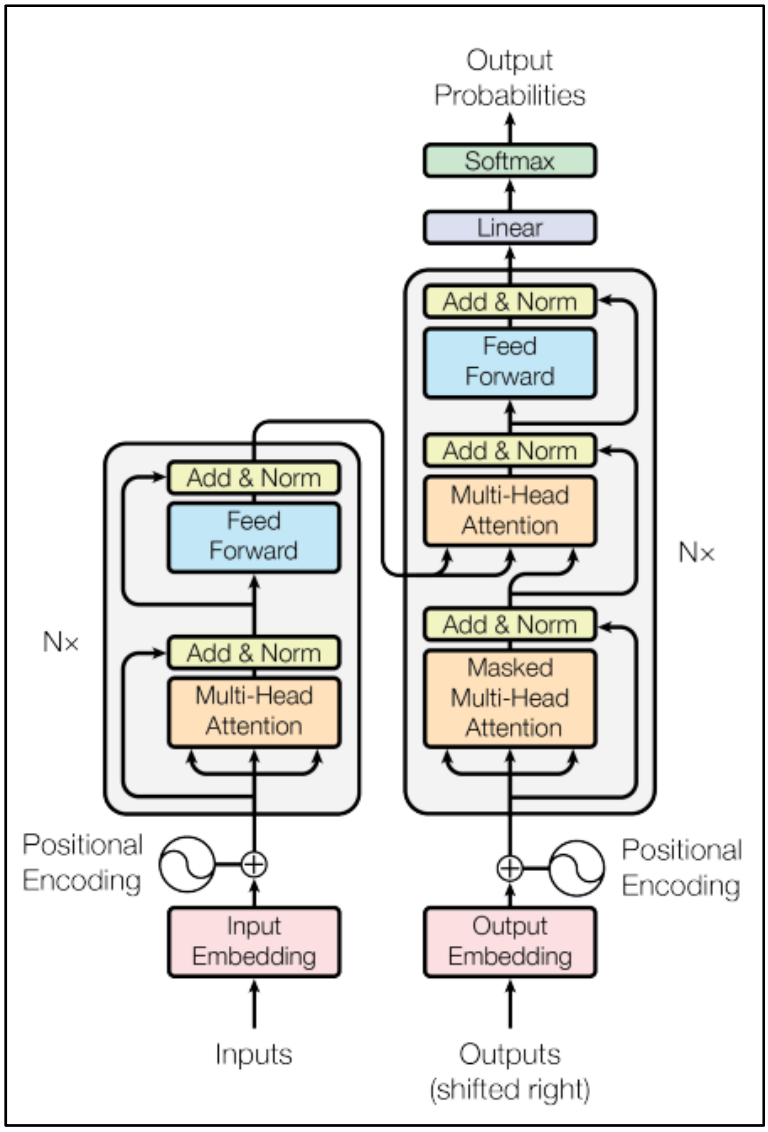


Figure 2.18 The Transformer model architecture
Taken from Vaswani et al. (2017)

2.4 General training concepts

In the following section, concepts essential to understanding to train the neural network for classification efficiently are discussed. To obtain optimal performance, it is crucial to understand the various hyperparameter that can be manipulated and their consequences on factors such as the accuracy of the network and the training time.

2.4.1 Supervised and unsupervised training

Supervised training means that we train the network using data that has been labelled by experts, while unsupervised training contains data for which we do not always know its class (Delua, 2021). Data annotation can be very expensive as it requires labelling all the data, knowing that neural networks generally require a tremendous number of samples to work efficiently. An unsupervised training algorithm has been developed to address this issue.

Supervised methods are mainly used for classification and regression problems. Knowing the actual classes of the dataset helps to classify unknown new data. Unsupervised training is used on clustering problems, in which data with similarities are regrouped into clusters. No label is required since they are compared between each other and not to the desired class value at the network's output.

2.4.2 Dropout rate

Overfitting is a common problem with neural networks, especially with a deep network with many parameters. The network learns too much the training data and gets a low error value, but it will perform poorly on testing data, utterly new to the system. A solution is to implement Dropout, a technique that randomly drops cells and connections during the training. It regularizes the network and offers better robustness since it forces the network to work with less available information.

This technique also helps to improve the training speed. In fact, there are fewer neuron cells in the network and fewer connections, meaning there are fewer computations. In Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014) work, adding Dropout to the network improved the performance in applications such as object classification, digit recognition, speech recognition, document classification and analysis of computational biology data. A standard Dropout is 0.2, which means that 20% of the cells count is randomly dropped at each training stage. Figure 2.19 illustrates the concept.

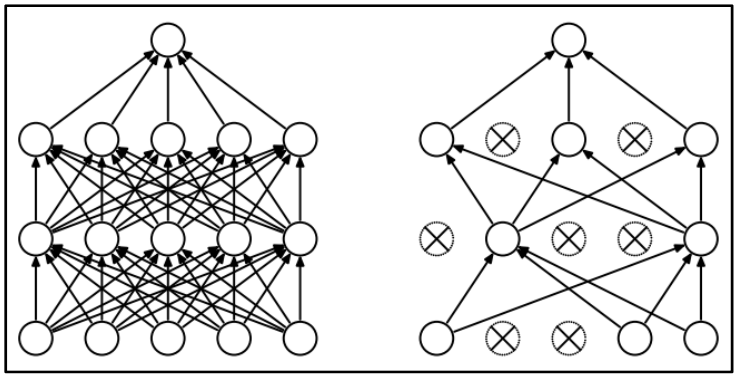


Figure 2.19 Standard model compared to Dropout model
Taken from Srivastava et al. (2014)

2.4.3 Epoch and batch size

The number of epoch consists of the number of times the network goes through the entire training dataset in random order. The batch size is the number of samples from the dataset processed before the model is updated. Having a big batch size is not always good because the processing hardware might not have enough memory to store the calculated values. On the other hand, a smaller batch size creates a higher gradient fluctuation.

2.4.4 Early stopping

Another solution to avoid overfitting is to add early stopping to the network. This technique consists of stopping the training when it reaches a particular condition. The most common trigger is to stop when the loss function stops improving on the validation data and recall the network with the lowest loss function. The epoch number is set at a high number (ex. 300), and the network automatically stops when it doesn't learn anymore.

2.4.5 Hyperparameters

Hyperparameters optimization is the key to obtain the best performance out of a neural network. Parameters, which are the network weights, are optimized during one training session that could last, for example, 300 epochs. Hyperparameters are optimized after several training

sessions, hence taking most of the available training duration. Identifying which hyperparameters influence the performance significantly takes many experiments, and it is hard to predict the accuracy variation. Examples of hyperparameters are:

1. Number of layers of neurons.
2. Number of encoders and decoders for the Transformer.
3. Learning rate during backpropagation.
4. Number of neurons on each layer of the network.
5. Dropout rate.
6. Number of heads in the Multi-Head Attention mechanism.
7. Early dropping trigger method.

2.4.6 K-fold Cross-Validation

Luck can be part of the play when a set of training data and test data obtains abnormal high performance compared to what it gives in real conditions. To avoid this situation, a k-fold Cross-Validation technique is used (Raschka, 2018). The principal idea is that all samples in the dataset get a chance to be used as test data. Some variants use validation for each training step and separate testing data. Below are the steps for a k-fold Cross-Validation for K=5, which means we perform the training tasks with 5 sets of testing data:

1. Separate the whole dataset to 4/5 training and 1/5 testing data.
2. Perform the training and extract performance value on testing data.
3. Reset the whole network and parameters.
4. Separate the dataset again but using the second 1/5 as testing data.
5. Perform the training and extract the new performance value.
6. Execute 3 more times with different testing data each time.
7. The final performance is the average of all 5 performances.

Any abnormal performance is averaged, and the global performance value is more reliable to publish for other researchers to consider. Figure 2.20 illustrates the major steps of the process.

Figure 2.20 K-fold Cross-Validation procedure with K = 5 folds

CHAPITRE 3

NEURAL NETWORK EXPERIMENTATION

This chapter describes the experimentation methodology used to compare the different neural networks' performance accurately. Different hyperparameters are tested to minimize the error function and optimize the performance and training time ratio. Supervised training is used in this research to compare these four neural network models:

1. Multilayer perceptron (MLP) as a reference performance.
2. Long-Short Term Memory (LSTM).
3. Transformer.
4. LSTM enhanced with an Attention mechanism.

Figure 3.1 illustrates the procedure that has been used in this research. All the mentioned models follow the same architecture and the same data for a fair comparison.

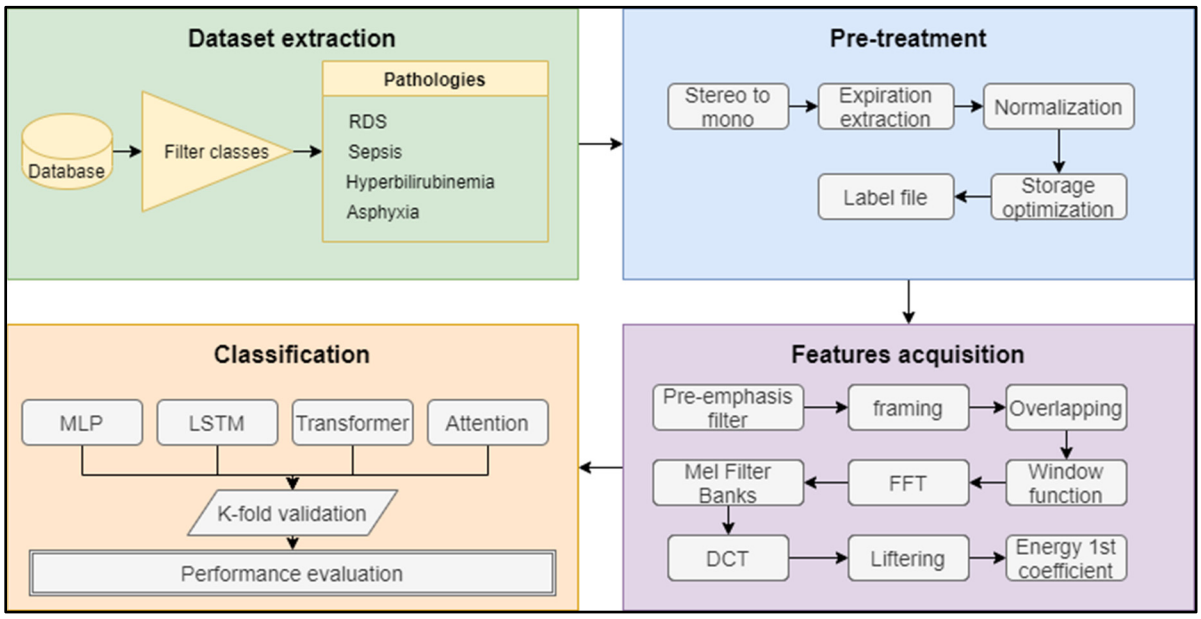


Figure 3.1 Flowchart of the neural network experimentation from dataset extraction to the classification of the pathologies

3.1 The initial database

The recordings used for the experiments are sourced from several hospitals in Canada and Lebanon and are stored in the École de Technologie Supérieure database. With the help of the nurses and experts, each recording is followed by helpful information such as the time and date of the recording, the infant's ethnic group and sexes, the cause of the cry, the affecting disease, and the newborn's gestational age.

Our research only uses the pathology or healthy tab followed by the gestational age. Protophones of preterm and full terms infants presents many spectrographic differences (Oller et al., 2019). We have used full terms only not to confuse the neural network. Also, there is not enough data to classify both types independently to compare their impact.

The recordings were done using an *Olympus* device with two channels. The data is gathered with a sampling frequency of 44.1 kHz, and each sample has a resolution of 16-bits. The microphone was being held between 10 cm and 30 cm from the infant. We must consider that the recordings contain noises from the hospital environment. During the training phase, these noises should be either considered or discarded using a pre-treatment algorithm.

Fortunately, Abou-Abbas et al. (2015) worked on this database to extract only the important parts of the signal by attributing labels to every segment of the audio. Figure 3.2 contains an example of the segmentation label file. In our research, only the expirations sessions are used to train the neural network, labelled "EXP".

3.1.1 Analyzed pathologies

Among the 61 identified diseases in the database, we have conducted research with four diseases. There is a minimal number of samples for most of the pathologies, and it is ineffective to compare neural networks trained with very few samples. We have decided to go with two that contain many samples and two that are rare but still acceptable for neural network training. The following table 3.1 contains the chosen pathologies and the numbers of recordings.

Table 3.1 Number of recordings for each studied pathology

Pathology	Number of recordings	Expirations
Healthy	806	39 240
Respiratory Distress	214	9738
Sepsis	90	4475
Hyperbilirubinemia	21	803
Asphyxia	10	411

A complete list of the diseases is presented in Annexe 1.

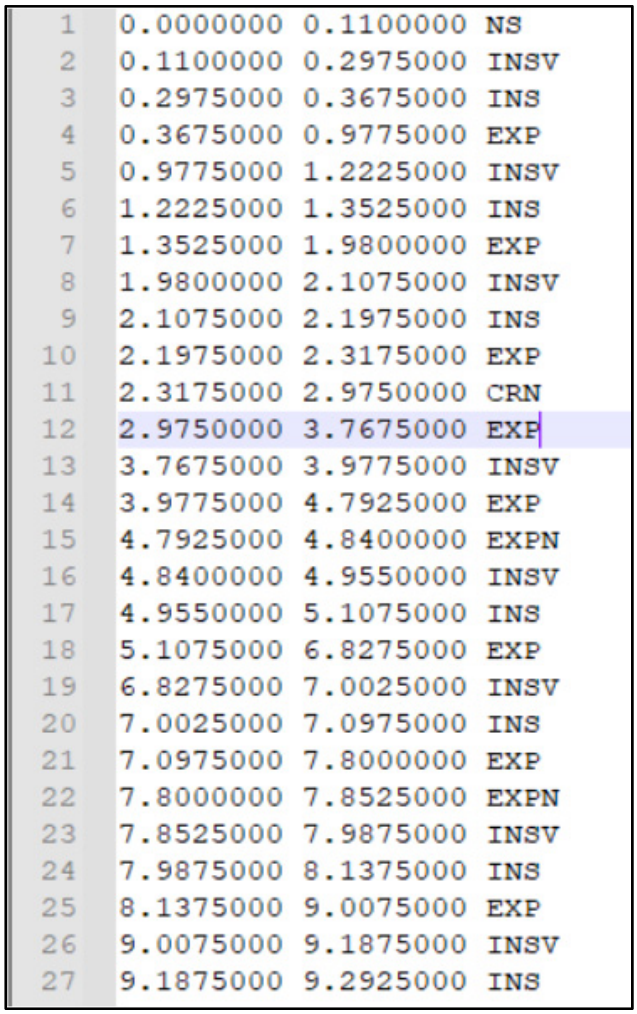


Figure 3.2 Example of segmentation annotation for a single recording file

The model performance is also compared to the previous studies on the subject, especially those used in (Farsaie Alaie et al., 2016). In their studies, they don't use specific diseases but categories. We have extended the case by choosing one or two pathologies in each category, except the cardiac problem-related pathologies, since all the classes contained very little data. The following figure 3.3 contains a list and a description of these categories.

Categories	Description
Healthy infants	Full-term infants without any major disorder or sickness
Heart problems	Full-term infants suffering from tetralogy of fallot, thrombus, complex cardio or congenital heart diseases
Neurological disorders	Full-term infants suffering from sepsis or meningitis
Respiratory diseases	Full-term infants suffering from respiratory distress or asphyxia diseases
Blood abnormalities	Full-term infants suffering from hyperbilirubinemia or hypoglycemia diseases
Other	Full-term infants suffering from other abnormalities or physical problems which are not in priority order for our system

Figure 3.3 List of health conditions in the previous study
Taken from Farsaie Alaie et al. (2016)

3.1.2 Classes definition

Our experiments follow a binary classification, which means there are only two classes per model. Each set of data contains healthy samples and one of the four pathologies identified in section 3.1.1. The last set contains healthy children and all the four pathologies combined. The idea is to determine if the neural network can identify the difference between a healthy infant and one containing a disease.

Some pathologies are easily identified as the features contain a certain pattern, but some are very resistant to similarities hunting. We want to identify the hardest ones to either avoid working with them in the future or finding more meaningful features that could distinctively offer better performance. The following table contains the different datasets used.

Table 3.2 Classes of the neural network

Set of data	1st Class	2nd Class
#1	Respiratory Distress	Healthy
#2	Sepsis	Healthy
#3	Hyperbilirubinemia	Healthy
#4	Asphyxia	Healthy
#5	All four pathologies	Healthy

3.2 Pre-treatment on the data

We worked on data entirely stored as a Numpy array. Every extraction process results in an array stored on the local computer, and every next step either uses an already extracted array or creates it from the database. Using this method saved a lot of time since any steps can be resumed at any moment without the need to rework the database.

We chose the Numpy array because it is very memory-efficient when working with a multidimensional array and offers a faster speed than conventional lists in Python (University of Central Florida, 2021).

The following steps are used to prepare the data for neural classification:

1. WAV file stereo to mono Numpy array.
2. Extraction of the expiration's sessions.
3. Normalization of every expiration in a single recording.
4. Optimization of the storage.
5. Creation of the label files with only full-term children for every disease.

3.2.1 WAV file to Numpy

Each recording is first converted to a Numpy array, with each value being a 16-bit integer. Since it contains two channels, we convert both arrays into one using a simple average function and rounding the value to keep it as an integer. The following formula is used:

$$dataMono[x] = int16\left(\frac{left[x]+right[x]}{2}\right) \text{ for } x=1,2,3\ldots N$$

N represents the number of samples, left and right are the two channels.

3.2.2 Expiration's sessions extraction

Not all data in the database is useful to us, as some contain silence, machines noise, doors shutting, etc. Also, there are data with interference, like people talking in the background, which could eventually alter the neural network's performance. As shown in figure 3.2 above, every recording segment has a label that defines the present sound in that segment. We gathered only the expiration sessions for our work.

In Salehian Matikolaie and Tadj (2020) work, the inspiration and the expirations sessions have been tested separately using a linear Support Vector Machine (SVM) classifier. They concluded that the expirations sessions always offer a higher accuracy no matter the number of MFCC coefficients.

Using a simple time to sample number algorithm, we extracted only those specific segments and created a new smaller dataset containing the several expiration segments of each recording. The following steps only use the newly created dataset.

3.2.3 Normalization of the data

As discussed in section 2.1.2, normalization is crucial to adjust the energy level of all the expiration sessions so the range of the values stays within certain boundaries. It helps maintain a balance during the training session with values that do not explode or disappear with many time steps by being either too big or too small.

After concatenating all the expiration sessions in a single recording, we calculate the mean and the standard deviation. Then, using the standardization formula below, the new values for the data are outputted and saved on the machine.

$$X' = \frac{X - \mu}{\sigma} \tag{3.1}$$

X is the current data, μ is the mean value and σ is the standard deviation.

3.2.4 Optimization of storage

As the steps accumulate, we noticed we have several copies of the data stored in the computer and it requires a lot of storage to keep them all. To continue with this mechanism of storing all the sub-steps to save time and work on processed data, we need to find a way to efficiently compress and optimize the data while only losing a negligible amount of information.

The results are stored as a floating number in the previous step with a 64-bit precision since there is a division in the standardization formula. By studying the largest number in the whole dataset, we determined that we can safely multiply each result by 2^{13} and store the new values as a 16-bit integer. By using a power of 2, the calculations are faster since it requires shifting the bits only. We proceeded to compress the data by a factor of 4, which is a significant difference in the amount of storage required. For further steps, this multiplication is considered, and only a bitshift division by 2^{13} is required to recover the initial value.

To ensure we did not lose much information by this compression, we compared the mean and the standard deviation of the original 64-bit floating-point of every single recording and gathered statistics on the difference after doing the compression. Table 3.3 below shows that the difference is negligible and should not affect the integrity of the neural network performance. The values shown are on the scale of $2^{13} = 8192$.

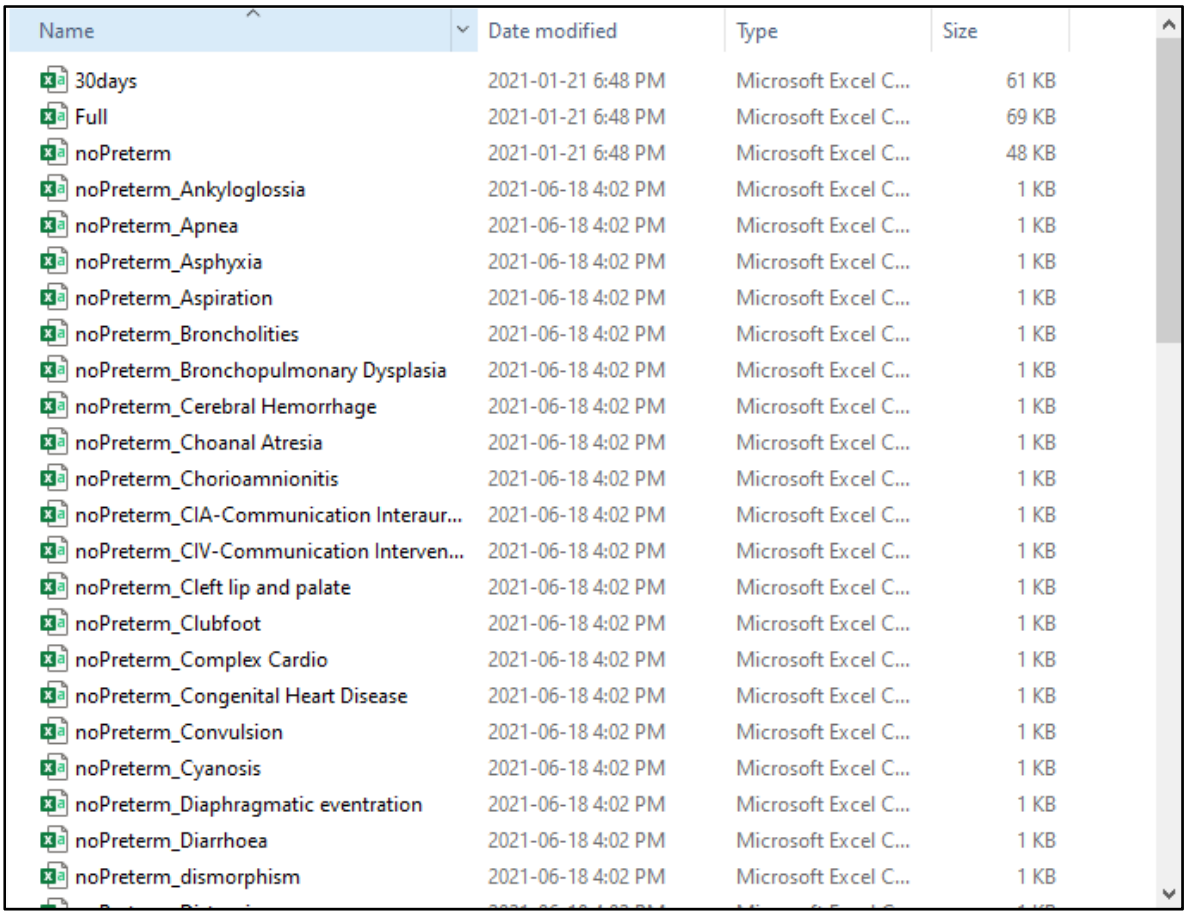
Table 3.3 Difference between compressed and uncompressed data

Factor	Minimum	Maximum	Average (scale of 2^{13})
Mean	-0.010920289	0.0454747	0.000458447
Deviation	0.000314677	0.013844347	0.000944568

For a normal standardization, the average should be around 0. We can see that there is no much variation on compressed data from that middle point.

3.2.5 Label file creation with full-term infants

In this step, the label file is processed. All the preterm infants are filtered out and the remaining are classified by their disease. The resulting output is a set of CSV files for every identified disease containing only the path to the recording file with no preterm infants. Figure 3.3 shows an example of what is stored on the processing computer after this step.



Name	Date modified	Type	Size
30days	2021-01-21 6:48 PM	Microsoft Excel C...	61 KB
Full	2021-01-21 6:48 PM	Microsoft Excel C...	69 KB
noPreterm	2021-01-21 6:48 PM	Microsoft Excel C...	48 KB
noPreterm_Ankyloglossia	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Apnea	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Asphyxia	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Aspiration	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Broncholities	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Bronchopulmonary Dysplasia	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Cerebral Hemorrhage	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Choanal Atresia	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Chorioamnionitis	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_CIA-Communication Interaur...	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_CIV-Communication Interven...	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Cleft lip and palate	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Clubfoot	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Complex Cardio	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Congenital Heart Disease	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Convulsion	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Cyanosis	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Diaphragmatic eventration	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_Diarrhoea	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB
noPreterm_dismorphism	2021-06-18 4:02 PM	Microsoft Excel C...	1 KB

Figure 3.4 Example of label files for every identified disease

3.3 MFCC features extraction

The features for our neural network are extracted following the procedure in section 2.2.5 and summarized in figure 3.5 below. An example of the result can be found in Annexe 3. The following steps indicate the processing of the outputted data from the pre-treatment section to the coefficients using the Python library from Lyons (2013) on Github:

1. Pre-emphasis filter with $H(z) = 1 - 0.97z^{-1}$.
2. We separate each expiration by a frame of 1024 data, which corresponds to about 23.22ms of audio. Farsaie Alaie (2015) obtained their highest performance at 20ms framing. By using a power of 2, we ensure that no padding is required when calculating the Fast Fourier Transform (FFT) two steps below.
3. Overlapping of 50%, which means every 512 data, a new framing is produced.
4. The windowing function is the hamming window.
5. The FFT is calculated on each frame with a size of 1024 samples.
6. A set of 26 Mel Filter Banks is applied with frequencies ranging from 0 Hz to 4000 Hz.
7. Discrete Cosine Transform (DCT) is used to extract the first 12 coefficients.
8. Liftering is applied on the coefficients, with a sinusoidal window of L = 22.
9. The first coefficient is replaced by the total energy value of the expiration session.

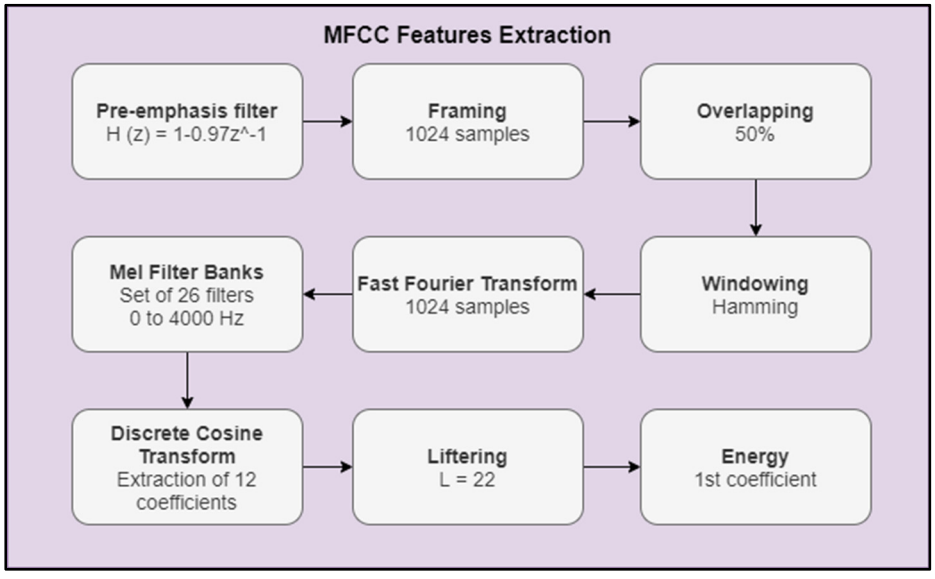


Figure 3.5 MFCC features extraction process steps

3.4 Neural network models training

We focused our research on four different neural networks and compared them based on the same criteria: accuracy and training time. The four models are:

1. Multilayer Perceptron (MLP).
2. Long-Short Term Memory (LSTM) cells.
3. Transformer.
4. Long-Short Term Memory enhanced by Attention mechanism.

For each model, different hyperparameters are tested. Unfortunately, these factors vary from one model to another, and some of the hyperparameters are not present in every model. The conducted experiments observe different ones, and the consequence on accuracy and training time is noted for each step.

The following approach was used:

1. Train the neural network with every hyperparameter individually and observe the accuracy and training time impact of each.
2. Gather the optimal hyperparameter for each category and test the overall network with the best hyperparameters.

In the following sections, we determine which hyperparameters are tested for each model and build the general architecture of the neural network.

3.4.1 MLP training

The MLP network is purely used as a reference model for our system. The accuracy and training time helps to determine if the studied models in this research improve our problem hypothesis. Figure 3.6 is the architecture of the MLP network.

Table 3.4 indicates the hyperparameters that were optimized during the process.

Table 3.4 Hyperparameters of the MLP network used in this work

Hyperparameters	Starting value	Ending value	Step
Number of layers	1	10	1
Number of neurons per layer	8	4096	$2^n,\ 3 < n < 12$
Dropout rate	0	0.5	0.05
Learning rate	0	0.3	0.02

3.4.2 LSTM training

Bidirectional LSTM is used in our research to unlock the full potential of the LSTM network. It requires more training time, but more substantial dependencies between past and future inputs are created. Figure 3.7 is the architecture of the LSTM network that has been deployed. The number of neurons in the dense neural network (DNN) has also been optimised.

Average pooling and Max pooling layers enable to reduce the dimensionality of the output after the LSTM cells. The masking layer enables to have a variable-length input. The inputs with zero values are considered the ending of an expiration session.

Table 3.5 indicates the hyperparameters that were optimized during the process.

Table 3.5 Hyperparameters of the LSTM network used in this work

Hyperparameters	Starting value	Ending value	Step
Number of layers	1	10	1
Number of neurons per layer	8	1024	$2^n,\ 3 < n < 10$
Dropout rate	0	0.5	0.05
Number of neurons in the DNN	8	8192	$2^n,\ 3 < n < 13$

3.4.3 Transformer Training

Our Transformer does not require a decoder since there is only one output, the predicted binary class. Only the encoder layer is used and outputs the result directly. Figure 3.8 shows the architecture of the model used in our research. Nx is the chosen number of Encoder layers in the model. The Linear Projection Dense layer has been used by Pham et al. (2019) in their

research called “Very Deep Self-Attention Networks for End-to-End Speech Recognition” and the results are promising.

Table 3.6 indicates the hyperparameters optimized during the process.

Table 3.6 Hyperparameters of the Transformer used in this work

Hyperparameters	Starting value	Ending value	Step
Number of Encoders	1	10	1
Number of neurons per layer	8	4096	$2^n,\ 3 < n < 12$
Dropout rate	0	0.5	0.05
Number of Attention Heads	1	12	dividers of 12

3.4.4 Attention enhanced LSTM training

The final model is a combination of the Attention properties from the Transformer model and the LSTM network. We do not benefit from parallelism in data processing since the LSTM cells force the sequential treatment of the inputs. On the other hand, the lack of positional encoding enhances the integrity of the original data. Comparing this model to the original LSTM architecture would allow acknowledging the benefits of the Attention mechanism. Figure 3.9 represents the general architecture of this model.

Table 3.7 indicates the hyperparameters optimized during the process.

Table 3.7 Hyperparameters of the Attention enhanced LSTM used in this work

Hyperparameters	Starting value	Ending value	Step
Number of Encoders	1	10	1
Number of neurons per layer	8	1024	$2^n,\ 3 < n < 10$
Dropout rate	0	0.51	0.03
Number of Attention Heads	1	12	1

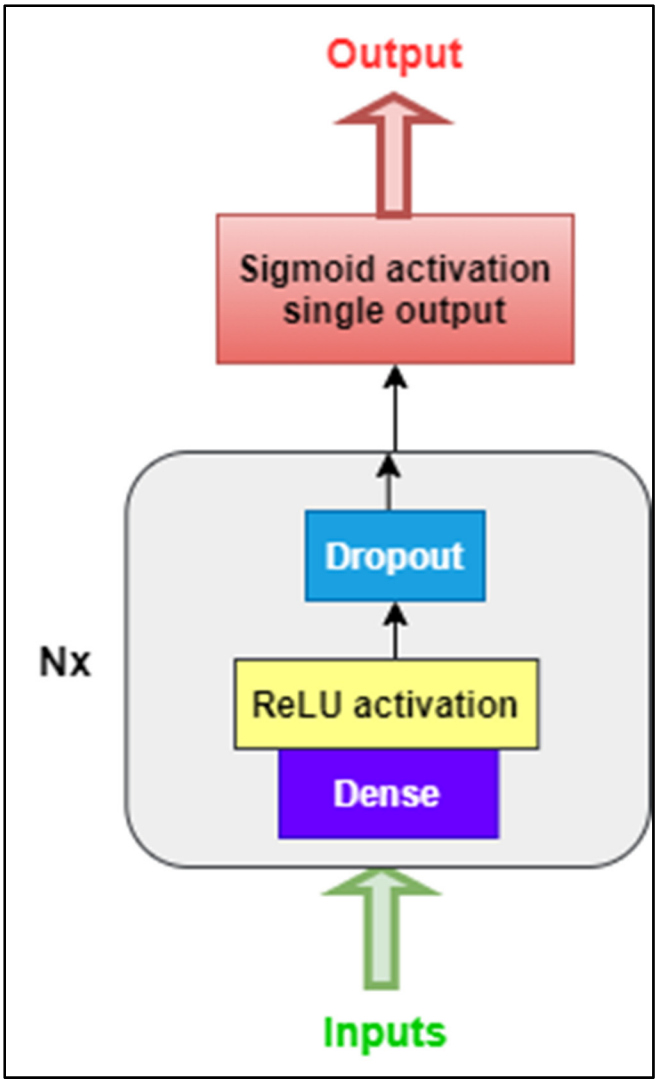


Figure 3.6 Architecture of the MLP network used in this work

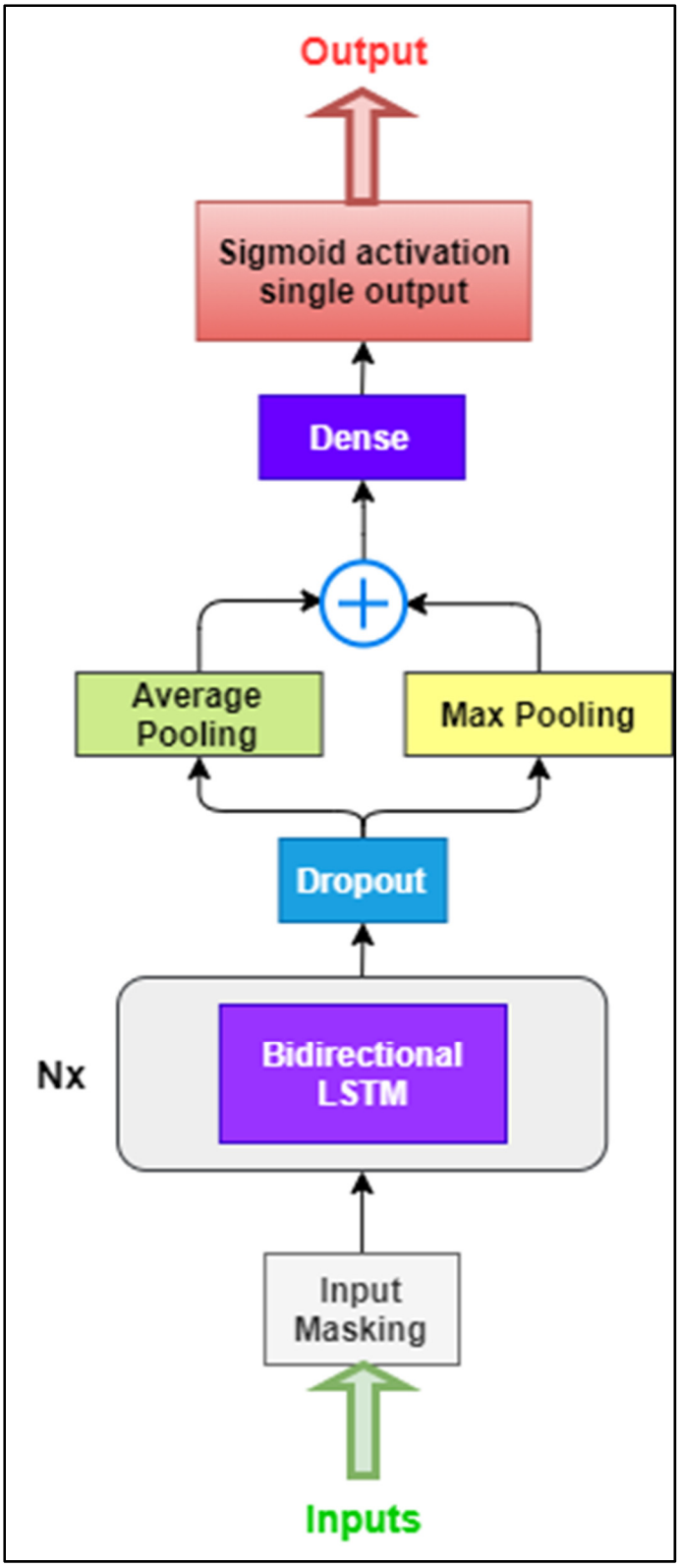


Figure 3.7 Architecture of the LSTM network used in this work

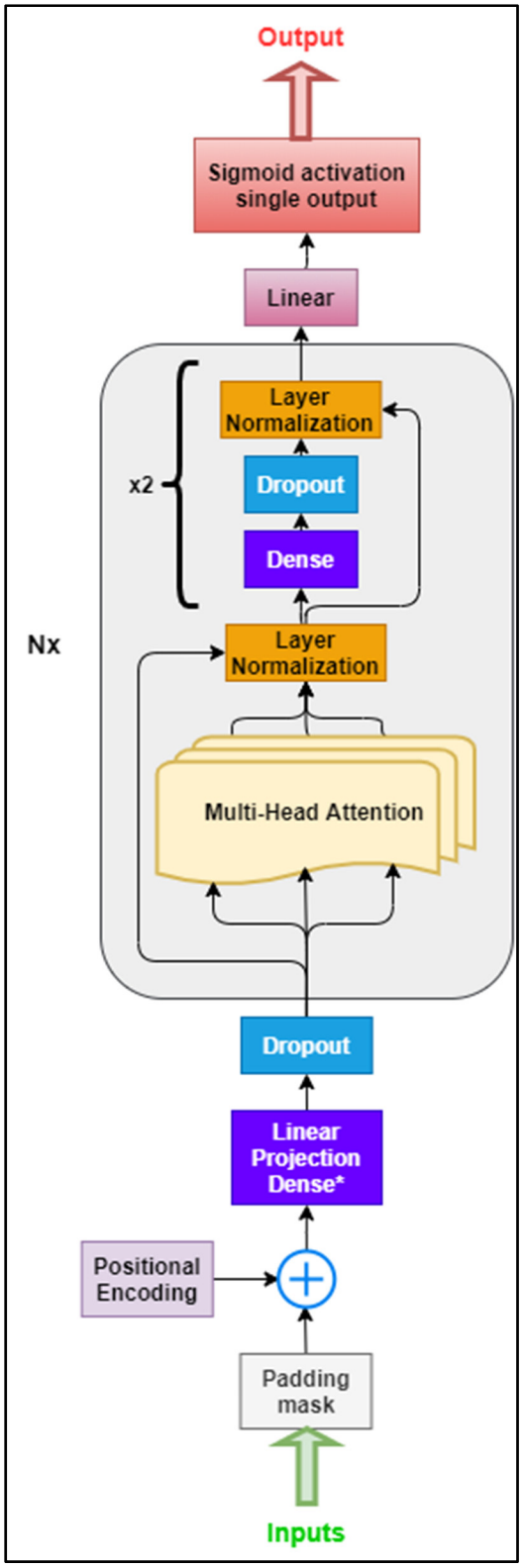


Figure 3.8 Architecture of the Transformer used in this work

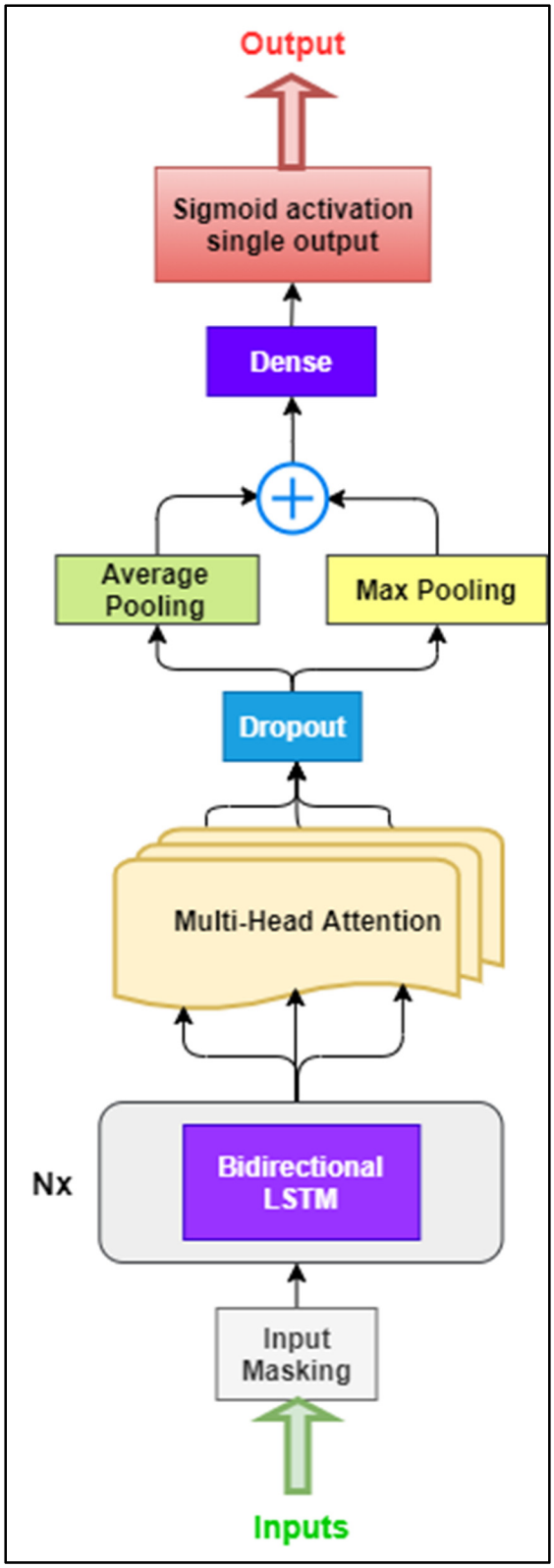


Figure 3.9 Architecture of the Attention enhanced LSTM used in this work

3.5 Training parameters

3.5.1 K-fold cross-validation and Early stopping

K-fold cross-validation with K=5 is implemented to confirm the accuracy value for each hyperparameter optimization. Only the average accuracy of the 5 folds is kept as the result.

Early stopping allows to follow the loss function on validation data and stop the training once there is no improvement for at least 10 epoch. The maximum number of the epoch is 300.

Adam optimizer is used for all our models.

3.5.2 Number of samples per dataset

Healthy infants samples are the most common in the database. To not create bias during the training phase, we randomly chose two times the number of samples in the pathology from the healthy database. We concatenated them to create the new dataset.

For example, there are 39240 healthy samples and 4475 sepsis disease samples. The corresponding dataset contains 13425 expiration sessions (4475 from sepsis and 8950 from healthy).

3.6 Performance evaluation

3.6.1 Accuracy

Binary classification is used in this research. The binary cross-entropy loss function is implemented on all our models. The accuracy determines the number of well-predicted classes relative to the number of total samples. It is a robust measure that gives an overall idea of the network's performance.

3.6.2 Training time

The training time is an important parameter for our research since the idea behind the Transformer network is to efficiently use a modern graphics processing unit (GPU) and parallelize the training session. LSTM network, on the other hand, is a sequential mechanism and cannot efficiently exploit the full potential of the hardware equipment.

We compared the training time to determine if we can make a compromise between accuracy and training time since this factor can become crucial in the future as the number of data exponentially grows in the database.

3.7 Tools

From the preprocessing to the model training, Python language version 3 was used. A list of used libraries is present in Annexe 2. All the code has been developed under the Google Colaboratory platform, enabling it to run separate blocs of codes individually without affecting the other blocs (Garbade, 2021). The sections are also organized like a notebook.

Google allows the user to run the code on their server, which comes preinstalled with several libraries. Our research used the local equipment since it allows a better storage and processing power suited for our needs. All the networks have been training using the following specs on hardware equipment:

- AMD Ryzen 7 2700X Eight-Core Processor.
- 32 GB of DDR4 memory.
- 1 TB of M.2 NVMe SSD storage.
- NVIDIA GeForce RTX 3080.

CHAPITRE 4

PRESENTATION OF THE RESULTS

This chapter presents the results we obtained for each experimentation and then a discussion of their meaning. We believe that the accuracy for each system is not the only important factor in choosing a neural network for a specific application. Sometimes a system may present a higher performance by 2-3% compared to another system, but the training time and required resources might not always justify this improved accuracy by a few points. We focused our research on finding each hyperparameter's impact and utilizing that information to estimate the optimal hyperparameters.

4.1 Simulation results

The simulation results are shown directly in this section for both tests. The first test optimizes the hyperparameters for each neural network, and the second test uses the optimal hyperparameters to obtain the best overall accuracy for all the studied pathologies. The results are then used to compare all the neural networks and the difficulty of identifying a particular disease in a newborn’s cry signal.

4.1.1 Optimal performance

We have gathered the best performances based on the most optimal hyperparameters for each neural network and present them below in tables 4.1 to 4.4. Three performance indicators were gathered to compare the networks. The sensitivity and specificity are beneficial since the number of samples in each class are not equal. In fact, the healthy class of each dataset contains about 66% of the dataset size, while the disease class only contains 33%. The performance indicators are the average of five values obtained with the K-fold cross-validation, making the comparison process more trustworthy. The performance indicators are detailed in this section below as well as their equation.

1. The accuracy: computed by the number of well-predicted classes on the total number of expirations sessions on the test data in each dataset. A return value between 0 and 0.50 included indicates a healthy class. A value between 0.50 excluded and 1 indicates a class with the disease.
2. The sensitivity: represented by the ratio of true positives (infant predicted as having a disease and actually having a disease).
3. The specificity: represented by the ratio of true negatives (infant predicted as healthy and is actually healthy).

The equation of the accuracy is as follows:

$$Accuracy\ (\%) = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

The equation of the sensitivity is as follows:

$$Sensitivity\ (\%) = \frac{TP}{TP + FN} \tag{4.2}$$

The equation of the specificity is as follows:

$$Specificity\ (\%) = \frac{TN}{TN + FP} \tag{4.3}$$

1. True positives (TP) represent the number of newborn cries classified as sick by the neural network, which is correct.
2. True negatives (TN) represent the number of newborn cries classified as healthy by the neural network, which is correct.
3. False positives (TP) represent the number of newborn cries classified as sick by the neural network, which is incorrect.
4. False negatives (FN) represent the number of newborn cries classified as healthy by the neural network, which is incorrect.

Table 4.1 Best performance obtained for the MLP network

Dataset	Accuracy	Specificity	Sensitivity
RDS / Healthy	77.95%	76.07%	75.45%
Sepsis / Healthy	82.25%	86.04%	82.12%
Hyperbilirubinemia / Healthy	80.07%	84.29%	79.12%
Asphyxia / Healthy	82.30%	87.10%	85.76%
All Pathologies / Healthy	76.42%	80.31%	73.34%

Table 4.2 Best performance obtained for the LSTM network

Dataset	Accuracy	Specificity	Sensitivity
RDS / Healthy	85.09%	84.79%	82.86%
Sepsis / Healthy	89.27%	87.71%	89.62%
Hyperbilirubinemia / Healthy	89.55%	85.98%	82.44%
Asphyxia / Healthy	92.22%	90.45%	91.37%
All Pathologies / Healthy	83.43%	86.60%	82.83%

Table 4.3 Best performance obtained for the Transformer network

Dataset	Accuracy	Specificity	Sensitivity
RDS / Healthy	80.12%	78.28%	72.26%
Sepsis / Healthy	85.30%	84.41%	80.94%
Hyperbilirubinemia / Healthy	83.26%	81.77%	73.12%
Asphyxia / Healthy	86.74%	84.05%	87.22%
All Pathologies / Healthy	79.68%	80.89%	77.35%

Table 4.4 Best performance obtained for the Attention LSTM network

Dataset	Accuracy	Specificity	Sensitivity
RDS / Healthy	84.36%	81.70%	81.15%
Sepsis / Healthy	88.79%	88.55%	88.92%
Hyperbilirubinemia / Healthy	87.92%	85.12%	82.12%
Asphyxia / Healthy	91.03%	87.13%	89.51%
All Pathologies / Healthy	84.68%	86.98%	84.02%

4.1.2 Accuracy per hyperparameter

As discussed in chapter 3, we evaluated each hyperparameter individually to observe the impact on training time and network accuracy. Since the models are compared to themselves, we did not test for each individual pathology. The difference in performance would be reflected in each of the dataset's categories.

This section presents the simulation for each of the neural network's classifications for the respiratory distress syndrome (RDS) binary classification. The training time and accuracy are presented for each simulation, along with the sensitivity and specificity.

Although the number of samples is uneven in all datasets, the specificity and the sensitivity results are correlated with the accuracy of the network. In the following section, only the accuracy is analyzed for this reason.

4.1.2.1 MLP network

As presented in figure 4.1, the multilayer perceptron (MLP) network does not show high accuracy for our application. In fact, the best accuracy is 78% for 7 layers of neurons. The training time relatively stays very low for this network, ranging from around 2 to 4 seconds depending on the complexity. Although being a very simple network, it obtains an acceptable performance for a meagre training time compared to the other networks. In our work, the MLP only serves as a reference model for comparison.

The highest score hyperparameters are presented in table 4.5 below.

Table 4.5 Best hyperparameters for MLP

MLP	**Layers**	**Neurons**	**Dropout**	**Learning**
	7	32	0.10	0.26
Accuracy	78.47%	76.91%	75.31%	75.44%

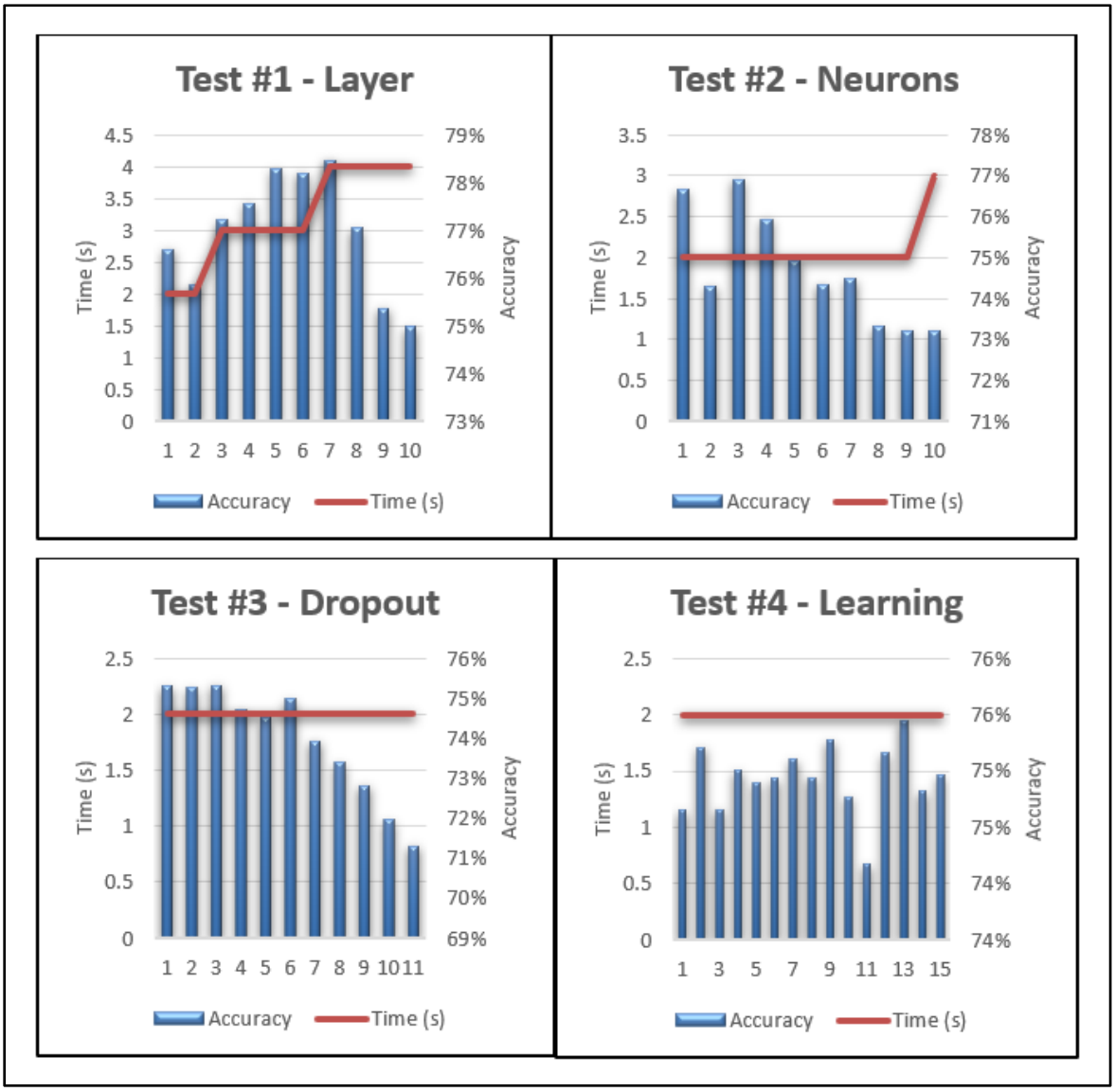


Figure 4.1 Hyperparameters simulation results for MLP

4.1.2.2 LSTM network

The results for the LSTM network are presented in figure 4.2. Although the training time is more than 10 times that of the MLP network, the accuracy is relatively higher for this network. The average performance is 7% higher than the best MLP combination.

1. Training time rapidly increases for each added layer without necessarily increasing the accuracy of the network.

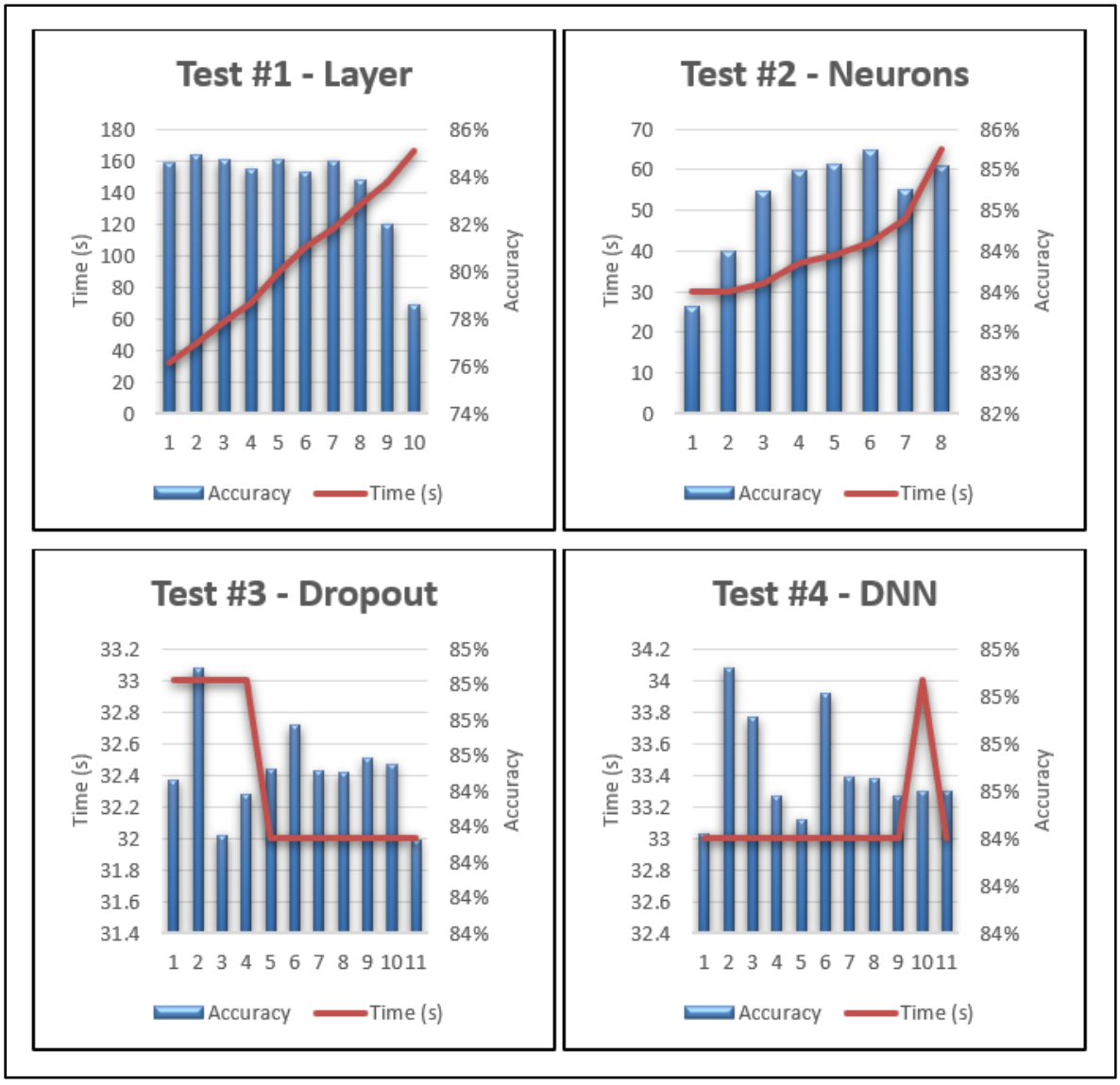


Figure 4.2 Hyperparameters simulation results for LSTM

2. Adding neurons in the hidden layer increases the accuracy by 2%, but the training time also doubles for that increase.
3. There is not much movement when varying the dropout of the network. In fact, the accuracy ranges from 84.15% to 85.09% (for a 0.1 dropout rate). This hyperparameter is a very low contribution for the network. Also, theoretically, having a higher dropout rate should decrease the training time since there are fewer computations with neurons being dropped. Still, we observe only a decrease of around 3% (from 33s to 32s).

4. Adding neurons in the dense neural network (DNN) has no significant impact on accuracy or training time.

For each simulation, our highest scores are presented in table 4.6.

Table 4.6 Best hyperparameters for LSTM

LSTM	**Layers**	**Neurons**	**Dropout**	**DNN**
	2	256	0.25	256
Accuracy	84.89%	85.24%	84.77%	85.18%

4.1.2.3 Transformer network

As illustrated in figure 4.3, the Transformer network do not show promising results for our application. The average accuracy is more than 1% higher than our reference model, but the training time does not justify this increase. Compared to the LSTM, the variation on the hyperparameters greatly impacts the performance of this architecture.

1. Four encoders is the peak of the performance for this network. The accuracy drops beyond this point as there are too many parameters for the number of samples. The training time is linear as we suspected since each layer is an exact copy.
2. The accuracy improves with the number of neurons in the dense layer, reaching a top performance of 78.72% at 512 neurons. The training time only increases by 1s every three power of 2.
3. Although the training time does not change, the accuracy falls the more dropout there is. With 0 dropout, we have the maximum performance.
4. Playing with the number of heads do not significantly change the accuracy. There are not many discriminative features to focus on. The training time increases linearly.

For each simulation, our highest scores are presented in table 4.7.

Table 4.7 Best hyperparameters for the Transformer

TRF	**Encoders**	**Neurons**	**Dropout**	**Heads**
	4	2048	0.00	3
Accuracy	77.14%	79.63%	77.32%	75.94%

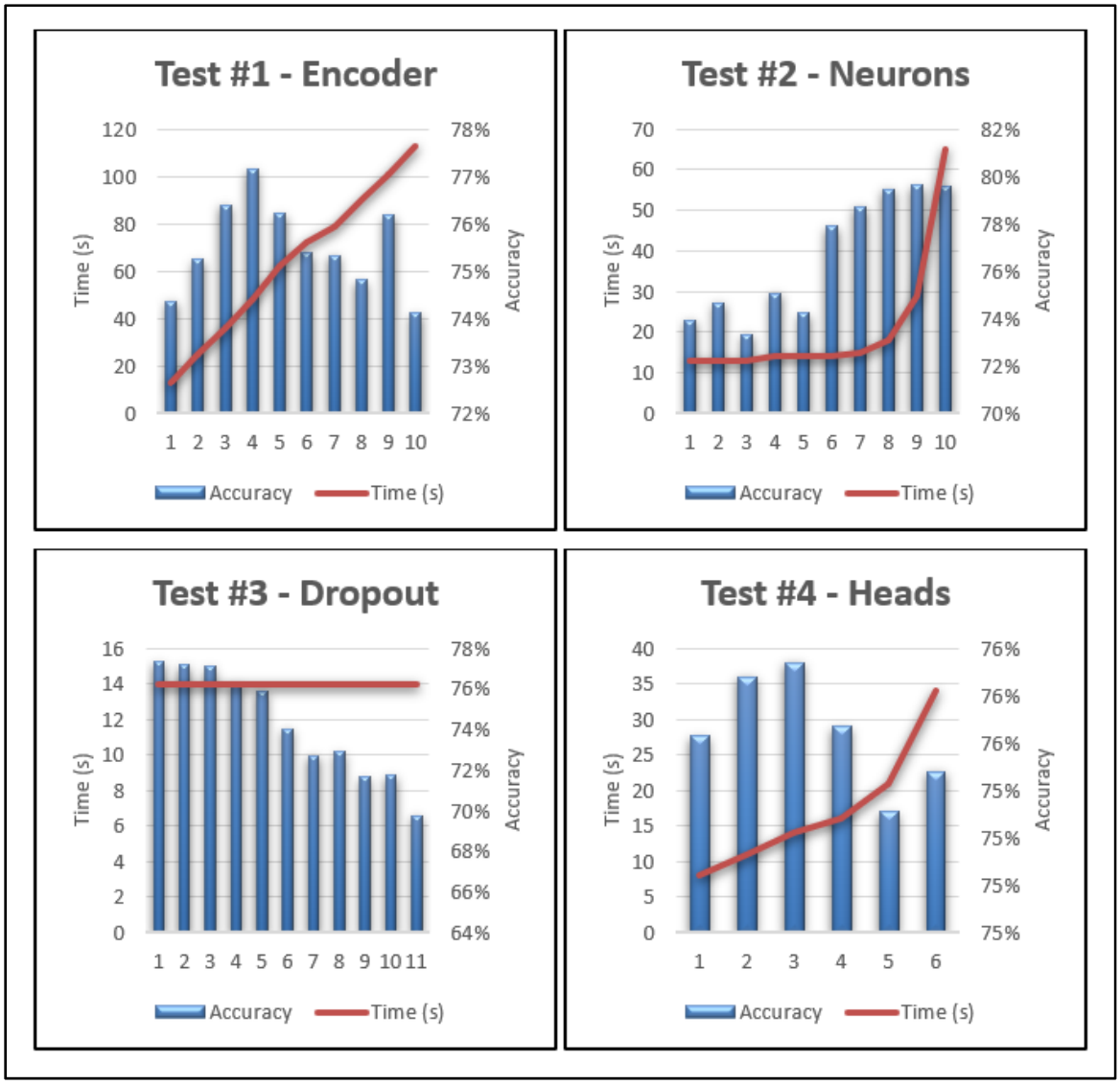


Figure 4.3 Hyperparameters simulation results for Transformer

4.1.2.4 Attention enhanced LSTM network

Figure 4.4 presents the results of the simulation for Attention enhanced LSTM network. Once again, compared to the MLP network, the processing times are larger and the accuracy is better. Adding the attention mechanism increased the training time by more than 20 seconds, which is almost twice the initial time. This increase is not worth, because the accuracy did not increase. In fact, the accuracy decreased by 1% on average for all the simulations. The attention layer proves to be ineffective at this stage of individual hyperparameter simulation.

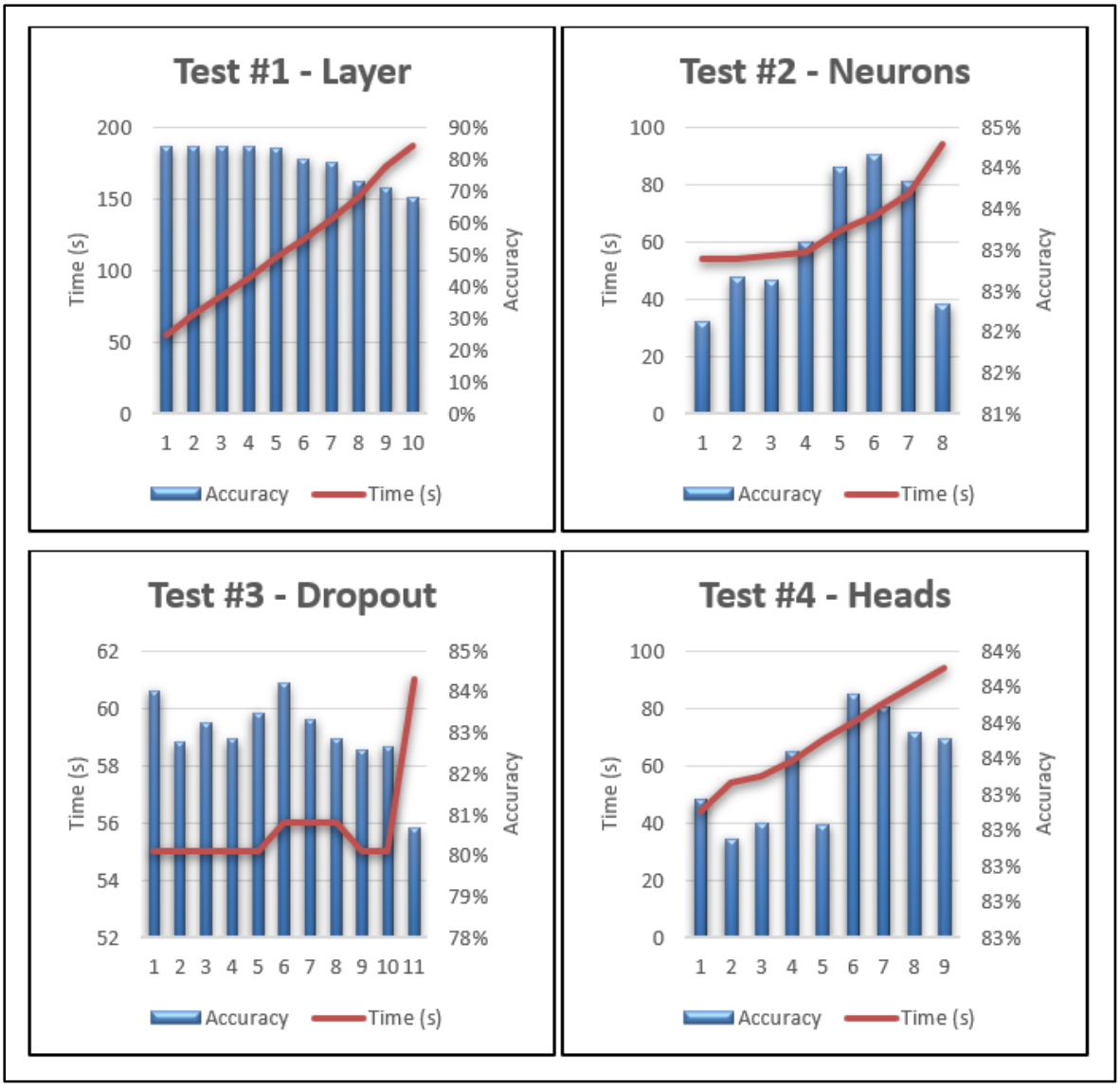


Figure 4.4 Hyperparameters simulation results for Attention enhanced LSTM

1. Increasing the number of encoder layers seems to reduce the accuracy after a certain point while linearly increasing the training time of the network. There is no noticeable difference in accuracy under 5 layers of encoders.
2. After 64 neurons in the hidden layer, the training time increases by a factor of around 25% each time the number of neurons doubles. However, the accuracy reaches its maximum for 256 neurons and decreases after this value.

3. The dropout rate does not decrease the training time, as observed in the LSTM model. Also, for an unknown reason, it increases by 6 seconds between 0.45 and 0.50 dropout values. There is no logical equation for the accuracy, which is maximum for a dropout rate of 25%.
4. As expected, adding Attention heads to the network proportionally increases the training time. The accuracy value slightly changes, only varying by 0.81% in random order. There is no focus point on which the machine can separate and learn discriminative features from the input data.

For each simulation, our highest scores are presented in table 4.8.

Table 4.8 Best hyperparameters for Attention enhanced LSTM

ATT	**Layers**	**Neurons**	**Dropout**	**Heads**
	3	256	0.25	6
Accuracy	83.80%	84.16%	84.19%	83.96%

4.2 Observation on the results

One of the main objective of this research is to compare the performance of the Transformer network and potentially use this state-of-art neural network in the language processing field to solve our problematic. This model uses the full potential of modern hardware to train the network to classify the data efficiently. There is no sequential treatment like the LSTM network, thus avoiding the waiting time before the previous cell outputs a value.

Unfortunately, the Transformer does not show promising results compared to the classical LSTM. The performance for each pathology is relatively lower for the Transformer network. Removing the sequential neurons and entirely depending on the self-attention mechanism did not benefit our application like it did for several other research (Vaswani et al., 2017), (Uszkoreit, 2017). One advantage of the Transformer is that the training time is more than two times lower than the LSTM, meaning that we are benefiting from the processor parallelization of this model.

Although, compared to our reference MLP network, the Transformer offers higher accuracy and can replace the conventional Deep Neural Network if employed alone. It is also an indicator for us that this model has been deployed adequately in our simulation.

However, the Attention enhanced LSTM network proved to be an improvement to the classical recurrent neural network variant. Adding the multi-head Attention with the Encoder-Decoder architecture boosted the accuracy of the network. Even if the Transformer's model did not offer the results we were hoping for, some of its features could be used in our field.

4.2.1 Overfitting

Overfitting on the training data is a problem that occurs more frequently than we think. In our LSTM model, we observed that the training accuracy dangerously approaches 100% in a matter of 30 epochs, while the performance on the test data stagnates around 20% below. Figure 4.5 illustrated this phenomenon during our simulation tests. On the Transformer network, this problem seems unnoticeable. In fact, the training and testing accuracy curve is very close to each other during the whole machine learning process.

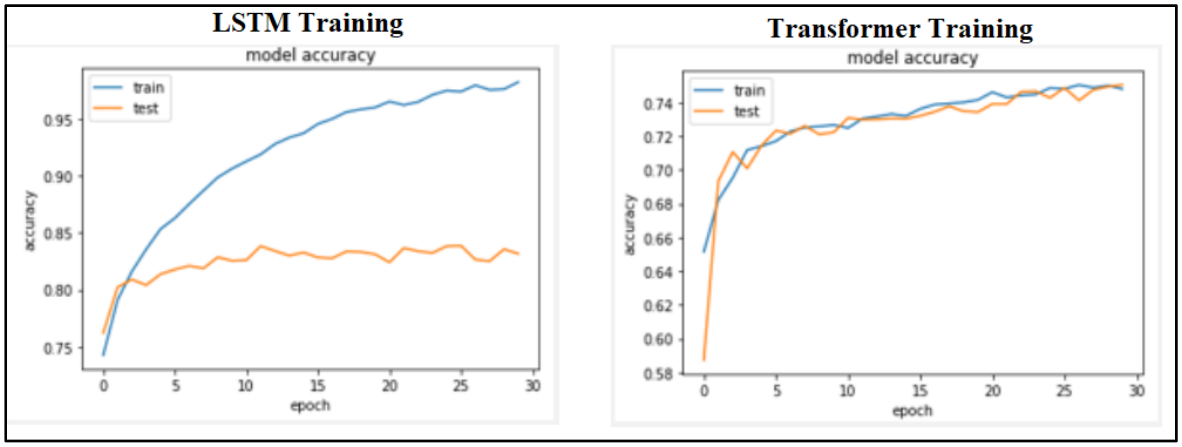


Figure 4.5 Difference between train and data test accuracy for the LSTM and Transformer network on RDS dataset

Both the LSTM and the Transformer have a dropout layer to reduce overfitting, but it is inevitable for the LSTM to learn the training data too much. Our theory is that since the LSTM learns the inputs sequentially, the context vectors from the first samples are learnt quickly, and the network creates a pattern at the very beginning of the expiration. The gates of the LSTM cell might need more data in the dataset to learn to efficiently filter out the information as new inputs approach. This long-term dependency eventually prohibits the network from concentrating on the last sets of input and changing its direction in the classification of unknown data.

The Transformer network does not have sequential data and all inputs of the expiration are introduced at the same time to the model. There is less chance that the model observes only the beginning or any part of the signal individually. It is true that the accuracy is not comparable to the LSTM network, but at least the predicted performance is more representative of the actual output from a classifying device in a real-time environment.

4.2.2 Training time

The training time of a model determines the feasibility of the project. Being able to optimize the hyperparameters with a k-fold validation requires a tremendous amount of time if each epoch consumes several minutes. For example, if one epoch takes 5 minutes to complete, and we have 100 epochs with a 5-fold validation, it would take approximately 42 hours to test only one set of hyperparameters. In a typical neural classification, there are usually more than 10 000 combinations of hyperparameters. It is the reason why it is important to consider both training time and accuracy on a neural network classification.

Our models are complex enough to reach a minute per epoch when the number of parameters is very high. Losing 0.50% accuracy to save hundreds of hours of training is beneficial for both the researcher and the environment. GPU consumes a decent amount of wattage power and heats up at temperatures around 80C.

4.2.3 Number of parameters

Optimizing the accuracy necessarily means minimalizing the loss function of the neural network. To do so, the network parameters are modified gradually as the network learns the inputs (Keim, 2019). Theoretically, the more parameters there are, the more calculations and processing units are required. Table 4.9 shows the number of parameters for each of the studied neural networks in our research.

Table 4.9 Number of trainable parameters in each neural network

	Neural Network			
	MLP	**LSTM**	**Transformer**	**Attention LSTM**
Number of parameters	1 147 265	3 032 577	553 977	4 950 821

As the LSTM model grows, the number of parameters proportionally increases. As we can observe, there are 6 times more parameters in the LSTM network compared to the Transformer network. The LSTM cell has many learnable factors for each gate. The dense layer of the Transformer network contains almost all optimizable parameters. Although there is a significant gap in the number of parameters, the training time is only about the half for the Transformer. Even if the training session is parallel, the computation requirements are much higher in this architecture (Vaswani et al., 2017). The Scaled Dot-Product Attention, as an instance, requires a dot-product multiplication, which can be very demanding for big matrices, even for the latest processing hardware. As a matter of fact, adding the Attention mechanism to the LSTM almost doubles the required training time.

4.3 Comparison of the classifiers

The LSTM, The Transformer and the Attention enhanced LSTM obtained an accuracy above the MLP network, which served as a reference model for our research. Overall, the Transformer architecture performs below the LSTM variant for all the datasets. The training time is, however, lower.

The hyperparameter optimization simulation showed that the performance of the LSTM does not vary drastically as we change the number of layers, the number of neurons, the dropout rate and the number of neurons in the dense layer. For the Transformer network, the accuracy increases by over 4% as we increase the number of neurons. This factor can also be noticed with the Attention layer on the LSTM model.

The Transformer is an interesting path considering its success in the natural language processing (NLP) field (Uszkoreit, 2017), but it is not beneficial for our specific task of pathology classification.

4.4 Comparison of the dataset accuracy

Some diseases are more challenging to identify than others by only using the cries. We can observe that statement in the accuracy difference between the four studied pathologies. Asphyxia obtains the highest accuracy with all neural networks, followed by sepsis, hyperbilirubinemia and respiratory distress syndrome.

Our test cannot conclude that asphyxia contains the most discriminative features to obtain that high accuracy, mostly because the number of data for asphyxia is very limited compared to the other pathologies. There is not enough data to generalize the performance. We believe that the highest count of samples can justify the lower performance on respiratory distress syndrome. In fact, there is a direct correlation between the number of available data in each dataset and the obtained accuracy.

One important conclusion is that our study showed over 1% accuracy improvement on the Attention mechanism compared to the LSTM for the “All pathologies” dataset. We believe it is due to the high number of samples, since the Attention layer requires many data to distinguish the discriminative features properly. Working with large databases may show the full potential of the Attention mechanism used in the Transformer network.

4.5 Comparison with the previous study

We have compared our study with Brault (2018), who worked on the same database and used a neural network for the classification. In his research, the author implemented the multilayer perceptron (MLP) as a reference model, the convolutional neural network (CNN) and the Long-Short Term Memory (LSTM). They used binary classification for seven diseases. Among the list of diseases, the common ones with our research are:

- Respiratory Distress Syndrome (RDS).
- Hyperbilirubinemia.
- Asphyxia.

Table 4.10 compares the best-obtained accuracy for both types of research for the common pathologies and the neural networks (MLP, CNN, LSTM and Transformer).

Table 4.10 Comparison of our work accuracy with Brault

Dataset	**Brault**			**Our research**			
	MLP	CNN	LSTM	MLP	LSTM	Transformer	Attention
RDS	75.28%	79.59%	81.36%	77.95%	85.09%	80.12%	84.36%
Hyperbilirubinemia	82.25%	85.55%	86.62%	80.07%	89.55%	83.26%	87.92%
Asphyxia	95.27%	97.97%	96.76%	82.30%	92.22%	86.74%	91.03%
All Pathologies	73.44%	76.69%	80.50%	76.42%	83.43%	79.68%	84.68%

There are a few differences between our research which explains the difference:

1. F. Brault only uses one hidden layer for both networks. The number of layers is a hyperparameter we optimized in our research.
2. In the LSTM model, we use a pooling layer to reduce the dimensionality of our vector.
3. Also, in the LSTM, our model uses bidirectional LSTM cells, compared to unidirectional in Brault's work.
4. Brault uses 16 MFCC from 20 filter banks, while we use 12 MFCC from 24 filter banks.
5. Brault only took expirations longer than 400ms with framing of 50ms. Our work uses expirations longer than 200ms with framing of 23.22ms.
6. We use overlapping of 50%, while Brault did not implement overlapping.

4.6 Performance reproducibility hyperparameters

Table 4.11 summarizes all the hyperparameters used in the tests to obtain the highest accuracy for all the studied neural networks.

Table 4.11 Highest accuracy hyperparameters for all four neural networks

Neural Network	Hyperparameter	Value
MLP	Number of layers	7
	Number of neurons per layer	32
	Dropout rate	0.1
	Learning rate	0.26
LSTM	Number of layers	2
	Number of neurons per layer	256
	Dropout rate	0.25
	Number of neurons in the dense layer	256
Transformer	Number of encoders	4
	Number of neurons in the dense layer	2048
	Dropout rate	0
	Number of heads in the Multi-Head Attention	3
Attention	Number of layers	3
	Number of neurons per layer	256
	Dropout rate	0.25
	Number of heads in the Multi-Head Attention	6

The LSTM obtains the best performance in our case, while the Transformer is not too much behind. The training time of the latter compensates for the accuracy. However, compared to our reference model, the Transformer is more accurate. Adding the Attention mechanism to the LSTM network does not increase its performance, except for the classification of sick and healthy infants database.

CONCLUSION

Early diagnosis in newborns, especially in developing countries, allows reducing the infant mortality rate by giving a chance to sick children to be treated accordingly before the deadline. Our work consists of analyzing the possible neural networks that can achieve acceptable performance for the medical experts to trust the system along with their expertise presumptions.

The recent Transformer network uses an Encoder-Decoded architecture and the Attention mechanism to focus on discriminative elements of the input data. By adding a positional encoding, the parallel learning algorithm can exploit the full capabilities of modern graphics processing units (GPU) while conserving the dependencies between the data on their rightful chronological, temporal or spatial order.

With the help of the École de Technologie Supérieure database, regrouping children's cries from Canada and Lebanon hospitals, we extracted expiration sessions only from the infant's cries. It was possible due to the previous work consisting of manually labelling the segments in each cry (Abou-Abbas et al., 2015).

The features fed into the neural network are the Mel-Frequency Cepstrum Coefficients (MFCC). After normalizing the data and implementing a pre-emphasis filter, we have gathered 12 coefficients out of 26 filter banks using a framing of 23.22ms and an overlap of 50% on frequencies ranging from 0 Hz to 4000 Hz. The first coefficient has been replaced by the total energy of the expiration session.

We created four models of neural networks to classify using a binary setup for four pathologies. The Multilayer Perceptron (MLP) was purely used as a reference model for comparison. K-fold validation with K=5 has been used to validate the performance by averaging five results on different samples as test data.

Our work compared the Transformer capabilities to the most popular variant of the recurrent neural network (RNN), the Long-Short Term Memory (LSTM). Unfortunately, we did not achieve the desired performance on the Transformer network as it is 5% lower than its competitor. The training time, however, is more than two times lower than the LSTM. The machine training parallelization is helping to lower the time compared to the sequential training of the LSTM.

Adding the Attention mechanism to an existing LSTM network did not improve the results; it only took more processing power and time. Our theory is that there are not enough distinctive features on the input data for the Attention layer to focus on different aspects of the MFCC. Also, the number of samples is very limited for the neural network to adequately learn the features and learn to pay attention to some crucial elements. We can observe this phenomenon on the dataset containing all the pathologies. We had an improvement of over 1% in performance when classifying sick and healthy infants. We may not have gathered enough data for detecting a specific pathology in an infant, but our research shows that we could efficiently distinguish a healthy infant from one having a disease using the Attention mechanism.

However, the lower results do not mean that this path should be entirely closed. In fact, the Transformer network is relatively new in the timeline, and the community does not fully understand its behaviour. It is a powerful tool for processing units when correctly implemented with the right set of features. In the next section, we discuss more the possible ways to test and increase the network's performance.

RECOMMENDATIONS

Many simulations can be done to unlock the Transformer's potential fully:

1. Apply different factors for MFCC extraction, such as a different framing window or the number of coefficients per frame. The number of filter banks can be varied also.

2. Work on other diseases and conditions. Since the Attention mechanism tries to focus on distinctive elements of the input data, there might be a condition that contains enough factors to help the Transformer with multi-heads.

3. We used time-based input embedding to integrate sequential information. There is also the possibility to use spatial input embedding, associating each input as a randomly initialized multidimensional vector, just like in language translation. The training algorithm automatically reduces the distance between those vectors with inputs containing similarities.

4. Include several other time-based or frequency-based features to the neural network, such as the fundamental frequency, the intensity of the signal, the Bark Frequency Cepstral Coefficients (BFCC), Zero-crossing rate (ZCR) and the rhythm. It will also provide the multi-head Attention mechanism better distinguishable features.

5. Combine other types of neural networks, such as the convolutional neural network (CNN), with the Attention layer. Perhaps it may result in a different performance.

With adequate time and resources, the Transformer network can provide desirable results. Our field of application is very close to the domain in which this model acquired state-of-art performance. The newer Transformer-XL network can also be explored, which uses the recurrence mechanism and relative positional encoding.

ANNEXE I

LIST OF DISEASES IN THE DATABASE

The following table I-1 presents all the diseases and conditions present in École de Technologie Supérieure Database collected from hospitals in Canada and Lebanon.

Table-A I-1 List of diseases and conditions of infants in the database

Samples	Disease / Condition
806	Healthy
214	Respiratory Distress
90	Sepsis
58	Grunting
35	Jaundice
33	Fever
23	Cyanosis
23	Vomit
21	Hyperbilirubinemia
18	Retraction
17	Tachypnea
17	Dyspnea
17	Broncholities
15	Meconium Aspiration Syndrom
14	Pneumonitis
13	Hypoglycemia
11	Diarrhoea
11	Hypotonia
10	Asphyxia
10	Apnea
8	Peritonitis
7	CIV-Communication Interventriculaire
6	Distension
6	Aspiration
6	Meningitis
5	Congenital Heart Disease
5	Cleft lip and palate
4	Gastrochisis
4	Thrombose

Table-A I-2 List of diseases and conditions of infants in the database (continuation)

Samples	Disease / Condition
4	Shoulder Fracture
3	Duodenal Atreasia
3	Down Syndrom
3	Dismorphism
3	Spinal Bifida
3	Kidney Failure
3	Hypothermia
3	Seizure
3	Intra Uterine Growth Retardation
3	Ankyloglossia
3	Tetralogy of Fallot
3	HTAP-Hypertension de l'artère pulmonaire
3	Complex Cardio
3	Convulsion
3	Cerebral Hemorrhage
3	Diaphragmatic Eventration
3	Choanal Atresia
3	Myelomeningocele
2	Bronchopulmonary Dysplasia
2	CIA-Communication Interauriculaire
2	Clubfoot
2	Polymalformation Syndrome
1	Trisomy
1	Potbelly
1	Chorioamnionitis
1	Nasal Septum Deviation
1	Ductus Arteriosus
1	HIV GR1
1	Situs Inversus
1	Intestinal Malrotation
1	Shone's Complex
1	Intestinal Perforation
1580	**Samples (61 diseases)**

ANNEXE II

LIST OF PYTHON USED PYTHON LIBRARIES

Many libraries helped speed up the process of our research:

System:

1. Os
2. Random
3. Sys
4. Time

Data manipulation:

1. Wave
2. CSV
3. Numpy
4. Pandas
5. TQDM
6. MFCC and logfbank from python_speech_features

Classification:

1. Shuffle from sklearn.utils
2. Tensorflow.keras: models, layers, callbacks, backend and initializers
3. KFold from sklearn.model_selection

Visualization

1. Librosa
2. Pyplot from matplotlib

ANNEXE III

EXAMPLE OF MFCC SPECTROGRAM

Figure 4-1 illustrates an example of a 2-dimensional matrix of the MFCC values extracted from the database. The name of the file has been removed for confidentiality reasons.

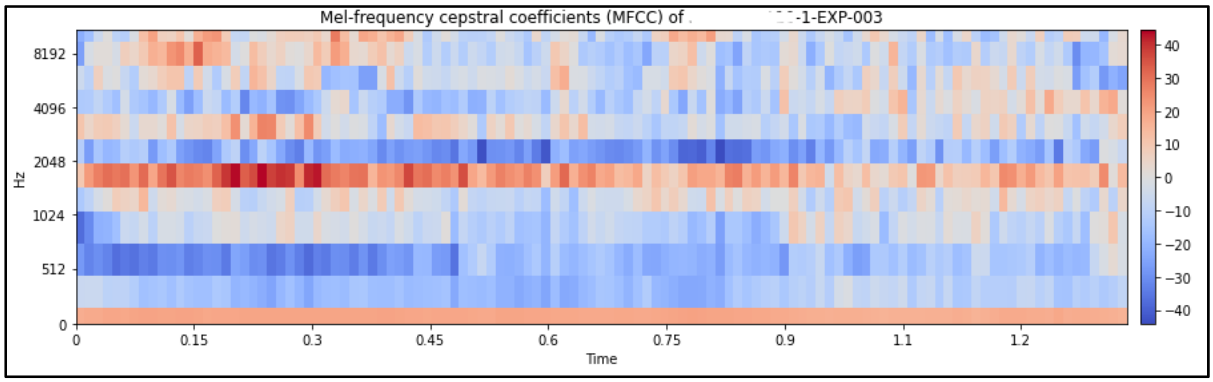


Figure-A III-1 MFCC features spectrogram of a random expiration session

LIST OF REFERENCES

Abou-Abbas, L., Alaei, H. F., & Tadj, C. (2015, 3-6 May 2015). *Segmentation of voiced newborns' cry sounds using wavelet packet based features.* Paper presented at the 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE).

Adaloglou, N. (2020). How Attention works in Deep Learning: understanding the attention mechanism in sequence models. Retrieved from https://theaisummer.com/attention/

Affendi, A., & Yusoff, M. (2019). Review of anomalous sound event detection approaches. *IAES International Journal of Artificial Intelligence (IJ-AI), 8*, 264. doi:10.11591/ijai.v8.i3.pp264-269

Ahire, J. B. (2020). Demystifying the XOR problem. Retrieved from https://dev.to/jbahire/demystifying-the-xor-problem-1blk

Anders, F., Hlawitschka, M., & Fuchs, M. (2020). Automatic classification of infant vocalization sequences with convolutional neural networks. *Speech Communication, 119*, 36-45. doi:https://doi.org/10.1016/j.specom.2020.03.003

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks, 138*, 14-32. doi:https://doi.org/10.1016/j.neunet.2021.01.026

Badreldine, O. M., Elbeheiry, N. A., Haroon, A. N. M., ElShehaby, S., & Marzook, E. M. (2018, 29-30 Dec. 2018). *Automatic Diagnosis of Asphyxia Infant Cry Signals Using Wavelet Based Mel Frequency Cepstrum Features.* Paper presented at the 2018 14th International Computer Engineering Conference (ICENCO).

Bano, S., & RaviKumar, K. M. (2015, 25-27 Feb. 2015). *Decoding baby talk: A novel approach for normal infant cry signal classification.* Paper presented at the 2015 International Conference on Soft-Computing and Networks Security (ICSNS).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157-166. doi:10.1109/72.279181

Bğnicğ, I., Cucu, H., Buzo, A., Burileanu, D., & Burileanu, C. (2016, 27-29 June 2016). *Baby cry recognition in real-world conditions.* Paper presented at the 2016 39th International Conference on Telecommunications and Signal Processing (TSP).

Bhandari, A. (2020). Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization. Retrieved from

https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Brault, F. (2018). *Dépistage de pathologies par analyse de cris néonataux à l'aide de réseaux de neurones.* (Master's degree). École de Technologie Supérieure.

Chang, C., Hsiao, Y., & Chen, S. (2015, 2-4 Sept. 2015). *Application of Incremental SVM Learning for Infant Cries Recognition.* Paper presented at the 2015 18th International Conference on Network-Based Information Systems.

Chang, C., & Li, J. (2016, 27-29 May 2016). *Application of deep learning for recognizing infant cries.* Paper presented at the 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW).

Chauhan, P. M., & Desai, N. P. (2014, 6-8 March 2014). *Mel Frequency Cepstral Coefficients (MFCC) based speaker identification in noisy environment using wiener filter.* Paper presented at the 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE).

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, oct). *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*, Doha, Qatar.

Delua, J. (2021). Supervised vs. Unsupervised Learning: What's the Difference? *IBM*. Retrieved from https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

Dewi, S. P., Prasasti, A. L., & Irawan, B. (2019, 16-18 July 2019). *The Study of Baby Crying Analysis Using MFCC and LFCC in Different Classification Methods.* Paper presented at the 2019 IEEE International Conference on Signals and Systems (ICSigSys).

Exxact. (2019). 5 Types of LSTM Recurrent Neural Networks and What to Do With Them. Retrieved from https://www.exxactcorp.com/blog/Deep-Learning/5-types-of-lstm-recurrent-neural-networks-and-what-to-do-with-them

Farsaie Alaie, H. (2015). *Diagnosis of diseases in newborn infants by analysis of cry signals.* (Doctoral thesis). École de technologie supérieure, Montreal.

Farsaie Alaie, H., Abou-Abbas, L., & Tadj, C. (2016). Cry-based infant pathology classification using GMMs. *Speech Communication, 77*, 28-52. doi:https://doi.org/10.1016/j.specom.2015.12.001

Felipe, G. Z., Aguiar, R. L., Costa, Y. M. G., Silla, C. N., Brahnam, S., Nanni, L., & McMurtrey, S. (2019, 5-7 June 2019). *Identification of Infants' Cry Motivation Using Spectrograms.* Paper presented at the 2019 International Conference on Systems, Signals and Image Processing (IWSSIP).

Ferretti, D., Severini, M., Principi, E., Cenci, A., & Squartini, S. (2018, 3-7 Sept. 2018). *Infant Cry Detection in Adverse Acoustic Environments by Using Deep Neural Networks.* Paper presented at the 2018 26th European Signal Processing Conference (EUSIPCO).

Garbade, D. M. J. (2021). What is Google Colab? Retrieved from https://blog.education-ecosystem.com/what-is-google-colab/

Geller, S. (2019). Normalization vs Standardization — Quantitative analysis. Retrieved from https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf

Gers, F., & Schmidhuber, J. (2000). Recurrent nets that time and count. *Proceedings of the International Joint Conference on Neural Networks, 3*, 189-194 vol.183. doi:10.1109/IJCNN.2000.861302

Glorot, X., Bordes, A., & Bengio, Y. (2011). *Deep Sparse Rectifier Neural Networks*. Paper presented at the Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research. http://proceedings.mlr.press/v15/glorot11a.html

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 38*. doi:10.1109/ICASSP.2013.6638947

Gu, G., Shen, X., & Xu, P. (2018, 25-27 May 2018). *A Set of DSP System to Detect Baby Crying.* Paper presented at the 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation, 9*, 1735-1780. doi:10.1162/neco.1997.9.8.1735

Jam, M. M., & Sadjedi, H. (2009, 2-4 Dec. 2009). *Identification of hearing disorder by multi-band entropy cepstrum extraction from infant's cry.* Paper presented at the 2009 International Conference on Biomedical and Pharmaceutical Engineering.

Jeyaraman, S., Muthusamy, H., Khairunizam, W., Jeyaraman, S., Nadarajaw, T., Yaacob, S., & Nisha, S. (2018). A review: survey on automatic infant cry analysis and classification. *Health and Technology, 8*(5), 391-404. doi:10.1007/s12553-018-0243-5

Ji, C., Mudiyanselage, T. B., Gao, Y., & Pan, Y. (2021). A review of infant cry analysis and classification. *EURASIP Journal on Audio, Speech, and Music Processing, 2021*(1), 8. doi:10.1186/s13636-021-00197-5

Ji, C., Xiao, X., Basodi, S., & Pan, Y. (2019, 14-17 July 2019). *Deep Learning for Asphyxiated Infant Cry Classification Based on Acoustic Features and Weighted Prosodic Features.* Paper presented at the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData).

Keim, R. (2019). How to Train a Multilayer Perceptron Neural Network. Retrieved from https://www.allaboutcircuits.com/technical-articles/how-to-train-a-multilayer-perceptron-neural-network/

Kelly, A. C., & Gobl, C. (2011). *The Effects of Windowing on the Calculation of MFCCs for Different Types of Speech Sounds.* Paper presented at the Advances in Nonlinear Speech Processing, Berlin, Heidelberg.

Kheddache, Y. (2014). *Caractérisation des cris des nourrissons en vue du diagnostic précoce de différentes pathologies.* (Doctoral thesis). École de Technologie Supérieure,

Kheddache, Y., & Tadj, C. (2019). Identification of diseases in newborns using advanced acoustic features of cry signals. *Biomedical Signal Processing and Control, 50*, 35-44. doi:https://doi.org/10.1016/j.bspc.2019.01.010

Lavner, Y., Cohen, R., Ruinskiy, D., & Ijzerman, H. (2016, 16-18 Nov. 2016). *Baby cry detection in domestic environment using deep learning.* Paper presented at the 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE).

Lind, J., Vuorenkoski, V., Rosberg, G., Partanen, T. J., & Wasz-Höckert, O. (1970). Spectographic analysis of vocal response to pain stimuli in infants with Down's syndrome. *Dev Med Child Neurol, 12*(4), 478-486. doi:10.1111/j.1469-8749.1970.tb01943.x

Liu, L., Li, Y., & Kuo, K. (2018, 23-25 March 2018). *Infant cry signal detection, pattern extraction and recognition.* Paper presented at the 2018 International Conference on Information and Computer Technologies (ICICT).

Lyons, J. (2013). Welcome to python_speech_features's documentation! Retrieved from https://python-speech-features.readthedocs.io/en/latest/

Maghfira, T. N., Basaruddin, T., & Krisnadhi, A. (2020). Infant cry classification using CNN – RNN. *Journal of Physics: Conference Series, 1528*, 012019. doi:10.1088/1742-6596/1528/1/012019

Maklin, C. (2019). Fast Fourier Transform. Retrieved January 15, 2021, from Towards Data Science https://towardsdatascience.com/fast-fourier-transform-937926e591cb

Manikanta, K., Soman, K. P., & Manikandan, M. S. (2019, 20-21 Dec. 2019). *Deep Learning Based Effective Baby Crying Recognition Method under Indoor Background Sound Environments.* Paper presented at the 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS).

Medhat, F., Chesmore, D., & Robinson, J. (2020). Masked Conditional Neural Networks for sound classification. *Applied Soft Computing, 90*, 106073. doi:https://doi.org/10.1016/j.asoc.2020.106073

Menzies, T., Kocagüneli, E., Minku, L., Peters, F., & Turhan, B. (2015). Chapter 24 - Using Goals in Model-Based Reasoning. In T. Menzies, E. Kocagüneli, L. Minku, F. Peters, & B. Turhan (Eds.), *Sharing Data and Models in Software Engineering* (pp. 321-353). Boston: Morgan Kaufmann.

Moharir, M., Sachin, M. U., Nagaraj, R., Samiksha, M., & Rao, S. (2017, 7-9 April 2017). *Identification of asphyxia in newborns using gpu for deep learning.* Paper presented at the 2017 2nd International Conference for Convergence in Technology (I2CT).

Molaeezadeh, S. F., Salarian, M., & Moradi, M. H. (2012, 2-3 May 2012). *Type-2 fuzzy pattern matching for classifying hunger and pain cries of healthy full-term infants.* Paper presented at the The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012).

Morgan, B. G. N. (2002). Speech and Audio Signal Processing. In (Vol. Part- IV, pp. 189-203).

Muda, L., Begam, M., & Elamvazuthi, I. (2010). Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *J Comput, 2*.

Mukhopadhyay, J., Saha, B., Majumdar, B., Majumdar, A. K., Gorain, S., Arya, B. K., . . . Singh, A. (2013, 28-30 March 2013). *An evaluation of human perception for neonatal cry using a database of cry and underlying cause.* Paper presented at the 2013 Indian Conference on Medical Informatics and Telemedicine (ICMIT).

O'Shaughnessy, D. (1987). *Speech Communication Human and Machine*. New York: Addison-Wesley.

Oller, D. K., Caskey, M., Yoo, H., Bene, E. R., Jhang, Y., Lee, C.-C., . . . Vohr, B. (2019). Preterm and full term infant vocalization and the origin of language. *Scientific Reports, 9*(1), 14734. doi:10.1038/s41598-019-51352-0

Phi, M. (2018). Illustrated Guide to LSTM's and GRU's: A step by step explanation. Retrieved from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Radford, A., Sutskever, I., Józefowicz, R., Clark, J., & Brockman, G. (2017). Unsupervised Sentiment Neuron. Open AI. Retrieved from https://openai.com/blog/unsupervised-sentiment-neuron/

Raschka, S. (2018). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. arXiv:1811.12808. Retrieved from https://ui.adsabs.harvard.edu/abs/2018arXiv181112808R

Reyes-Galaviz, O. F., Tirado, E. A., & Reyes-Garcia, C. A. (2004, 2004//). *Classification of Infant Crying to Identify Pathologies in Recently Born Babies with ANFIS.* Paper presented at the Computers Helping People with Special Needs, Berlin, Heidelberg.

Rosales-Pérez, A., Reyes-García, C. A., Gonzalez, J. A., Reyes-Galaviz, O. F., Escalante, H. J., & Orlandi, S. (2015). Classifying infant cry patterns by the Genetic Selection of a Fuzzy Model. *Biomedical Signal Processing and Control, 17*, 38-46. doi:https://doi.org/10.1016/j.bspc.2014.10.002

Rožanc, I., & Mernik, M. (2021). Chapter Three - The screening phase in systematic reviews: Can we speed up the process? In A. R. Hurson (Ed.), *Advances in Computers* (Vol. 123, pp. 115-191): Elsevier.

Salehian Matikolaie, F., & Tadj, C. (2020). On the use of long-term features in a newborn cry diagnostic system. *Biomedical Signal Processing and Control, 59*, 101889. doi:https://doi.org/10.1016/j.bspc.2020.101889

Sangeetha, J., Hariprasad, R., & Subhiksha, S. (2021). Chapter 8 - Analysis of machine learning algorithms for audio event classification using Mel-frequency cepstral coefficients. In N. Dey (Ed.), *Applied Speech Processing* (pp. 175-189): Academic Press.

Saraswathy, J., Hariharan, M., Yaacob, S., & Khairunizam, W. (2012, 27-28 Feb. 2012). *Automatic classification of infant cry: A review.* Paper presented at the 2012 International Conference on Biomedical Engineering (ICoBE).

Sharma, K., Gupta, C., & Gupta, S. (2019, 6-8 July 2019). *Infant Weeping Calls Decoder using Statistical Feature Extraction and Gaussian Mixture Models.* Paper presented at the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res., 15*(1), 1929–1958.

Staudemeyer, R., & Morris, E. (2019). *Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks*. Schmalkalden University of Applied Sciences. Germany.

University of Central Florida. (2021). Python Lists vs. Numpy Arrays - What is the difference? Retrieved 18/06/2021 https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference

Uszkoreit, J. (2017). Transformer: A Novel Neural Network Architecture for Language Understanding. *Google AI Blog*. Retrieved from https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A Review on the Long Short-Term Memory Model. *Artificial Intelligence Review, 53*. doi:10.1007/s10462-020-09838-1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *arXiv e-prints*, arXiv:1706.03762. Retrieved from https://ui.adsabs.harvard.edu/abs/2017arXiv170603762V

Wahid, N. S. A., Saad, P., & M., H. (2016). Automatic infant cry pattern classification for a multiclass problem. *Journal of Telecommunication, Electronic and Computer Engineering, 8*, 45-52.

Wang, T., Chen, P., Amaral, K., & Qiang, J. (2016). *An Experimental Study of LSTM Encoder-Decoder Model for Text Simplification*. University of Massachussets. Boston.

Weiss, G., Goldberg, Y., & Yahav, E. (2018). On the Practical Computational Power of Finite Precision RNNs for Language Recognition. *CoRR, abs/1805.04908*. Retrieved from http://arxiv.org/abs/1805.04908

World Health Organization. (2020). *Newborns: improving survival and well-being*. Retrieved from https://www.who.int/news-room/fact-sheets/detail/newborns-reducing-mortality

Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., . . . Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

Yamamoto, S., Yoshitomi, Y., Tabuse, M., Kushida, K., & Asada, T. (2013). Recognition of a Baby's Emotional Cry towards Robotics Baby Caregiver. *International Journal of Advanced Robotic Systems, 10*(2), 86. doi:10.5772/55406

Yoshifumi, O., Kentarou, F., & Tomomasa, N. (2011). Iterative Forward Selection Method Based on Cross-validation Approach and Its Application to Infant Cry Classification. *Lecture Notes in Engineering and Computer Science, 1*.

Zabidi, A., Mansor, W., Lee, Y. K., Yassin, I. M., & Sahak, R. (2011, 4-6 March 2011). *Binary Particle Swarm Optimization for selection of features in the recognition of infants cries with asphyxia.* Paper presented at the 2011 IEEE 7th International Colloquium on Signal Processing and its Applications.

Zeifman, D. M., & St James-Roberts, I. (2017). Parenting the crying infant. *Current Opinion in Psychology, 15*, 149-154. doi:https://doi.org/10.1016/j.copsyc.2017.02.009