Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Master's Degree in Artificial Intelligence

Master Thesis

Towards an Efficient and Accurate Speech Enhancement by a Comprehensive Ablation Study

Author: Lidia Abad Azcutia
Tutor: Bojan Mihaljevic y Jorge Luque Serrano

Madrid, July 2024

This Master's Thesis has been deposited at the ETSI Informáticos de la Universidad Politécnica de Madrid for its defense.

Master Thesis
Master's Degree in Artificial Intelligence

Title: Towards an Efficient and Accurate Speech Enhancement by a Comprehensive Ablation Study

July 2024

Author: Lidia Abad Azcutia
Tutor: Bojan Mihaljevic y Jorge Luque Serrano
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Acknowledgements

I would like to express my sincere gratitude to those that have helped me through the development of this thesis. I would mainly like to thank Fernando López Gavilánez. Thanks to him, to his valuable feedback and contributions, I have learned a lot about deep learning for speech enhancement. I am deeply grateful for his guidance. It is largely thanks to him that this work has been developed.

Resumen

Las tareas de mejora del habla son métodos que mejoran la calidad y la inteligibilidad de las señales de audio ruidosas. Los modelos de mejora del habla se entrenan con el objetivo de distinguir entre señales limpias y ruidos, para después eliminar dichos ruidos de las señales originales. Las señales nítidas proporcionan una comunicación más sencilla y agradable. Adaptar estos modelos a tareas cotidianas no está exento de dificultades. Los modelos con los mejores resultados en cuanto a mejora del habla, no son lo suficientemente ligeros para ser utilizados en dispositivos con recursos limitados. Además, muchos modelos requieren excesivo tiempo para procesar audios ruidosos, lo que impide su uso en tiempo real.

Recientemente, se ha prestado especial atención a la eficiencia de los modelos de mejora del habla. Para popularizar el uso de este tipo de modelos, estos deben poderse ejecutar en dispositivos de capacidad reducida. Con ello se mejoran las prestaciones en relación a la privacidad de los usuarios. Para reducir la latencia, tamaño y operaciones de los modelos, se han desarrollado técnicas de aprendizaje profundo. Sin embargo, debido a la compensación entre eficiencia y rendimiento, sigue siendo un reto obtener modelos eficientes y con buenos resultados a nivel de mejora del habla.

En este proyecto se han diseñado tres modelos eficaces de mejora del habla. Los modelos desarrollados son modelos de tiempo-frecuencia, lo que significa que se centran tanto en las características temporales como en las frecuenciales. El dominio de la frecuencia se divide en la magnitud y la fase de la señal de audio. Los tres modelos son redes generativas adversarias con una estructura codificador-decodificador en el generador y mecanismos de atención en el espacio latente.

De los tres modelos, sólo uno de ellos, denominado MiniGAN, obtiene resultados competitivos. Estos resultados incluyen métricas objetivas y subjetivas utilizadas habitualmente en la mejora del habla. Dichas métricas hacen referencia a la calidad e inteligibilidad de las señales de audio mejoradas. Los tres modelos son altamente eficientes, tanto en tamaño, como en número de parámetros y operaciones. Además, algunos de ellos pueden ser utilizados en aplicaciones en tiempo real. Este proyecto contribuye al estado del arte en la mejora del habla, ya que propone modelos eficientes y competitivos. El trabajo futuro debería centrarse en mejorar las métricas de calidad e inteligibilidad obtenidas a través de los modelos.

Abstract

Speech enhancement tasks are methods that improve the quality and intelligibility of noisy audio signals. To that end, speech enhancement models are trained to distinguish between clean signals and noises, and then remove the noises from the original signals. Enhanced signals provide easier and more pleasant communication. Adapting these models to everyday tasks is not without difficulties. The models with the best speech enhancement results are not small enough to be used in devices with limited resources. On top of that, many models require a long time to process noisy audio, which prevents their use in real time.

Recently, a strong focus on how to make speech enhancement models efficient has been developed. In order to extend the use of the models, they must be able to run on low-resourced devices. This improves performance in terms of user privacy. To reduce model latency, size and number of operations, deep learning techniques have been developed. However, due to the trade-off between efficiency and performance, it is still a challenge to obtain efficient models with good speech enhancement results.

In this project, three efficient speech enhancement models are designed. The models developed are time-frequency models, which means that they focus both on time and frequency characteristics. The frequency domain is divided into the magnitude and phase of the signal. The three models are generative adversarial networks with an encoder-decoder structure in the generator and attention mechanisms in the latent space.

Of the three models, only one of them, called MiniGAN, obtains competitive results. These results comprise the objective and subjective metrics commonly used in speech enhancement. These metrics refer to the quality and intelligibility of the audio signals enhanced through the neural networks. All three models are highly efficient, both in size and in number of parameters and operations. In addition, some of them can be used in real-time applications. This project contributes to the state of the art in speech enhancement, as it proposes efficient and competitive models. Future work should focus on improving the quality and intelligibility metrics obtained from the models.

Contents

List of Figures

List of Tables

List of Abbreviations

ASR	Automatic Speech Recognition
BiLSTM	Bidirectional Long Short Time Memory
BRNN	Bidirectional Recurrent Neural Network
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DFT	Discrete Fourier Transform
DNS	Deep Noise Suppression
FFNN	Feed Forward Neural Network
FLOPS	Floating point Operations per Second
GAN	Generative Adversarial Network
GCRN	Gated Convolutional Recurrent Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Units
IQR	Inter Quartile Range
ISTFT	Inverse Short Time Fourier Transform
ITU	International Telecommunication Union
LLR	Log-likelihood Ratio
LSD	Logarithmic Spectral Distance
LSTM	Long Short Time Memory
MAE	Mean Absolute Error
MMSE	Minimum Mean Square Error
MOS	Mean Opinion Score
MSE	Mean Square Error
MUSHRA	Multiple Stimuli with Hidden Reference and Anchor
NFFT	Number of points in Fast Fourier Transform
NMR	Non-Matching Reference
PCS	Perceptual Contrast Stretching
PESQ	Perceptual Evaluation of Speech Quality
PReLU	Parametric Rectified Linear Unified
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RTF	Real Time Factor
SDR	Signal-to-Distortion Ratio
SI-SDR	Scale Invariant Signal-to-Distortion Ratio
SNR	Signal-to-Noise Ratio
SQUIM	Speech Quality and Intelligibility Measures
SSM	State Space Model

SSNR	Segmental Signal-to-Noise Ratio
STFT	Short Time Fourier Transform
STOI	Short-Term Objective Intelligibility
TCN	Temporal Convolutional Network
TF	Time-Frequency
WSS	Weighted Spectral Slope

Chapter 1

Introduction

Speech is one of the main means of communication for humans. Such communication partly determines our social nature as a species. Through speech we can send and receive messages. Hence, being able to extract the content of a message from an audio signal becomes a crucial task. Humans are able to do so by focusing on specific parts of the audio signals we receive. This is known as selective auditory attention. Although it may seem a trivial task, it is a much more complex one, as it is the result of evolution and the long learning process we experiment in the early years of our life (Gomes et al., 2000).

Similar to selective auditory attention, speech enhancement models focus on extracting the relevant information from audio signals. Such information corresponds to the clean speech signal. By having noisy signals as input, models distinguish between the clean speech and the noises that degrade it. After that, noises can be eliminated from the input signals, remaining only the clean speech that contains the relevant information. Speech enhancement can be divided into several subgoals. The two main ones include the improvement in the quality and intelligibility of the noisy signals (Bäckström et al., 2022, Gelderblom, 2023). The quality of an audio refers to the opinion of a person about the signal. The intelligibility, to how much a person understands the content.

To effectively create speech enhancement models, it is necessary to understand audio data. Audio data is time dependent. Each audio is conformed by several time points. Each time point is a sample with specific values. Because of the sequential nature of audio data, the order of these samples, should be taken into account when performing speech processing tasks. Audios are generally represented as waveforms in time domain, showing how the amplitude of the signal varies with time. Transformations can be applied to obtain frequency domain signals. Frequency analysis is quite common in speech tasks, as it allows studying the spectral content of the signal (Mehrish et al., 2023).

To selectively extract the clean signal, speech models focus on different features from the input signal. These features can represent time or frequency characteristics. Therefore, speech enhancement models can be developed in time or frequency domain (Mehrish et al., 2023). Time domain is much simpler as no transformations of the signals are required. However, frequency domain offers better results. The main problem that arises from frequency domain is that it treats signals as time indepen-

dent variables. As speech is non-stationary, another domain called time-frequency domain is required (Bäckström et al., 2022, Ochieng, 2023). Recent trends in speech enhancement involve models in time-frequency domain. This domain offers advantages from using both time and frequency domains. The results obtained with time-frequency domain models are generally better, but they imply higher computational cost, as more computations are required.

Speech enhancement models are currently based on deep learning. It was not always this way. Previously, statistical methods were used (Zheng et al., 2023). However, deep learning allows to easily work with high amounts of data, offers more flexibility and better results. Several deep learning elements have been studied in speech enhancement. Recurrent neural networks are employed because of their ability to capture temporal dependencies (Zheng et al., 2023). Convolutions are used in most of the speech enhancement models due to their efficiency in parameter sharing techniques and faster training (Mehrish et al., 2023). Denoising autoencoders are also quite popular in speech enhancement. They use the noisy signal as input and try to reconstruct an enhanced signal that is as similar to the clean audio as possible (Vincent et al., 2010). Generative adversarial networks, in which the generator is responsible for the enhancement, have also been explored (Skariah and Thomas, 2023). Finally, attention mechanisms are one of the most employed elements currently. Their ability to capture the most significant information from the noisy signal, makes them an ideal element for speech enhancement tasks (O'Shaughnessy, 2024).

Speech enhancement methods face several difficulties. Some of them are related to the variety of noises in real world applications, which makes the detection of noises more difficult. The evaluation of enhanced audios is also challenging, as the assessment of the enhanced signal quality is not a closed-formula operation. To that end, subjective and objective metrics are employed (Benesty et al., 2006). Other challenge addressed by speech enhancement models involves the variability in the ratio between the clean signal and noises. It is also difficult to use speech enhancement models in real time. An increasingly important challenge comes in relation to the implementation of speech enhancement models in small size devices (Zheng et al., 2023). Deep learning introduces complexity in the models, increasing the number of parameters and operations, and therefore limiting their efficiency.

The primary goal of this project is to develop an efficient and high performance speech enhancement model with deep learning techniques. Speech enhancement models have improved vastly in performance in recent years. However, the main problem of most of the models remains: most of them cannot be used in daily life tasks as they are not real time nor lightweight models. The focus of this work is precisely on this limitation. To achieve this goal, three main actions are taken. First, a framework for training and evaluating the models is developed, with strong focus on ensuring model generalization ability. Second, several efficient deep learning algorithms are designed. Finally, the models are trained, validated and evaluated. The models performance and efficiency results are obtained and analysed.

This document is structured as follows. Chapter 2 reviews the state of the art in speech enhancement. This chapter includes basic knowledge about speech enhancement, the most popular databases, data augmentation techniques, deep learning methods, objective and subjective metrics and the latest trends in speech enhancement. Chapter 3 presents the problem statement and the main goal of this project.

Chapter 4 explains the methods used for the development of the models of this project. This section does not only outline the design decisions, but also all the reasons behind them. Chapter 5 shows and discusses the results obtained for the models developed. Finally, Chapter 6 indicates the main conclusions drawn from this work and the future lines of work. For ease of reading, each chapter begins with a paragraph indicating the structure of the chapter.

Chapter 2

Speech Enhancement: State of the Art

In this chapter the state of the art in speech enhancement is explained. A first section with basic knowledge about speech enhancement and audio data is introduced. Then, common databases used as benchmarks in speech enhancement are presented. Methods for increasing the volume of available data are also explored. Following, current trends of deep learning techniques employed for speech enhancement are explained. The main networks include recurrent neural networks (RNN), convolutional neural networks (CNN), autoencoders, generative adversarial networks (GAN) and attention mechanisms. Losses used to train the networks are also included in the deep learning section. The metrics used to evaluate the models in speech enhancement are as well covered in this chapter. It is important to understand the metrics, as the results shown for this project are based on them.

Most of the speech enhancement models combine several elements of those explained in this chapter. Therefore, the first five sections deal with elements common to multiple models and the last two sections refer to specific models. The second to last section covers well-known models in speech enhancement. The last section explains in detail the best state of the art models. This last section is crucial to understand the models developed in this project.

2.1 Basic knowledge about speech enhancement

Humans are able to perceive noisy signals and discriminate between the desired signal and the unimportant information. This can be seen as an enhancement task, in which we are able to clean the signal, or remove the noise, keeping only the information corresponding to the part we are interested in. Speech enhancement refers to methods which extract the noise, to make speech sounds more pleasant, reduce listening effort and improve intelligibility (Bäckström et al., 2022). According to Gelderblom, 2023, the goal of speech enhancement can be divided into two main sub-purposes: improving the quality of the speech signal and improving the intelligibility of the signal. The first idea refers to the opinion of a person about that signal. The second, to how much a person understands the content.

Bäckström et al., 2022 indicate that speech enhancement has seven focuses: (i)

noise attenuation, (ii) echo cancellation, (iii) bandwidth extension, (iv) deverberation, (v) source separation, (vi) beamforming and (vii) active noise cancellation. Noise attenuation is the most common one and it is based on extracting the desired, or clean, signal that is originally distorted by background noises. Echo cancellation consist of first identifying the signal that is repeated with delay and then remove its repetitions. Bandwidth extension refers to methods which expand the frequency range of a signal where the interesting content is placed. The deverberation is the process of removing the reverberation, or reverb of a signal. The reverberation refers to the effect of room acoustics, due to signal reflections. Source separation methods isolate sounds of a single source. Similarly, beamforming focus on extracting sounds coming from a single direction. Finally, active noise cancellation, attenuates background noise by adding a second sound specifically designed to counteract the noise. Overall, it can be said that speech enhancement goal is to obtain an enhanced signal that resembles the clean signal, using as input a noisy signal. Thus, the quality and intelligibility of the input signal is improved.

Audio data is time dependent. A sound is time-varying motion of air (or some other medium) with an accompanying change in pressure. This change in pressure, which is time dependent, is an analogical signal. The analogical signal of the sound is recorded by a device and converted to a digital signal (Puckette, 2014). The digital signal represents the audio signal and is the one processed in speech enhancement tasks.

To perform speech enhancement, and other speech processing tasks, speech features are used. Speech features are numerical representation of speech signals used for speech analysis (Mehrish et al., 2023). These features can be in time domain or frequency domain. Therefore, speech enhancement tasks can be divided into time domain and frequency domain, depending on the features the model focuses on. Models in time domain focus on the amplitude or energy of the speech signals as a function of time. Frequency features involve representing the signal as a function of the frequency instead of as a function of time. This is done by the Discrete Fourier Transform (DFT). The equation of the DFT is:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}}. \tag{2.1}$$

where X_k are complex numbers that represent the magnitude and phase of an input signal x_n. X_k is the output in the frequency domain of the frequency index k. For a time interval $[0, N]$, at each time point n, the input signal is multiplied by a complex exponential term that represents the wave at frequency k/N. Overall, the result is a complex number that represents the amplitude and phase of a signal as a function of the frequency.

The main problem is that speech signals are not stationary (Bäckström et al., 2022, Ochieng, 2023). Stationary signals have properties that do not change over time. Speech signals show variations in both frequency and time. Hence, a third domain known as the time-frequency (TF) domain is defined. This is known as short-time analysis. The common approach is to compute a spectrogram which involves both the frequency and time variables. This is done by computing the Short Time Fourier Transform (STFT). This can be understood as the sum of DFT for different windows

of small intervals of time (Bäckström et al., 2022). The equation for the STFT is:

$$X_{k,m} = \sum_{n=0}^{N-1} w_n x_{n+mH} e^{-i2\pi \frac{kn}{N}}. \tag{2.2}$$

Similarly to the DFT (equation 2.1), the STFT (equation 2.2) computes the magnitude and phase information of a signal corresponding to a frequency k. The main difference is that the STFT does so for each window of time w using a hop factor H which represents the shift parameter in the time window used. Overall, the output $X_{k,m}$ is a time dependent variable. Models that use the STFT to transform the time signal into frequency signal, also use the Inverse Short Time Fourier Transform (ISTFT). The ISTFT does the inverse process, converting the output frequency signal into a time signal. Hence, the enhanced signal is, as the input signal, in time domain. It is common in the state of the art to refer to the time-frequency domain as frequency domain, although they are not strictly the same (Zheng et al., 2023). They both obtain frequency features. However, frequency domain only takes into account frequency features, while TF domain focus on both time and frequency features.

In figure 2.1, the signals in the three domains can be seen. The time domain waveform on the left shows the amplitude against the time. The frequency domain plot in the middle shows the spectral components of the audio. The graph represents the magnitude of the complex spectrogram. Note that human speech has low frequency values. Hence the amplitude of the speech signal shown in the figure is higher for low frequencies. Finally, the magnitude of the spectrogram on the right, corresponding to the TF domain, illustrates the frequency of the signal against the time. The amplitude is implicitly shown by the color of the plot. Yellow color correspond to higher amplitude regions, while blue color indicates lower amplitudes (Malinverno, 2023). Again, higher values correspond to low-frequency values.

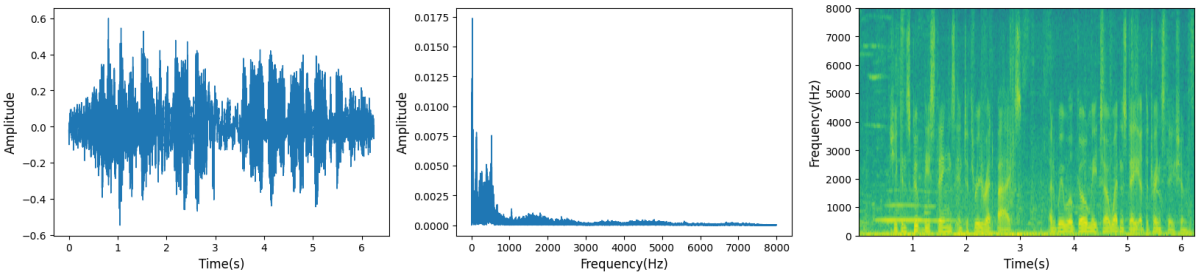

Figure 2.1: Speech signal represented in time domain (left), frequency domain (middle) and time-frequency domain (right).

The results obtained with frequency domain methods tend to be better than those obtained with time domain methods (Zheng et al., 2023). Also, time domain features pose difficulties such as the large input space or impossibility to capture some contents of the speech signal (Ochieng, 2023). Frequency domain features tend to be more sparse (Zheng et al., 2023), making it easier to detect and eliminate noise. Currently, the best state of the art models focus on TF domain, as it offers the advantages of time and frequency domain.

2.2 Speech enhancement databases

Speech enhancement databases constitute crucial elements for the speech enhancement field. The databases are conformed by noisy audios to be cleaned. The goodness of each model is assessed by its ability to remove the noises from a noisy signal obtaining an enhanced signal as output. To build speech enhancement databases it is common to mix automatic speech recognition (ASR) datasets and noises datasets. Virtually, every combination of these two kinds of datasets can conform a speech enhancement database. The obvious advantage of this strategy is that clean signals are known and therefore the comparison of the output of the models and clean signals is easily done. Thus, the goodness of the model is defined as the similarity between the output of the model and the clean signals. Different benchmarks have been defined for speech enhancement tasks. In this section, the most common ones are explained.

The database VoiceBank + DEMAND (Valentini-Botinhao, 2017), also known as Valentini after its developer, is created by combining the Voice Banking Corpus (Veaux et al., 2016) and the DEMAND (Thiemann et al., 2013) databases. The first corresponds to a clean speech database, while the second one is a database of noises. The Voice Banking Corpus is conformed by approximately five hundred speakers. All the speakers are native English speakers selected from different regions to capture the variety of accents. The dataset includes speakers between twenty and ninety years old and with males and females in similar proportions. The main drawback of the dataset is that the number of speakers from the upper social class is much higher than the number of speakers from the working class, which is almost non-existent. This likely results in the under-representation of regional accents of working class. The DEMAND dataset includes noises classified in six categories: (i) domestic, (ii) office, (iii) public, (iv) transportation, (v) nature and (vi) street. Each category is further divided into three subcategories. These noises try to represent real-world and common noises.

The VoiceBank + DEMAND database combines sixty speakers from the Voice Banking Corpus and thirteen types of noises from the DEMAND dataset. Two extra noises are synthetically created to simulate noises caused by human speech. They correspond to babble and speech-shaped noise. The babble noise is a mix of six speakers of the VoiceBank database, while the speech-shaped noise correspond to white noise with frequency of a male speaker. For each noise condition, different Signal-to-Noise Ratio (SNR) are employed. Different speakers, noise types and SNR are employed in the train and test set.

Deep Noise Suppression (DNS) Challenge (Reddy et al., 2020) again combines ASR datasets with noises datasets. The clean, ASR, signal dataset is Librivox (McGuire, 2005), which has recordings of volunteers reading over ten thousand books in various languages. Although some recordings are of good quality, the majority present background noises, speech distortions and other types of undesired signals. Therefore, when the DNS database was created, a meticulous filtering was performed to select only clean signals of high quality. The noises come from the Audioset (Sound and video understanding teams at Google, 2017), Freesound (Music Technology Group of Universitat Pompeu Fabra, 2005) and DEMAND (Thiemann et al., 2013) databases. The first one is composed of two million videos of ten seconds from YouTube. The samples from such dataset are selected so that it present class balance. Audios with speech were deleted so that they do not affect the speech enhancement task.

Some noises from Freesound and DEMAND databases were lastly added to the pool of noises from the Audioset database.

Other common benchmark is ReVerb (REverberant Voice Enhancement and Recognition Benchmark) Challenge (Kinoshita et al., 2013). The dataset is conformed by recordings of one stationary and distant-talking speaker in a reverberant room. The dataset has real recordings and simulated recordings obtained by merging clean signals with recorded background noises. The reverberation times are modified to have a broader representation of the reverberation in the dataset. Other examples of popular datasets include WHAM! (Wichern et al., 2019) with data of two speakers in a noisy environment, WHAMR! (Maciejewski et al., 2020) which adds synthetic reverberated sources to WHAM! and EasyCom (Donley et al., 2021) for the cocktail party effect.

2.3 Data augmentation for speech enhancement

The speech databases used as benchmarks in speech enhancement are generally not large. Hence, strategies to increase the volume of that data are required. Data augmentation refers to methods used to reach such goal. These techniques are based on applying transformations to the datasets, obtaining new data. By increasing the volume of data, so does the variability of the datasets used to train models. Thus, models are not limited to a specific type of signal and can generalize better. Data augmentation is one of the most popular techniques used to avoid overfitting. In databases with a small to medium volume of data, data augmentation becomes a powerful strategy to obtain optimal results (Braun and Tashev, 2020). Multiple techniques of data augmentation have been developed for audio data (Abayomi-Alli et al., 2022, Ferreira-Paiva et al., 2022, Alex et al., 2023). In this section the main data augmentation techniques applied in speech enhancement tasks are explained.

Many speech enhancement databases are created by mixing audios with clean signals and audios with noises. Consequently, it is not costly to apply data augmentation techniques, as it is easy to create new combinations of noisy audios in time domain. Noisy audios can be easily obtained by applying additive noise to the clean audios. Transformations can be applied to the clean audios, the noises and/or the noisy audios. As audio signals constitute a complex form of data with many features in different domains, the range of possible transformations to be applied is very broad.

An easy strategy to increase the volume of data is based on corrupting audio with white noise. This is because of two reasons: it is easy to generate and it helps generalizing as it can cover all frequencies. The white noises are generated by a Gaussian or random distribution. The noises are added to the clean signal at different SNR. Many audios can be obtained from the same white noise and clean signal by adjusting such SNR. Creating samples with low SNR, helps the model learn how to remove noises even in highly noisy environment. The white noises can be stationary or non-stationary (Loo, 2020).

One of the most common strategies is to add random background noises to the clean signals. It is a very popular approach as noisy audios from speech enhancement databases are created by mixing the clean audios and the noises. A random noise from the noises collection can be selected and added to a clean audio, creating a new variation of noisy audio. The SNR between the clean signals and background noises

can vary. It has the same principle as the white noise addition, but instead of using synthetic noises, it uses the noises that are already present in the database. It is is a fairly simple but effective method.

Data augmentation techniques can also be based on reverberation (Tang et al., 2018, Defossez et al., 2020). Reverberation occurs because sounds reflect off the surfaces of an enclosed space before reaching the recording microphone. When the reflected sound is distinguishable from the original sound, it is known as an echo. Reverberation is a consequence of room acoustics. It can be easily simulated. The audio at a point of time is selected, extracted and added to the signal with a certain delay and lower amplitude. This can be repeated several times for the same sound, using different delays and amplitude levels. In this way the model is trained to be able to eliminate the reverberation and the volume of data can be easily increased.

Modifications cannot only be made in time domain, but also in frequency domain. BandMasking techniques (Defossez et al., 2020) refer to strategies that mask specific frequency bands by setting the amplitude to zero. Different filters can be applied. By masking specific and different bands of frequency, the number of audios increases and the model can learn to focus on specific robust signal features. This is specially useful to mask frequencies that are not human-perceivable, so that the model does not focus on them. Similarly, time masking, in which a time band amplitude is set to zero, can be performed. SpecAugment (D. S. Park et al., 2019), a famous data augmentation technique in speech enhancement, combines both time and frequency masking. Specaugment also involves time warping, in which the speed of some parts of the audio is increased, while the speed of other parts is reduced.

Time stretch and pitch shift strategies have also been employed for data augmentation in speech tasks (Wei et al., 2020). Time stretch refers to changing the speech speed and therefore the duration of the audio, while maintaining the pitch. Pitch shift refers to the opposite operation, in which the pitch is modified, while the speed of the audio is unchanged. The pitch can be modified to different semitones.

2.4 Deep learning for speech enhancement

Deep learning has not always been the standard procedure for speech enhancement (Zheng et al., 2023). Actually, it was not until Y. Wang and Wang, 2012 proposed using deep learning to deal with the cocktail party problem, that neural networks started being used for speech enhancement. Before, statistical methods were used. Statistical methods assume that the clean signal and the noises are independent and that one of them follows a specific distribution. Therefore, statistical or traditional methods are based on statistical treatment of one of the signals, clean signal or noise, that conform the input. These techniques were quickly replaced by deep learning. Deep learning techniques pose an incredibly well suited strategy to perform speech enhancement. The results of speech enhancement tasks have improved by using neural networks (Zheng et al., 2023).

Neural networks are representation-learning method as they obtain the representation needed to perform an specific task from raw data (LeCun et al., 2015). Their main unit is called neuron. Neurons are organized in layers and each layer is connected to the next one. Each connection has a weight assigned and each layer a bias. By adjusting the weights and biases, the model learns to give more or less importance

to specific features of the input data in order to obtain the desired result. The weights and biases are adjusted by the backpropagation gradient (Rosenblatt, 1962). A loss is fixed to compare the output of the model and the target output. By minimizing the loss, the gradient is computed and backpropagated so that weights and biases are adjusted. Training a neural network can be understood as finding the best trainable parameters so that the loss is minimized.

In this section, the main deep learning motifs used in speech enhancement are explained. The types of neural network elements include (i) Recurrent Neural Networks (RNNs), (ii) Convolutional Neural Networks (CNNs), (iii) autoencoders, (iv) Generative Adversarial Networks (GANs) and (v) attention mechanisms. The enumerated elements are explained in the first five subsections. The last subsection of this section refers to the losses commonly used to train speech enhancement models. This section does not include the explanation of specific speech enhancement models, but of the elements they are conformed of. This section is used to ensure a solid foundation in the different elements of deep learning that are employed by speech enhancement models. Specific models are detailed in later sections with reference to the elements described in this section.

2.4.1 Recurrent Neural Networks

Recurrent neural networks (RNN) (Rumelhart et al., 1986) are neural networks that use feed back loops in contrast to traditional feed forward neural networks (FFNN), which only allow forward connections between neurons (Zell, 1994). In RNN the output of a neuron can be used as input of that neuron or neurons from previous layers as shown in figure 2.2. Since input can be given sequentially and the output of one neuron can affect neurons in the same or previous layers, it follows that previously processed elements affect subsequently processed elements. In other words, the output of a neuron depends on previous elements of the sequence that were processed before. This makes RNN excellent architectures to work with sequential data as speech signals. RNN are said to have inherent memory, as previous elements in the sequence influence current and future learning.

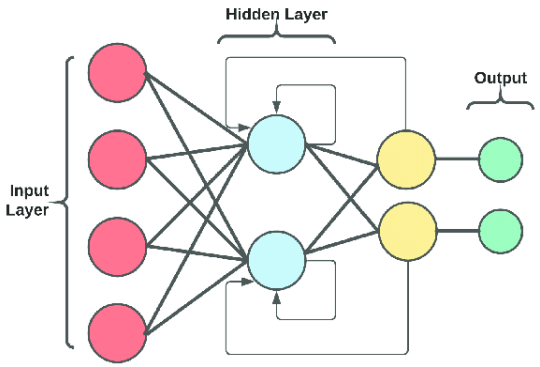

Figure 2.2: Recurrent neural network with two hidden layers and recurrence between different layers and within the same layer (Arias et al., 2022)

Traditional RNN have two main problems: the context at each stage depends only on previous elements, not future elements, and they suffer from short time memory. The short time memory is a consequence of the vanishing gradient problem. Earlier words

are forgotten as their gradient vanishes earlier (Wisam, 2022). Some modifications of traditional RNN have been developed in order to overcome these problems.

Bidirectional Recurrent Neural Networks (BRNN) (Schuster and Paliwal, 1997) focus on the first problem. They can be understood as a combination of two RNN: one which uses previous elements to determine current elements, and other that goes in the other direction, using future elements to affect the current ones. This can be useful, for example, to decipher a word from a sentence, where future words may give more information than earlier words.

Long Short Time Memory (LSTM) (Hochreiter and Schmidhuber, 1997) deals with the second problem. LSTM learns only the important information and forgets the irrelevant one. This is desirable when the current learning depends on elements that are not in the recent past. To do so, LSTMs use three gates: input gate, output gate and forget gate. By using the forget gate, the network eliminates the information that is not relevant and the crucial information is passed to the memory. The memory at each unit is combined with the input to obtain the output. The process of selecting only the important information to modify the memory is repeated at each unit. Bidirectional Long Short Term Memory networks (BiLSTM) (Graves and Schmidhuber, 2005) are a combination of BRNN and LSTM.

Gated Recurrent Units (GRU) (Cho et al., 2014) also cope with the short time memory of RNN. Instead of input, forget and output gates used by LSTM networks, GRU use update and reset gates. The update gate determines the information that goes to the next state, while the reset gate determines the information that is discarded.

The main advantage of recurrent neural networks is that they do not assume that outputs and inputs are independent. Hence, RNN are very well suited for sequential data, which is the scenario in speech enhancement. However, because of its architecture, RNN can be very slow to process, both during training and inference (Shelf, 2024).

2.4.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) (LeCun et al., 1989) are other type of neural networks commonly used in speech enhancement. CNNs have three types of layers: convolution layer, pooling layer and fully-connected layer. The first layer is typically a convolution layer and the last one a fully-connected layer. Between them, several convolution and pooling layers are placed.

Convolution layers use the input data, the feature map and a filter or kernel. The kernel moves across the input data checking if a feature is present. A dot product is computed between the input and the kernel. The dot product is passed to the output and the kernel shifts by a stride. This is repeated until the kernel has been applied to the whole input data. This process is known as convolution. Thus, the name of the layer. The number of filters, the stride and the padding need to be set as hyperparameters in this type of layer. The number of filters determine the output depth. The stride is the number of data samples that the kernel moves over the input matrix, before performing the next convolution operation. Padding is used when the kernels do not perfectly fit the input data. Padding refers to adding artificial data samples to the input data so that it fits the kernels and dot products can be performed without any loss of edges information. The most typical padding is zero-padding in

which elements that are outside the input are set to zero, although different padding strategies can be used.

After convolution layers, generally a pooling layer or downsampling layer is used. This type of layer conducts dimensionality reduction. It uses another filter to aggregate different data samples and reduce dimensionality. Generally, average or max pooling is used. In the first one, the average value of the data samples of the input data, selected by the pooling filter, replaces the whole section. Similarly, the second one replaces the group of data samples by the maximum value of the input data group. In this layer, a lot of information is lost but it makes convolution neural networks more efficient. CNN tend to have several convolution layers followed by pooling layers before reaching the last layers. Serialized convolutions and pooling layers extract the main features which are fed to the fully connected layers. Based on this information, the fully connected layers classify the input.

Figure 2.3 illustrates an example of CNN in which the network classifies images depending on the object they present. It can be seen how the hidden layers are composed by several convolution and pooling layers. The last layer, which is fully connected, is used to classify the input image. Two activation functions are indicated in the image: Rectified Linear Unit (ReLU) after convolution layers and softmax after fully connected layers.

Activation functions are employed to transform the output of neurons depending on the input, weights and biases (Nwankpa et al., 2018). Activation function can be either linear or non-linear. In figure 2.3 a linear and a non-linear activation function are employed. The ReLU, which is a linear activation function, gives a value of zero to the output of the neuron when it is equal or lower than zero. For positive values of the output, it maintains the output value. The softmax function is not linear and is used in classification tasks, as its output is within the range $[0, 1]$ (Nwankpa et al., 2018). Non-linear activation functions generally help with the vanishing gradient problem and are used in the output layer of a classification task, such as the one of figure 2.3.

The main problem of convolutional networks is their inability to capture long range dependencies. When this type of dependencies are needed, many layers are required. A solution is to use dilated convolutions (F. Yu and Koltun, 2016). Dilated convolutions imply expanding the kernel by skipping some data samples in the kernel. In other words, dilated convolutions apply a separation between data samples of the kernel so as to cover a broader region of the input data. Therefore, longer range dependencies can be captured with practically the same computational cost.

Another important architecture based on convolutional networks is the U-net (Ronneberger et al., 2015). The U-net has a contracting path and an expansive path. The contracting path can be understood as traditional hidden layers of a convolutional network. The expanding path upsamples the data instead of downsampling it. Skip connections are used between the corresponding layers of the contracting and expanding path. Skip connections allow the combination of outputs of different layers that are not sequential, obtaining an overall improvement of the results. The first layer of the contracting path connects to the last of the expanding path, the second of the contracting to the second last of the expanding, and so on. In the end, the network presents a U-shape from which it inherits its name.

Bai et al., 2018 standardized the term Temporal Convolutional Network (TCN) to refer

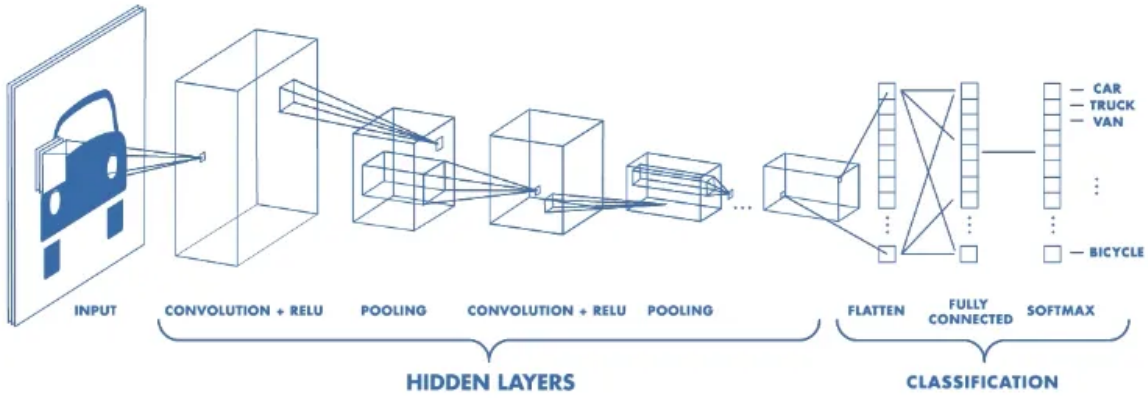

Figure 2.3: Convolutional neural network. The input data is taken by a convolution layer followed by a pooling layer. Hidden layers are defined by a varying number of convolution and pooling layers. After those, the fully connected layer is used to classify the image (Pingel and Patel, 2017).

to a family of CNN architectures that are causal. A causal model is one that uses current information and past information, not future information. This is useful in real time applications, where future data is not known. Because of this causality property, TCNs are employed for real time speech enhancement. TCNs intend to solve the CNNs inability to capture temporal dependencies.

Other type of CNN are the Convolutional Recurrent Neural Network (CRNN) (defined by Keren and Schuller, 2017 based on the work of Pinheiro and Collobert, 2013). CRNN mix CNN and RNN to combine the advantages of the two network types. The convolution and pooling layers of CNN are used for feature extraction and dimensionality reduction, while the recurrent layers capture temporal dependencies. CRNN pose a solution to the main problem of CNN, which is the inability to effectively capture temporal dependencies. However, CRNN introduce high computational cost associated with RNN.

Convolutions are quite popular in speech enhancement for two reasons. First, they allow efficient feature extraction. Second, they are fast to train and have low number of trainable parameter, due to their parallel processing and parameter sharing property.

2.4.3 Autoencoders

Autoencoders (Kramer, 1991) are other important structure in speech enhancement. Autoencoders are conformed of an encoder, a code or latent space and a decoder. As figure 2.4 shows, the encoder and decoder are symmetrical. In the encoder, the number of neurons per layer decreases from one layer to the next one, while in the decoder it increases. The encoder is responsible of reducing the dimension of the input. The decoder is responsible of reconstructing the input from the latent space.

Autoencoders have several applications including anomalies detection, dimensional-

ity reduction or denoising. In speech enhancement autoencoders become popular architectures as they are used for the denoising task. According to O'Shaughnessy, 2024 an encoder-decoder structure is one of the most appropriate architectures for speech enhancement. The encoder finds the embeddings, or hidden vector representations, of the input signal. It learns the most important features of an input. The decoder is trained to reconstruct the input signal. Therefore, if the most important features learned by means of the encoder do not include noises, the decoder reconstructs a clean version of the input signal.

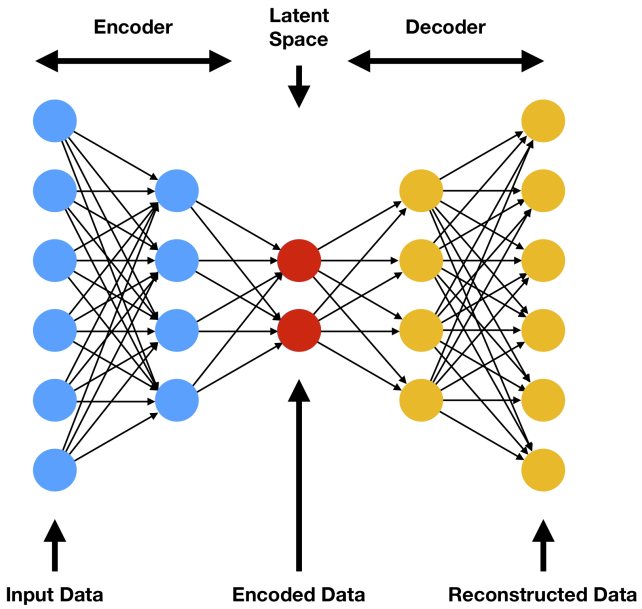

Figure 2.4: Autoencoder. The input data enters the encoder. The encoder learns the most important features and reduce the dimensionality of the data in the latent space or code. The encoded data is given as input to the decoder which reconstructs the data. (Flores, 2019).

Variational autoencoders (Kingma and Welling, 2022) are a type of autoencoders in which the latent space is defined by a distribution. Hence, the latent space is designed to be smooth and continuous, so any point of the it can generate significant outputs. This is useful in applications like speech enhancement, as it allows for more reliable and robust reconstruction of the enhanced signal. Conversely, if random points are picked from the latent space in traditional autoencoders, they may produce distorted outputs, which can lead to issues known as hallucinations.

Denoising autoencoders (Vincent et al., 2010) are the type of autoencoders that are most common in speech enhancement. Standard autoencoders compare the output of the decoder to the input of the encoder and try to minimize the difference between them. Conversely, in denoising autoencoders, the output of the decoder is not compared to the input of the encoder, as it corresponds to the noisy signal. Instead, it is compared to the clean version of the input signal. Thus, the difference between the enhanced signal and the clean signal is minimized.

Denoising autoencoders are, by far, one of the most common architectures in speech enhancement. They are more than appropriate structures to subtract noises from the noisy signals. However, it should be taken into account that autoencoders can

easily overfit the data. Since they learn the distribution of noises and clean signals of the training data, they have difficulties generalising given noises or speech signals that follow different distributions.

2.4.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are neural networks that involve two separate networks: the generator and the discriminator. Figure 2.5 gives the overall structure of GANs. The generator goal is to produce data indistinguishable from the training dataset. It takes as input a simple distribution, such as uniform or Gaussian, and by means of the inverse transform, searches for a transformation so that its output resembles the training data. Note that in figure 2.5, the input noise does not refer to noises from the noisy audios used in speech enhancement, but to the random distribution from which the fake data is generated. The discriminator purpose is to be able to differentiate between the real training data and the fake data created by the generator. The weights of the generator are updated in order to try to deceive the discriminator into labeling its generated data as real data. The weights of the discriminator are updated in order to easily differentiate between the data coming from the train set and the data generated by the generator.

During the early epochs of the training process, the generator does not fool the discriminator. Later, the generator learns to create data more similar to the data in the train set and begins to fool the discriminator. At the end of training, the generator produces data practically indistinguishable from the train set data and the discriminator is unable to distinguish between real and generated data.

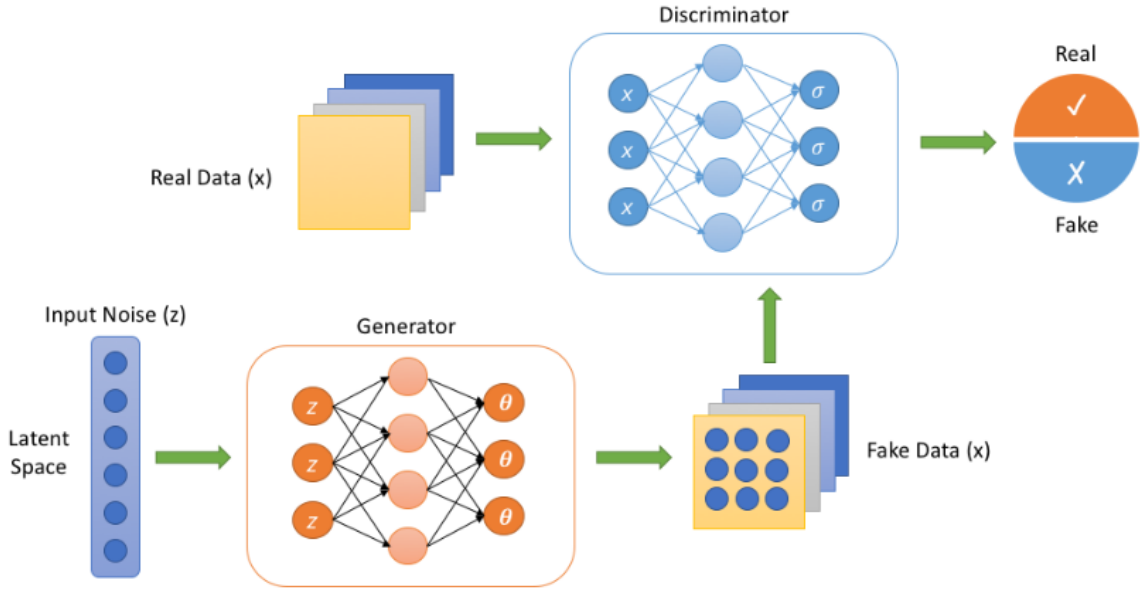

Figure 2.5: Generative adversarial network. The input noise, which follows an specific distribution, is used by the generator to produce fake data. The real and fake data are given as input to the discriminator. The discriminator then labels the data as real or fake (Verma, 2019).

In speech enhancement, the generator uses the noisy signals as input and outputs an enhanced version of the input. The generator goal is to clean the noisy signal so

well, that the discriminator is not able to distinguish between enhanced and clean data. Therefore, the generator is the one that performs the speech enhancement task, while the discriminator is used to assessed how well the generator is performing the enhancement. In figure 2.5 the input noise (z) corresponds to noisy audios, the fake data to enhanced audios and the real data to clean audios. The discriminator labels the data as clean or enhanced.

GANs have been used by several models in speech enhancement (Skariah and Thomas, 2023). They are gaining popularity in recent years, as they are achieving impressive results in speech enhancement (Cao et al., 2022, Zadorozhnyy et al., 2022, Y.-X. Lu et al., 2023, Z. Liu et al., 2024). GANs offer high-quality enhanced signals. However, their training process is quite unstable and they have a higher number of hyperparameters to be optimized than other networks.

2.4.5 Attention based architectures

Attention mechanisms were introduced by Bahdanau et al., 2014. They are based on the idea of focusing on specific sections of the input. They assign weights to different parts of the input, determining their relative importance. The weights can change dynamically. It was not until the work of Vaswani et al., 2017 that attention mechanisms became a standard practice in many deep learning tasks.

Vaswani et al., 2017 define a new architecture based on attention mechanisms, called transformers. Transformers have gained importance in language processing tasks due to their ability to focus on specific parts of the input. Because of the success of encoder-decoder structures in language processing tasks, the first transformer was also composed of these elements. However, transformers can also be used in encoder-only or decoder-only mode (Tay et al., 2022).

Figure 2.6 indicates the main structure of the encoder-decoder transformer. From left to right it includes information of the attention mechanism, the multi-head self-attention, the encoder and the decoder of the transformer.

The scaled-dot product that defines the attention mechanism, shown on the left of figure 2.6, is based on three elements: queries (Q), keys (K) and values (V). A query refers to the element for which information is sought. A key is the information against which the query is compared. A value is the information associated to each element. The values are weighted by the score obtained from the comparison of query and keys, which means that queries more similar to the keys, have higher values and therefore more influence in the output. The attention formula over the queries, keys and values can be defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

where d_k refers to the dimension of the keys vector. Its square root is used as a scaling factor to stabilize the training. The softmax function normalizes the weights. K^T refers to the transpose of the keys used to compute the dot product with the queries.

Figure 2.6 shows that transformers use variants of this general attention mechanism. Four variants of the attention mechanism are combined and used in transformers. These variants are (i) self-attention, (ii) cross-attention, (iii) multi-head attention and

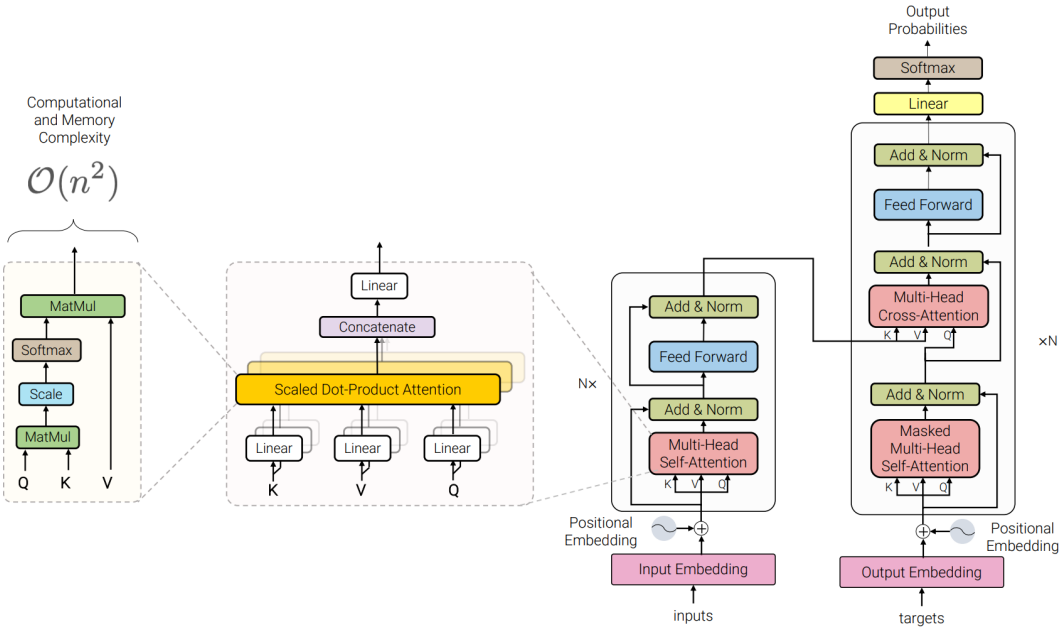

Figure 2.6: Transformer architecture. From left to right: scaled dot product attention, multi-head self attention, encoder of the transformer and decoder of the transformer. Q, K and V refer to Queries, Keys and Values respectively. (Tay et al., 2022)

(iv) masked attention. Self-attention allows the model to pay attention to parts of the same input sequence that is being processed. This means that the keys and queries used are from the same sequence. Cross-attention uses keys and queries from different sequences, fostering attention between different sequences (Kosar, 2022). Multi-head attention, refers to the fact of performing multiple times attention in parallel and then concatenate the outputs, obtaining a much richer result. This is shown in the second element from the left of figure 2.6. Finally, in masked attention, future inputs are masked at $-\infty$ in order to maintain the causal property. Hence, models cannot use information about the future (Kierszbaum, 2020, Vaswani et al., 2017). The original transformers combine these elements giving raise to the multi-head self-attention of the encoder and the masked multi-head self-attention and multi-head cross-attention mechanisms of the decoder.

Figure 2.6 also shows the encoder and decoder structure of the transformer, on the two elements on the right-hand side of the figure. The encoder is composed of N layers, where $N = 6$ in the original paper. Each layer has two sublayers: a multi-head self-attention layer and a feed forward layer. The output of each sublayer is added to the input of the sublayer and normalization is applied. The decoder is again composed of N layers, where $N = 6$ in the original paper. Each layer is composed of three sublayers: a masked multi-head self-attention sublayer, a multi-head cross-attention sublayer and a feed forward sublayer. The multihead-cross attention sublayer uses keys and values from the encoder output and queries from the previous sublayer of the decoder output. Analogously to the encoder, the output of each sublayer is added to the layer's input and normalized. A linear transformation and softmax activation function is applied to the decoder output to obtain the output probabilities.

Motivated by the success of transformers and CNN in speech tasks, Gulati et al.,

2020 defined a variation of the transformer called conformer. The conformer is based on the idea that merging convolutional architectures and self-attention mechanisms gives better results than using them separately (Bello et al., 2019). Conformers are mainly used in an encoder-only mode, as Gulati et al., 2020 do not focus on the decoder part of the architecture. Figure 2.7 shows the overall structure of this encoder-only conformer architecture.

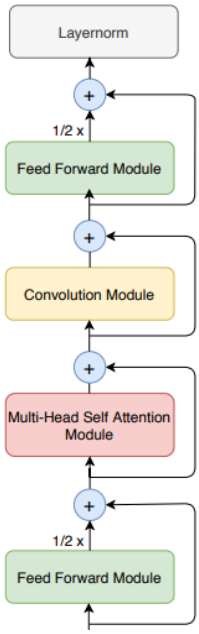

Figure 2.7: Conformer encoder structure. It has a macaron net structure with two feed forward modules with half-step residuals at the beginning and end of the encoder. After the first feed forward module a multi-head self-attention module is placed and followed by a convolution module. The two of them use residual connections. After the last feed forward module, layer normalization is performed (Gulati et al., 2020).

The conformer is similar to the transformer but it has some differences. Instead of an encoder with two modules, the conformer encoder employs four modules. Note that the modules of the conformers are the equivalent of the sublayers of the transformers. First a feed forward module is placed and followed by a multi-head self-attention module. After that, a convolution module is used. Finally, another feed forward module is employed (Y. C. Liu et al., 2021). The output of each module is added to its input and normalized. After the last feed forward module a layer normalization takes place. The 1/2x in the residual connections of figure 2.7 refer to half-step residual connections instead of standard connections. The output of each feed forward module is multiplied by 1/2 before being added to its input and passed to the next module. This structure with two feed forward modules with half-step residuals is known as Macaron Net. It shows better results and lower error than using only one feed forward module with standard residual connections as a first module (Y. Lu et al., 2019).

Branchformers are other architectures that use attention mechanisms (Peng et al., 2022). They are similar to conformers. The main difference is that the attention module and convolution module are not used in series but in parallel. They have different branches that focus on local and global dependencies. The local dependen-

cies are captured through the convolution module, while the global ones are obtained through the attention module. The branches are merged before the last feed forward module. In ASR, the state of the art is based on the E-branchformer (K. Kim et al., 2022). E-branchformer is a variation of branchformers in which the merging mechanisms of the two branches is improved.

Transformers, conformers and branchformers are popular architectures in speech processing tasks. Their main advantage is that they can focus on relevant information and capture long-range dependencies. They can also handle huge amounts of data. Their main drawback is that they require a high number of parameters and are quite complex and computationally expensive. Many studies have proposed modifications of these architectures in order to develop more efficient structures. Tay et al., 2022 review a number of these strategies, all of which take into consideration the complexity-performance trade-off. For example, Luo et al., 2022 propose using parameter sharing, Sun et al., 2020, reducing layer sizes and S. Chen et al., 2022 and Sanh et al., 2019, using online knowledge distillation. Online knowledge distillation is a process where a smaller, student model, learns from a larger, teacher model, continuously. This strategy allows to have a small model but with high performance (Ochieng, 2023).

In addition to these general strategies, some approaches have been developed specifically for attention structures. Some examples of this include the longformer (Beltagy et al., 2020), the reformer (Kitaev et al., 2020), the linformer (S. Wang et al., 2020) and the fastformer (C. Wu et al., 2021). The longformer uses sparse, or dilated, attention mechanisms, reducing the number of computations. The reformer uses locality sensitive hashing attention mechanism. This means that instead of computing values for all pairs of keys and queries, the reformer defines some buckets and only computes values for keys and queries in the same bucket. The linformer uses linear projections to decompose the original attention matrix into smaller matrices and therefore with lower computational cost associated. The fastformer uses additive attention mechanisms instead of dot product, reducing also the computation attached. These elements do not only show remarkable attention capabilities, but are also efficient. On top of that, many ASR models integrate these efficient elements. As ASR constitutes the main task in speech processing, it sets trends that other tasks, as speech enhancement, typically follow.

2.4.6 Losses

Deep learning models are based on finding the parameters of the neural network so that it presents a determined behaviour. The learnable parameters include the weights of each connection between two neurons of adjacent layers and the bias of each layer. Other parameters may also be learnable, such as batch normalization parameters. This learning process is done by the backpropagation gradient (Rosenblatt, 1962). The network with a specific set of parameters produces an output. With the output and the target values, the loss is computed, capturing the difference between them. The training process goal is to obtain output values that resemble as much as possible the target values. Thus, the goal can be understood as minimizing the loss. The gradient of the loss function, representing the direction and rate of change in the loss with respect to the parameters, is computed by deriving the loss function with respect to the parameters. This gradient is then propagated backwards to update the

parameters. They are adjusted to minimize the loss. Note that in the case of GANs, two losses are required, one for the generator and another for the discriminator.

Neural networks learn their parameters based on the loss selected. Hence, the loss formula is quite an important factor in deep learning. In speech enhancement many formulas have been used. In the following equations, y represents the target value and $\hat{y}$ the output of the model. y and $\hat{y}$ can represent different features from the target and output signal, both in time and frequency domain.

Some of the most common loss formulas include the Mean Square Error (MSE) or L2 loss and the Mean Absolute Error (MAE) or L1 loss. Their formulas correspond to:

$$L2 = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2 \tag{2.4}$$

$$L1 = \frac{1}{n}\sum_{i}^{n}|y_i - \hat{y}_i| \tag{2.5}$$

Other metric, commonly used in speech enhancement, is the Logarithmic Spectral Distance (LSD). It is similar to MSE, but the magnitudes are transformed to logarithmic scale with base ten. This arises from the fact that the human hearing has logarithmic perceptual nature. The LSD formula is:

$$LSD = \frac{1}{n}\sum_{i}^{n}(log_{10}(y_i) - log_{10}(\hat{y}_i))^2 \tag{2.6}$$

In speech enhancement, ratios are also employed as loss functions. The main advantage of using ratios is that they are scale invariant and thus more stable. This equation is:

$$LR = \frac{||y - \hat{y}||_i}{||y||_i} \tag{2.7}$$

where $||\cdot||_i$ refers to L1 or L2 depending on the loss function employed from equations 2.4 and 2.5.

Most speech enhancement models employ a loss with several terms (S.-W. Fu et al., 2017). Each term measures the difference between the target and the output, focusing on different features of the signal. Each term has the form of one of the equations explained above. The most common one is the MSE (equation 2.4). However, the remaining loss formulas explained and many others, as indicated by Braun and Tashev, 2021, can be used.

Based on the feature that y represents, several loss terms can be defined. According to Zheng et al., 2023, the losses used in speech enhancement can be divided in three groups: (i) frequency domain, (ii) time domain and (iii) perceptual losses. Williamson et al., 2016 propose combining losses from frequency domain and time domain. This has become an standard practice in speech enhancement models that work in the time-frequency domain. The most common losses in frequency domain are L_{mag}, L_{RI} and L_{phase}. In the first one, the y represents the magnitude. The second one, which compromises two terms, has y as the real and imaginary parts of the signal. In the third one, the y represents the phase of the signal. In time domain a unique term, L_{time}, in which the y refers to the waveform signal, is commonly employed.

In the case of perceptual losses, the y refers to speech enhancement perception metrics. The most common metrics include the Perceptual Evaluation of Speech Quality (PESQ) (Rix et al., 2001) and the Short-Term Objective Intelligibility (STOI) (Taal et al., 2011). The first measures the quality of the enhanced signal, while the second measures the intelligibility of the signal, although quality and intelligibility are closely related. These metrics are also used to evaluate model performance. Note that if the same metric is used for the loss calculation and for the evaluation of the model, the results obtained after the evaluation can be misaligned (de Oliveira et al., 2024). However, since multiple metrics are used for model evaluation, there is usually a balance in the final assessment. This is because compensation by the other metrics not used in the loss, takes place (S.-W. Fu et al., 2019).

2.5 Metrics for speech enhancement

In speech enhancement, and other deep learning tasks, specific metrics are required to asses the goodness of the model. Several metrics for speech enhancement have been proposed. In this section, the most common ones are elucidated. These metrics aim to capture human perception. The metrics can be either subjective or objective (Bäckström et al., 2022). Subjective metrics are those based on human evaluation. The main aspects that are evaluated by the listener are (i) the sound or speech quality, (ii) the interaction and communication quality and (iii) the service quality and user experience. The first one includes characteristics like the noisiness, the distortion and the intelligibility. The second one includes the delay or the echo of the signal. The last one refers to the system, device or interface.

Subjective metrics can be divided into two types depending on the listeners who evaluate the audios. The listeners could be experts or *naïve*. If the listeners are experts, the standard applied is the Multiple Stimuli with Hidden Reference and Anchor (MUSHRA), recommended in ITU, 2003, where each sample is rated from 1 to 100. ITU stands for International Telecommunication Union. Higher ratings correspond to better audio qualities. As the listeners are experts, a low number of them is required. However, quality control with expert listeners is not as common as with *naïve* listeners.

For non-expert listeners, the standard applied is the Mean Opinion Score (MOS) recommended in ITU, 1996, although many other recommendations can also be implemented (Streijl et al., 2016). Each sample is rated from 1 to 5, with better qualities corresponding to higher ratings. Some MOS metrics that are commonly applied include SIG, BAK and OVRL which measure the speech signal, the background noise and the overall quality respectively, as Hu and Loizou, 2006 indicate. In their work, the authors compare these subjective metrics with objective metrics. They show that some objective metrics lack the ability to efficiently capture human perception and seem uncorrelated with subjective measures. Therefore, even thought obtaining subjective measures is more expensive, due to the need of experimental set up, subjective metrics are desired.

In order to reduce the cost of subjective metrics, models have been developed to predict MOS. Manocha and Kumar, 2022 develop a model which do so by comparing the target enhanced audio and clean audios of the database. The clean audios selected are known as Non-Matching Reference (NMR). This means, that the clean audios em-

ployed to compute the MOS of an enhanced audio are not the clean version of the enhanced audio. The NMR audios, which are clean, are assigned the highest MOS value, due to the lack of background noise, the good quality and the intelligibility of the clean signals. The MOS score of the enhanced audio is computed using the NMR clean audio and a deep learning algorithm. The better the quality, the more resemblance between the enhanced and clean audio, and the higher the computed MOS value. The computed MOS values of their algorithm show low MSE with respect to real MOS values. Therefore, their model offers a plausible solution to easily obtain computed subjective metrics. Speech Quality and Intelligibility Measures (SQUIM) (Kumar et al., 2023) implements their model.

The second type of metrics, objective metrics, refer to measures that try to estimate the quality of a signal based on human perception (Bäckström et al., 2022). These type of metrics are the most common ones. Subjective metrics are expensive and not completely consistent, as the listeners who rate the signals are not always the same. Objective metrics are more widespread as they are cheaper, easier to obtain and more consistent. The six most common metrics that are used to compare the goodness of speech enhancement models are explained in what follows.

The Perceptual Evaluation of Speech Quality (PESQ) (Rix et al., 2001) measures the quality of the enhanced signal. It is the most common and most valued metric to indicate the goodness of a speech enhancement model. It is measured from -0.5 to 4.5, with 4.5 being the best value. The best state of the art models have a PESQ over 3.5. Other common measure is the Short-Term Objective Intelligibility (STOI) (Taal et al., 2011). STOI measures the intelligibility of the signal instead of the quality. STOI scores range from 0 to 1, with higher values indicating more intelligible signals. Current state of the art models have a STOI around 0.95. The computation of PESQ and STOI is quite complex and does not imply a simple closed-form formula. PESQ focus on distortions, time alignment and amplitude. STOI focus on frequency bands, short time analysis of the signal and the overall signal.

Another common metric is the Segmental Signal to Noise Ratio (SSNR) (Hansen and Pellom, 1998), which is a modification of the SNR measure. The main difference between the SNR and the SSNR is that the second one computes the SNR for short frames of time and then obtains the mean over the segments. The higher granularity in the sample processing, makes SSNR a preferred metric over SNR. This is because even very short audios can vary greatly over their duration, making it meaningless to obtain the difference in SNR over the entire audio. It makes more sense to get the difference for small frames and then obtain the average value over all those frames. On top of that, SSNR shows higher correlation with subjective metrics than SNR (Hansen and Pellom, 1998). The SSNR is computed as the mean of the SNR over different segments of fixed length. Its formula is:

$$\text{SSNR} = \frac{1}{K}\sum_{k=1}^{K} 10\log_{10}\left(\frac{\sum_{n=kN}^{(k+1)N-1}\mathrm{X}(n)^2}{\sum_{n=kN}^{(k+1)N-1}[\mathrm{X}(n)-X(n)]^2}\right) \tag{2.8}$$

where X is the clean signal, X the enhanced signal, K the number of frames and N the length of each segment. The numerator refers to the signal, by using only the

clean signal, which has no noise. The denominator refers to the remaining noise after the enhancement, by computing the difference between the clean signal, which has no noise, and the enhanced signal. Note that the SNR for each segment is computed using the logarithm in base ten. Alternatively to the SSNR, measures such as the Signal-to-Distorsion Ratio (SDR) or the Scale Invariant Signal-to-Distortion Ratio (SI-SDR) can be used. These measures are other representations of the ratios between the signal and noise of an audio (Le Roux et al., 2019).

The three remaining objective measures are the computed SIG, computed BAK and computed OVL, commonly found in literature as CSIG, CBAK and COVL respectively. They were developed by Ding et al., 2015 based on SIG, BAK and OVRL. Therefore, CSIG, CBAK and COVL are objective measures which again focus on the speech signal, the background noise and the overall quality respectively. These metrics are within the range 1 to 5, with higher values being more desirable. Current state of the art models have values around 4.5, 3.75 and 4 for CSIG, CBAK and COVL respectively. The three metrics are computed using the PESQ, SSNR, weighted spectral slope (WSS) and the log-likelihood ratio (LLR) mean.

The WSS measures the difference in the spectral slopes of the clean and enhanced signals. The spectral slope is the rate of change of the magnitude spectrum of a speech signal. It is computed for different segments of the signal. After that, the mean over the segments is obtained. The LLR mean measures the distortion in linear predictive coding coefficients of clean and enhanced signals. Linear predictive coding is based on modeling the speech signal at a time point, based on the speech signal at previous time points. LLR computes the linear predictive coding coefficients for both the clean and enhanced signals. Then, the logarithm of the coefficients ratio is computed for each segment. After that, the mean over all the segments is obtained. The CSIG, CBAK and COVL equations are:

$$\text{CSIG} = 3.093 - 1.029\text{LLR} + 0.603\text{PESQ} - 0.009\text{WSS} \tag{2.9}$$

$$\text{CBAK} = 1.634 + 0.478\text{PESQ} - 0.007\text{WSS} + 0.063\text{SSNR} \tag{2.10}$$

$$\text{COVL} = 1.594 + 0.805\text{PESQ} - 0.512\text{LLR} - 0.007\text{WSS} \tag{2.11}$$

Besides measuring the quality and intelligibility of the enhanced signal, it is common to analyse if speech enhancement models can be used in real time. This is measured by the Real-Time-Factor (RTF) (Pratap et al., 2020). The RTF is the ratio between the time taken to process an input sample and the input duration. For example, if an audio lasts five seconds, the RTF of the model on that sample is computed as the time it takes to process that audio divided by five. A real time system is one with RTF not greater than one. In the case of the five second audio, it would mean that the model processes the audio in a maximum of five seconds. RTF can be considered a metric of the efficiency of the model. To measure the efficiency of models other factors are taken into account. Some examples are the number of trainable parameters of the model, the number of operations performed by the model or the size of the model.

2.6 Speech enhancement models

In this section, some of the models with the elements explained in previous sections are reviewed. Note that as most models combine several elements of those explained,

a classification of the models becomes a difficult task. Most of the speech enhancement models employ an encoder-decoder structure with multiple convolutions. In the last years, attention mechanisms and generative adversarial networks have gained popularity. They are architectures which are being explored at the moment.

X. Lu et al., 2013 introduced the use of autoencoders in speech enhancement. The use of the encoder-decoder structure quickly became a widespread practice in speech enhancement and it is currently present in the vast majority of speech enhancement models. Denoising autoencoders (Vincent et al., 2010) generally lead to good results in speech enhancement. C. Yu et al., 2020 developed a Denoising AutoEncoder with Multibranched Enconder (DAEME). In this model, the data is divided into clusters and each cluster is used as input of an encoder. Hence, each encoder learns a different distribution. Each encoder can fit better the data of its corresponding cluster. The decoder combines the output of the multiple encoders. Despite being an innovative idea, it did not achieve outstanding results.

Recurrent Neural Networks were one of the firsts approaches used in speech enhancement due to their ability to capture temporal dependencies (Zheng et al., 2023). Weninger et al., 2015 perform speech enhancement using LSTM networks. They also explore the use of BiLSTM, but do not obtain remarkable results. Z. Chen et al., 2015 explore the use of BiLSTM to create a model that performs simultaneously speech enhancement and automatic speech recognition, obtaining better results than previous models.

Wichern and Lukin, 2017 study the use of BRNN in real time speech enhancement. The main problem of BRNN is that they take long time to train and are bidirectional models. As future information is required to enhance previous elements, they are not suited for speech enhancement in real time. The authors propose two strategies to cope with the two problems. To solve the high computational cost, they propose using block-wise operations. For the second problem, they suggest replacing the second direction going from future to current event by a look-ahead strategy (C. Wang et al., 2016). The look-ahead strategy is based on using information about near future for the current learning. Hence, only little information about the future is required at each time point. Their model is faster than traditional BRNN but does not offer significantly good results. Also, the look-ahead strategy does not solve the real time problem. Valin, 2018 employs a GRU architecture in their speech enhancement model. GRUSE (Cámbara et al., 2022) employs an encoder-decoder structure with GRU in their latent space. In the same work, the authors also propose RESSE, which has a similar structure but employs residual blocks and convolutions in its latent space.

Convolutional recurrent neural networks (CRN or CRNN) have also been used in speech enhancement. Tan and Wang, 2018 define a CRN which has an encoder-decoder structure. In the encoder and decoder, convolutional layers are used. In the latent space, several LSTM are placed. Similarly, Tan and Wang, 2019 explore this strategy and study the effect of more than one decoder, code and encoder, focusing on real and imaginary features of the audios. Their results show that using two decoders pose better results. On this basis, many models include two decoders in their structure.

Tan and Wang, 2020 follow such line of work and define the Gated Convolutional Recurrent Network (GCRN). The GCRN uses one encoder and two decoders, one for

the real part and other for the imaginary part of the signal. The encoder and decoders contain convolutional layers that connect via skip connections. In the latent space, two layers of LSTM are placed. Le et al., 2021 develop a model called Dual Path Convolution Recurrent Network which uses an encoder and a decoder again with convolution layers and skip connections. The main difference is that the authors do not use two LSTM layers, but an intrachunk with a BiLSTM and an interchunk with LSTM. The intrachunk is used to learn short range dependencies, while the interchunk learns long range dependencies. The interchunk is placed after the intrachunk in the latent space. Using intrablocks to get local dependencies followed by interblocks that get global information, has also become a widespread trend in speech enhancement.

A popular model that combines convolutional layers and recurrent layers is Demucs (Defossez et al., 2020). Demucs is one of the most famous architectures in speech enhancement. It is based on an encoder-decoder structure in which both the encoder and decoder use convolutions and skip connections between layers of the encoder and decoder, obtaining a U-shape architecture. In the latent space, LSTM is employed. The model allows to use unidirectional LSTM for real time speech enhancement tasks and BiLSTM for non causal tasks. Demucs is commonly used as a model against which the performance of new architectures is compared. Currently, the results of the new architectures exceed the results of Demucs. However, at the time of its publication it achieved results that were very competitive with the state of the art models.

Although recurrent neural networks are commonly used in speech enhancement because of their ability to capture temporal dependencies, they suffer from slow training and high computational cost. Convolutional neural networks offer a plausible solution to this problem by allowing parallel computation and therefore being faster (Mehrish et al., 2023). This is crucial if the speech enhancement is done in real time. Currently most of the speech enhancement models use convolutions. Wave-U-Net (Macartney and Weyde, 2018) is an example of architecture that employs multiple convolution layers in the U-net architecture. Wave-U-Net is a popular model in speech enhancement and is one of the models against which the performance of models is also compared.

Pandey and Wang, 2019 propose a model with an encoder-decoder structure that is based on a temporal convolution network. The encoder and decoder contain convolutions and the latent space dilated convolutions. This model allows for real time speech enhancement. A modification of this model is proposed by A. Li et al., 2020. The authors defined a model with two encoder-decoder structures that focus separately on the magnitude and the complex spectrum of the signal. All of the elements of their model are based on TCN. The PHASEN model (Yin et al., 2019) also separates the signal in two paths: one for the magnitude and other for the phase. Both structures are defined by multiple convolutional layers. PHASEN model is also used for performance comparison of new models.

Since the definitions of transformers (Vaswani et al., 2017), attention mechanisms have also become a common practice in speech enhancement models. Many reference models for the performance comparison include attention mechanisms in their architectures. One example is the Two Stage Transformer Neural Network (TSTNN) (K. Wang et al., 2021). The TSTNN employs a two stage transformer consisting of a

local transformer and a global transformer. The local transformer is applied to different parts of the input, obtaining local or short range dependencies. Then, the global transformer fuzzes those outputs, obtaining global or long range dependencies.

Oostermeijer et al., 2021 propose a variation in which only local attention mechanisms are employed. Hence, the computational cost is reduced significantly. However, the long range dependencies are not captured in their lightweight model, compromising its performance. Another model which focus on local dependencies is the SETransformer (W. Yu et al., 2022). The SETransformer has an encoder-only structure. The encoder is similar to the original encoder of the transformer architecture, but includes two modifications in each layer of the encoder: the feed forward sublayer is replaced by a convolutional sublayer and before the multi-head self-attention a local LSTM module is placed. This module is the main responsible for the local focus of the model.

Conversely, the model developed by Z. Wu et al., 2020 uses attention mechanisms to obtain global dependencies. Their model is a two branch structure in which one of the branches uses attention to obtain global dependencies, and the other uses convolutions to get local dependencies. As CNN and attention mechanisms have given good results for speech enhancement it is common to combine them in speech enhancement. For example, Lin et al., 2021 propose a multi-stage strategy for speech enhancement, in which each part employs a self-attention and a temporal convolutional network block.

The Dual Path Transformer Full-Band Sub-Band network (DPT-FSNet) (Dang et al., 2022) also employs an encoder-decoder structure with convolutions and attention blocks. Based on the TSTNN (K. Wang et al., 2021) they employ a two stage transformer with an intratransformer and an intertransformer. The intratransformer captures the local dependencies while the intertransformer captures the global ones. The Full-Band and Sub-Band terms in the name of the model refer to the global and local dependencies respectively. The term band refers to the frequency band. The dual path transformer is placed in the latent space between the encoder and the decoder. The encoder and the decoder utilize several convolution layers. The results obtained by this model are remarkable. The DPT-FSNet is also commonly used for performance comparison of new architectures.

Another standard model for performance evaluation, which uses a similar structure to DPT-FSNet and has outstanding results, is the DB-AIAT (G. Yu et al., 2022). The Dual-Branch Attention-In-Attention transformer (DB-AIAT) has two branches, each with an encoder-decoder structure. The two paths in this case correspond to the magnitude and complex features. The complex branch includes information about the real and imaginary parts of the audio. The magnitude path suppresses almost all the noise. The complex beam focuses on fine-grained details that the first branch may have not captured. The two paths have an encoder and an attention-in-attention module in the latent space. The attention-in-attention module is conformed by four encoder-only transformers and an adaptive hierarchical attention module that combines the transformers outputs. The magnitude branch has then a decoder, while the complex branch has two decoders, one for the real part and other for the imaginary part of the signal.

The Multi-view Attention Network for Noise ERasure (MANNER) model (H. J. Park et al., 2022) is also one of the models in speech enhancement that shows noteworthy

results and is used for performance comparisons. It is based in an encoder-decoder structure with convolutions and multiview attention blocks in each layer. The multiview attention block does not only have the local and global branches of its predecessors, but it adds a third branch known as channel branch. The channel attention module learns the importance of each feature of the input data.

Other famous architecture in speech enhancement is based on combining attention mechanisms with the U-shape encoder-decoder. The model presented by Y. Li et al., 2023, uses a U-shaped architecture in which several attention layers alternate with feed forward layers. It uses three types of attention blocks: time attention, low frequency band attention and high frequency band attention. The U-former (Y. Fu et al., 2022) combines the U-net and conformers with dilated convolutions. The architecture is dual-path: one path focuses on the magnitude and is responsible for the majority of the enhancement and the other focuses on the complex spectrum and is only used for small details, as in DB-AIAT. The conformers are placed in the latent space, between the encoder and the decoder. Similarly, the SE-Conformer (E. Kim and Seo, 2021) utilizes a U-net structure and has a conformer in the latent space. The SE-Conformer model is other of the standard models used for performance comparison. The model proposed by Kong et al., 2022 is similar to the SE-Conformer but employs a transformer in the latent space, instead of a conformer.

Pascual et al., 2017 developed the first GAN used in speech enhancement, known as Speech Enhancement Generative Adversarial Network (SEGAN). In SEGAN, the generator and discriminator networks employ convolutions and the generator network exhibits an encoder-decoder structure. SEGAN is other standard model used for performance analysis. Although new models outperform significantly this model, the SEGAN model is considered quite important, due to the introduction of GANs in speech enhancement. Pascual et al., 2019 extended the SEGAN architecture, developing the SEGAN+ model. The authors included skip connections and larger convolutional strides. This translates into a reduction of the model size, obtaining a more efficient architecture and more stable results.

Other popular GAN model in speech enhancement is MMSEGAN (Soni et al., 2018). This model focuses on the TF domain instead of the time domain of SEGAN. On top of that, it uses MMSE error for the loss computation, hence its name. Previously, many models used MAE losses. The use of MMSE gives better results. S.-W. Fu et al., 2019 propose the MetricGAN model. MetricGAN uses perceptual metrics for the discriminator loss computation. Several metrics related to speech signal intelligibility or quality, such as STOI and PESQ, are suggested. The goal is to obtain a model that learns to enhance signals using human-like criteria to update the parameters. S.-W. Fu et al., 2021 improve the MetricGAN model, constituting the MetricGAN+. MetricGAN+ includes noisy data into the learning process of the discriminator. A term of the discriminator loss takes into account the noisy data, in addition to the clean and enhanced data. MetricGAN and MetricGAN+ show outstanding results and are examples of models frequently used to analyse the performance of new models.

S.-W. Fu et al., 2022 propose a neural network which only needs the noisy speech to perform speech enhancement. It does not require the clean signal nor the noises to compare the output to the clean signal or to the input signal with suppressed noise. Therefore, the authors call it MetricGAN-U which stands for MetricGAN-Unsupervised. Other example of generative adversarial network for speech enhance-

ment is the M-CRGAN-MSE (Zhang et al., 2020) in which convolutional recurrent networks and MSE are employed. A problem of GANs in speech enhancement is that they tend to focus on local speech features and not so much on global features. Therefore, long range temporal dependencies may not be captured. CPGAN (G. Liu et al., 2020) intends to solve this by using two discriminators: one for the local dependencies and other for the global dependencies.

In addition to autoencoders and convolutions, attention mechanisms and generative adversarial networks have proven to give good results in speech enhancement. Recently, many studies have focused on mixing both of them, creating new competitive models. Some examples of this strategy are the Self-Attention SEGAN (SASEGAN) (Phan et al., 2021) and the Multiscale Attention Metric GAN (MAMGAN) (Guo et al., 2023). The SASEGAN has a generator with an encoder-decoder structure. The encoder and the decoder present skip connections in a U-shape form. The encoder and the decoder also include self-attention layers coupled with convolutional layers. The discriminator structure is similar to the encoder part of the generator. MAMGAN also uses attention-blocks in the discriminator and the generator, and also has an encoder-decoder structure in the latter. By contrast to SASEGAN, the attention blocks in the generator of MAMGAN are only placed in the latent space.

2.7 Best state of the art models in speech enhancement

This section explains a family of neural networks that yields the best results in speech enhancement. Three models are explored: CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024) and MPSE-Net (Y.-X. Lu et al., 2023). The three models are considered a family of neural networks as they share the majority of their elements. This section includes a deep description of the three models. It is important to understand each element of their structure as they define some of the best architectures in speech enhancement and are crucial for the development of this project.

The three models are generative adversarial networks, in which the generator has an encoder-decoder structure and attention mechanisms in the latent space. On top of that, the three generators use two decoders, one of which is always a magnitude decoder. The magnitude decoder estimates a mask that is applied to the noisy signal magnitude. Figure 2.8 and table 2.1 give the information about the common structure and main differences of these architectures, respectively. The common structure and differences are analysed in what follows.

	Attention mechanism	Decoder II	Generator loss	Discriminator loss
CMGAN	Conformer	Complex	Eq. 2.16	Eq. 2.22
TPTGAN	Transformer	Complex	Eq. 2.23	Eq. 2.24
MP-SENet	Conformer	Phase	Eq. 2.25	Eq. 2.26

Table 2.1: Main differences between CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024) and MP-SeNet (Y.-X. Lu et al., 2023). The three first columns refer to the generator, while the last one to the discriminator.

The main differences between the models are within the generator due to its higher complexity and importance in the denoising task. As shown in table 2.1, the differences within the generator are three. First, the attention mechanism used in N

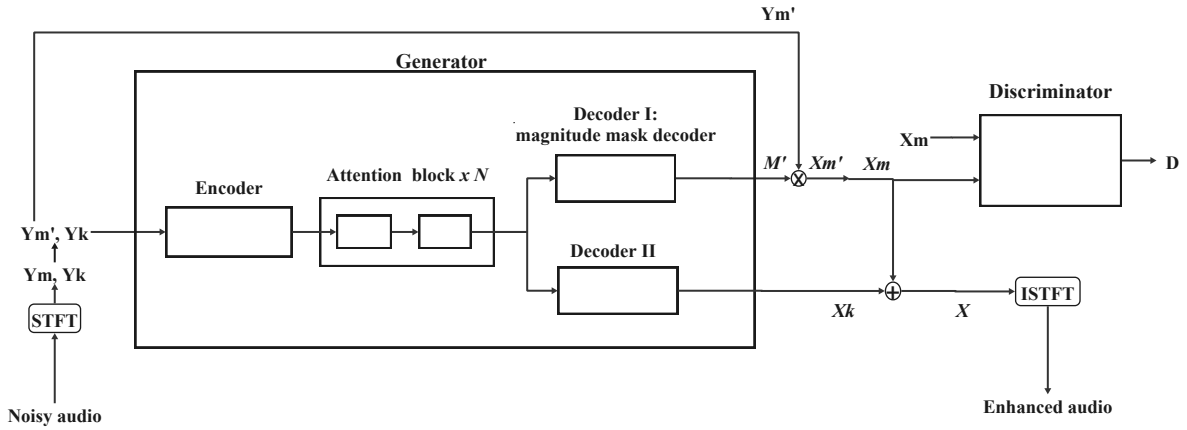

Figure 2.8: Common structure of CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024) and MP-SENet (Y.-X. Lu et al., 2023). Y is the noisy signal, X the clean signal and X the enhanced signal. M is the estimated magnitude mask. The subindexes refer to the magnitude (m) and the additional element (k) which could be either the phase or the real and imaginary components. The apostrophe indicates compressed feature.

blocks in the latent space of the encoder-decoder can be either a transformer or a conformer. In both cases, each block is conformed by two attention mechanisms. In the conformer block, one conformer is used for the frequency and the other for the time domain. In the transformer block, a first transformer is used to obtain local dependencies and the second to obtain global dependencies.

After the attention blocks of the latent space two decoders are used. The first decoder is always used for the magnitude, while the second can be either for the complex (real and imaginary) part of the signal, or for the phase. The last dissimilarity in the generator comes from the loss equation, which are shown in equations 2.16, 2.23 and 2.25. Besides the differences in the generator, the loss equations of the discriminator can also be dissimilar, as equations 2.22, 2.24 and 2.26 illustrate. It should be noted that this is the only difference in the discriminator as the three models use the same discriminator architecture. Common elements and differences in the architecture elements are explained first. Then, the losses are analysed.

The three models work in the TF domain. A STFT is applied to the waveform of the input before entering the model. A transformation is then performed to obtain the magnitude and the real and imaginary or phase components. A power compression is applied to the magnitude of the noisy signal, after performing the STFT and prior to the generator. Power compression is used because human perception is more sensitive to loud sounds than to quiet ones. By applying power compression to the magnitude, the importance of these two types of sounds can be adjusted. The compression factor used in these models is 0.3 as Wilson et al., 2018 suggest. The power-law compression is based on raising the magnitude to the power indicated by the compression factor. That means:

$$Y'_m = Y^c_m \tag{2.12}$$

$$X'_m = X^c_m \tag{2.13}$$

$$M' = M^c \tag{2.14}$$

where $c = 0.3$ is the compression factor, Y_m the magnitude of the noisy signal, X_m the magnitude of the enhanced signal and M the mask estimated by the magnitude mask decoder. The compressed noisy signal magnitude, Y'_m, and the other component values, Y_k, are concatenated and fed to the model. The output of the generator is used to obtain the final reconstructed signal. The two decoders output information about the compressed magnitude mask, M', which is applied to the compressed noisy signal magnitude, Y'_m, and about the enhanced signal phase or real and imaginary components. A decompression of the compressed enhanced magnitude, X'_m, takes place. The decompressed magnitude of the enhanced signal, X_m, and the other enhanced component, X_k, are combined to obtain the spectrogram of the enhanced signal, X. After that, through the ISTFT, the enhanced signal in time domain is obtained. This whole process is shown in figure 2.8.

The encoder and decoders of the generator are composed by several convolution blocks. These blocks include convolution layers, instance normalization (Ulyanov et al., 2017) and Parametric Rectified Linear Unified (PReLU) activation (He et al., 2015). In the case of TPTGAN, layer normalization, instead of instance normalization, is used. Instance Normalization normalizes across each channel in each training example. Note that this differs from layer normalization in which the normalization is done for all channels in each sample and from batch normalization in which the normalization is done for each channel in a batch of samples. The PReLU activation function is defined as shown in equation 2.15. This equation is similar to the Leaky ReLU activation function, but has one major difference. In the Leaky ReLU activation function, the x is multiplied by a constant hyperparameter value given that it has a negative value. Conversely, in PReLU the α value is a learnable parameter. PReLU has become a standard activation function in speech enhancement. PReLU formula is:

$$\mathrm{PReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases} \tag{2.15}$$

A dilated dense net (Pandey and Wang, 2020) is defined from the second to the fifth block of the encoder, with dilation rates of 1, 2, 4 and 8, respectively. The 3x3 kernel is applied over the input data with gaps between its values, controlled by the dilation rates. This increases the receptive field, allows for multi-scale feature learning and increases efficiency.

The blocks are identical in the encoder and the decoders. One of the differences between the encoder and the decoders, is that in the first block of the encoder, the number of channels is increased from two or three (magnitude and phase or magnitude, real and imaginary) to sixty-four. In the remaining blocks, the number of output channels is kept constant as sixty-four. However, the number of input channels increases as residual connections are employed. The output of each block is concatenated to the input of the block. Therefore, the number of input channels correspond to the number of input channels of the previous block plus sixty four. Residual connections help with the vanishing gradient problem and make sure that each block can see all the information that has been processed by previous blocks. These connections allow the backpropagation to previous layers without significant degradation. Hence, residual connections facilitate training deep networks with high number of layers.

In the last convolution block of the encoder, the frequency dimension is halved to

reduce the complexity of the attention mechanisms in the latent space. Note that the original dimension of the frequency correspond to the number of points used to perform the STFT and is therefore one of the specific hyperparameters used in speech enhancement. Figure 2.9 shows the structure of the encoder. In such figure, and the remaining ones of this section, B refers to the batch size, T to time frames, F to frequency bins and C to channels. F^* is the downsampled frequency.

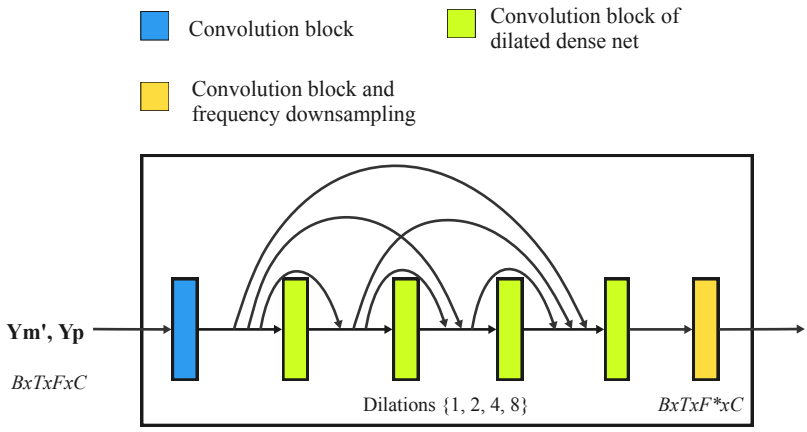

Figure 2.9: Common structure of encoder in CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024) and MP-SENet (Y.-X. Lu et al., 2023). Ym is the magnitude of the noisy signal. Yp is the phase of the noisy signal. The apostrophe indicates compressed feature. B refers to the batch size, T to time frames, F to frequency bins and C to channels. F^* is the downsampled frequency

As sixty-four is the number of output channels of the encoder, the attention mechanisms in the latent space also have sixty-four input and output channels. The attention mechanisms can be either conformer or transformers as table 2.1 indicates. Each conformer or transformer block consist of two conformers or transformers respectively. In the case of the conformers, the first one is used to obtain time dependencies, while the second one focus on the frequency. Thus, they are called time and frequency conformers respectively. To focus on one feature or the other, reshaping the input is necessary. This is shown in figure 2.10. On the other hand, the transformer block employs an intratransformer to get local dependencies and after that an intertransformer to obtain global range dependencies. Reshaping is not necessary in this case as figure 2.11 shows. The structures of figures 2.10 and 2.11 are used four times serially in the latent space of the generator.

Both, the conformers and transformers have the traditional structure in an encoder-only mode. They perform the feature extraction while maintaining the dimensionality of the data. The feature extraction determines the most important features of the audio. If the enhancement is done correctly, these features include clean audio attributes and exclude noise attributes. Hence, when the decoders reconstruct the audio, noise is not present.

The number of output channels in the attention structures is sixty-four. Thus, contrary to the encoder, the number of input channels in the decoders is sixty four. Therefore, a first convolution block that increases the number of channels is not necessary. The dilated dense net is in this case conformed by the first to fourth block,

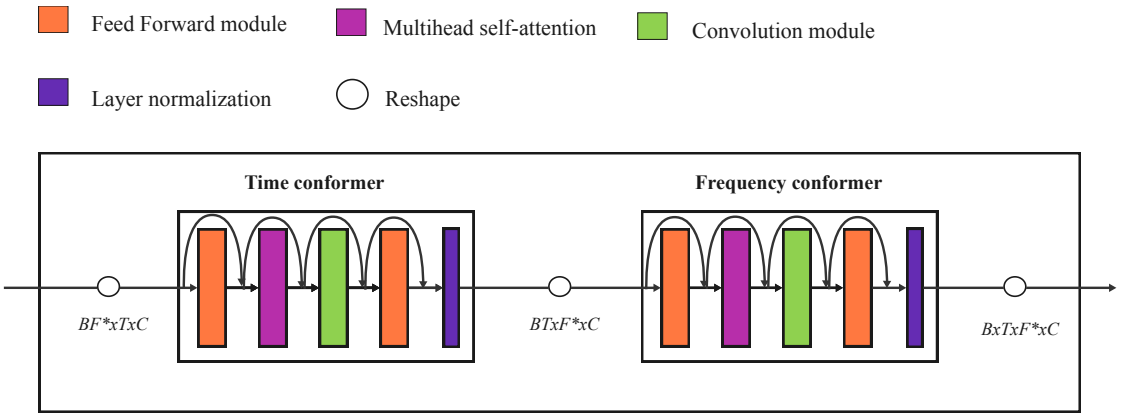

Figure 2.10: Conformer block of CMGAN (Cao et al., 2022) and MP-SENet (Y.-X. Lu et al., 2023). B refers to the batch size, T to time frames, F^* to the downsampled frequency bins and C to channels.

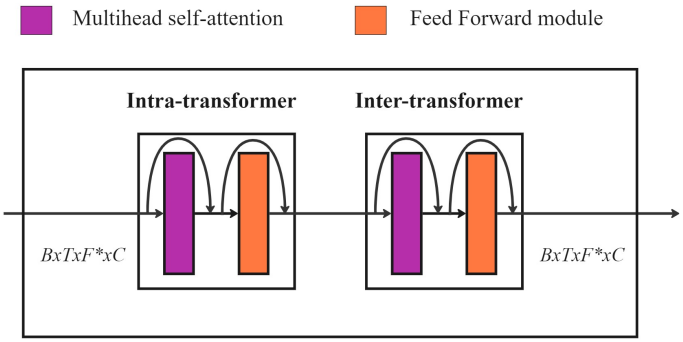

Figure 2.11: Transformer block of TPTGAN (Z. Liu et al., 2024). B refers to the batch size, T to time frames, F^* to the downsampled frequency bins and C to channels.

and uses the same dilation rates. After the dilated dense net, a convolution block is used to upsample the frequency dimension. This means that the dimension of the frequency is doubled to reconstruct the original frequency dimension. Finally, a last convolution layer is employed, only followed by an activation function in the mask decoder (see the difference between figures 2.12 and 2.13) .

In MP-SENet, the last activation function of the mask decoder corresponds to a learnable sigmoid instead of PReLU, as the authors discover that using the learnable sigmoid function improves the performance significantly. Also, in MP-SENet the last convolution layer of the second decoder is replaced by two separately convolution layers that are used in parallel and which output is merged in the end. This is because MP-SENet second decoder focuses on the phase. The last convolution layers used in parallel obtain the real and imaginary components separately. Then the arcotangent operation is performed to obtain the phase of the enhanced signal.

The discriminator compromises four convolution blocks. Then, an adaptative maxpool layer is used to reduce the dimensionality. Finally, two linear layers are employed. The linear layers are employed as in standard CNN to perform the classifica-

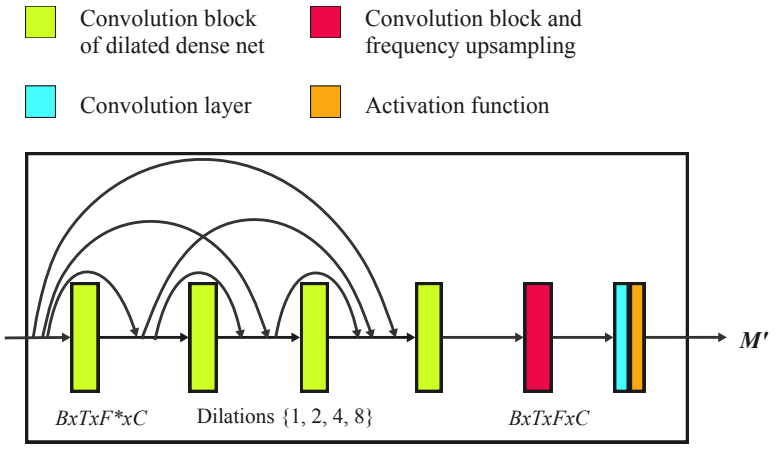

Figure 2.12: Common structure of magnitude mask decoder in CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024), and MP-SENet (Y.-X. Lu et al., 2023). M is the enhanced magnitude mask. The apostrophe indicates a compressed feature. B refers to the batch size, T to time frames, F to frequency bins and C to channels. F^* is the downsampled frequency.

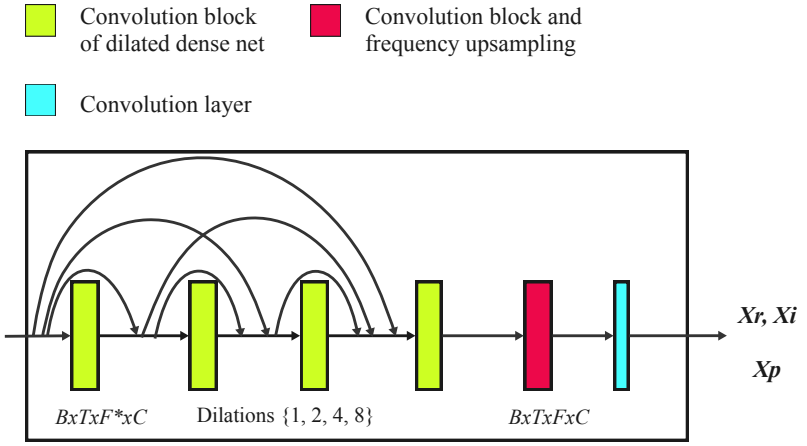

Figure 2.13: Common structure of complex or phase decoder in CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024), and MP-SENet (Y.-X. Lu et al., 2023). *Xr*, *Xi* and *Xp* are the real part, imaginary part and phase of the enhanced signal respectively. CMGAN and TPTGAN focus on real and imaginary parts, MP-SENet on the phase. B refers to the batch size, T to time frames, F to frequency bins and C to channels. F^* is the downsampled frequency.

tion. In this case the classification is between real clean audios and fake or enhanced audios. The first linear layer uses PReLU activation function and drop out. The drop out is used to reduce the variance of the discriminator. Note that dropout is not only present at this point of the model, but also in the latent space of the generator. This is because conformers and transformers use it. The activation function of the last linear layer corresponds to a learnable sigmoid. Employing a learnable sigmoid activation at the end of the discriminator helps adjusting the decision boundary between

real and fake data and stabilizes convergence. This is a common choice in GANs.

In each block of the discriminator, spectral normalization is employed after the convolution or linear layer. Spectral normalization (Miyato et al., 2018) helps stabilizing the discriminator training. This is crucial as the learning process of GANs is defined by its instability. In spectral normalization, the maximum weight is used to normalize all the weights, therefore limiting the scaled weights to a maximum value of one. This strategy stabilizes the learning process of a neural network, favouring its convergence. Figure 2.14 shows the common structure of the discriminator. Note that such structure is exactly the same for the three models.

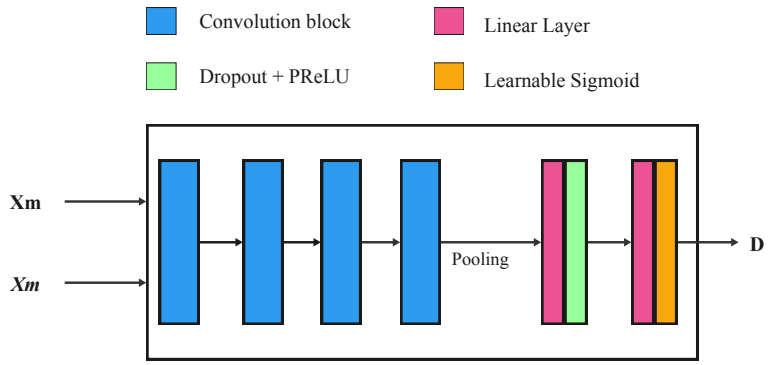

Figure 2.14: Common structure of discriminator in CMGAN (Cao et al., 2022), TPTGAN (Z. Liu et al., 2024) and MP-SENet (Y.-X. Lu et al., 2023). Xm is the magnitude of the clean signal. *Xm* is the magnitude of the enhanced signal. D is the discriminator probability output.

The discriminator uses the magnitude of the enhanced signals and clean signals as input and tries to discriminate between them. The discriminator output D corresponds to the $\mathcal{Q}_{\text{PESQ}}$, a normalized value of the PESQ. Therefore, the $\mathcal{Q}_{\text{PESQ}}$ range is $[0, 1]$, where the lowest value corresponds to an extremely noisy signal and the highest value to the clean signal. The discriminator is trained to give the best enhanced signals a $\mathcal{Q}_{\text{PESQ}}$ closer to one, and the worst enhanced signals a $\mathcal{Q}_{\text{PESQ}}$ closer to zero. The generator goal is to produce enhanced signals that obtain a $\mathcal{Q}_{\text{PESQ}}$ of one and are therefore, indistinguishable from clean signals. The discriminator goal is to, even for almost perfectly enhanced signals, be able to discriminate between clean and enhanced signals.

TPTGAN has an extra input of Y_m corresponding to the magnitude of the noisy signal. This is because including noisy signals for the discriminator training process has proved to give better results (Kawanaka et al., 2020, S.-W. Fu et al., 2021). Using noisy signals, facilitates the detection and $\mathcal{Q}_{\text{PESQ}}$ computation of poorly enhanced signals.

In each model different losses for the generator and the discriminator are used. CMGAN uses a generator loss, $L_{G_{CMGAN}}$, with three terms:

$$L_{G_{CMGAN}} = L_{TF} + L_{Time} + L_G \tag{2.16}$$

where L_{TF} is the loss in the time-frequency domain, that is between the spectrograms of the clean and enhanced signals, L_{Time} is the loss in the time domain, that is in the

waveform between the clean and enhanced signal, and L_G is the loss of the generator, referring to its enhancement capabilities. The L_{TF} can as well be decomposed in:

$$L_{TF} = L_{Mag} + L_{RI} \tag{2.17}$$

where L_{Mag} is the loss in the magnitude space and L_{RI} is the loss in the real-imaginary space. These are obtained by the two decoders of the generator. They can be expressed in expectancy terms:

$$L_{Mag} = \mathbb{E}_{X_m,\hat{X}_m}[||X_m - \hat{X}_m||^2] \tag{2.18}$$

$$L_{RI} = \mathbb{E}_{X_r,\hat{X}_r}[||X_r - \hat{X}_r||^2] + \mathbb{E}_{X_i,\hat{X}_i}[||X_i - \hat{X}_i||^2] \tag{2.19}$$

where X is the clean signal and $\hat{X}$ is the enhanced signal, and the subscripts m, r and i refer to the magnitude, real and imaginary values of each signal. The second term of the equation 2.16 can also be expressed in expectancy terms as:

$$L_{Time} = \mathbb{E}_{x,\hat{x}}[||x - \hat{x}||_1] \tag{2.20}$$

where x is again the clean signal and $\hat{x}$ the enhanced signal. They refer to the waveforms of the signals in the time domain. This is, before applying the STFT to the input and after applying the ISTFT to the ouptut. This ensures that the signal is enhanced focusing also in time domain (Abdulatif et al., 2021). The more similar the enhanced and clean signals are, the lower the differences in equations 2.18, 2.19 and 2.20. Finally, the third term of the generator loss, shown in equation 2.21, refers to the loss of the generator, related to the discriminator judgement of the enhanced signal. Its equation is:

$$L_G = \mathbb{E}_{X_m,\hat{X}_m}[||D(X_m, \hat{X}_m) - 1||^2] \tag{2.21}$$

The generator purpose is to enhance the signal to fool the discriminator so that it is not able to distinguish between clean and enhanced signal. By minimizing L_G, the generator learns to generate enhanced signals that the discriminator detects as real clean signals. Each of the terms of the generator loss is multiplied by the same coefficients which corresponds to 0.25 for the $L_{Mag}, L_{RI}, L_{Time}$ and L_G. Apart from a generator loss, the CMGAN model has a discriminator loss. Such loss is:

$$L_{D_{CMGAN}} = \mathbb{E}_{X_m}[||D(X_m, X_m) - 1||^2] + \mathbb{E}_{X_m,\hat{X}_m}[||D(X_m, \hat{X}_m) - Q_{PESQ}(X_m, \hat{X}_m)||^2] \tag{2.22}$$

where Q_{PESQ} is the normalized PESQ score. As equation 2.22 shows, the loss of the discriminator compromises two terms. The first term refers to the clean signal, while the second refers to the enhanced signal. If the input signal in the discriminator is clean, the discriminator output should be one. Therefore, the first term should be minimized. For the second term, where the input of the discriminator is an enhanced signal, the output should be a value which corresponds to how well was the signal enhanced. By minimizing the $L_{D_{CMGAN}}$ the discriminator learns to give values close to zero to poorly enhanced signals and close to one to clean and properly enhanced signals. The terms of the discriminator loss are not multiplied by any coefficient.

TPTGAN has similar losses to those of CMGAN. The losses of the generator, $L_{G_{TPTGAN}}$, and discriminator, $L_{D_{TPTGAN}}$, are indicated in equations 2.23 and 2.24 respectively:

$$L_{G_{TPTGAN}} = L_{G_{CMGAN}} \tag{2.23}$$

$$L_{D_{TPTGAN}} = L_{D_{CMGAN}} + \mathbb{E}_{X_m,Y_m}[||D(X_m, Y_m) - Q_{PESQ}(X_m, Y_m)||^2] \tag{2.24}$$

The loss of the generator is equal to the loss of the CMGAN generator. Therefore, no explanation of it is required. The coefficients in TPTGAN generator loss are 0.9, 0.1, 0.2 and 0.01 for the $L_{Mag}, L_{RI}, L_{Time}$ and L_G respectively. The discriminator loss however is slightly different as it adds a third term with respect to equation 2.22. This difference arises from the use of noisy signals Y, as their magnitude is also used as input of the discriminator. The noisy signal should obtain a Q_{PESQ} score close to zero as they are far from being clean signals. Again, the discriminator loss terms are not multiplied by any coefficient.

The last model of this section is MP-SENet. The losses of the generator, $L_{G_{MP-SENet}}$, and discriminator, $L_{D_{MP-SENet}}$, are indicated in equations 2.25 and 2.26 respectively:

$$L_{G_{MP-SENet}} = L_{G_{CMGAN}} + L_{Phase} \tag{2.25}$$

$$L_{D_{MP-SENet}} = L_{D_{CMGAN}} \tag{2.26}$$

The loss of the discriminator is exactly the same as the CMGAN discriminator loss, including the absence of coefficients, and therefore does not require any explanation. However, the generator loss is slightly different to the CMGAN one, as it includes a fourth term. This term corresponds to the phase features. Y.-X. Lu et al., 2023 indicate in their work that conventional phases losses, as L1 or L2 errors, are not appropriate because of phase wrapping. Phase wrapping occurs when the phase angle of a signal is not in the range $[0, 2\pi]$, which can happen in audio processing. When the angle is not in the range mentioned, distortions are common. Based on the work of Ai and Ling, 2023, the MP-SENet developers propose a phase loss with three terms as:

$$L_{Phase} = L_{IP} + L_{GD} + L_{IAF} \tag{2.27}$$

where L_{IP}, L_{GD} and L_{IAF} are the instantaneous phase loss, group delay loss, and instantaneous angular frequency loss respectively. Each term, as indicated in equations 2.28, 2.29 and 2.30, uses antiwrapping functions, f_{AW}, to avoid the error expansion issue caused by phase wrapping. The antiwrapping function ensure that the angle x is within the range $[0, 2\pi]$ by subtracting 2π to x given that $x > 2\pi$. It is applied element-wise in each of the following equations:

$$L_{IP} = \mathbb{E}_{X_p,\hat{X}_p}[||f_{AW}(X_p - \hat{X}_p)||_1] \tag{2.28}$$

$$L_{GD} = \mathbb{E}_{\Delta_{DF}(X_p,\hat{X}_p))}[||f_{AW}(\Delta_{DF}(X_p - \hat{X}_p))||_1] \tag{2.29}$$

$$L_{IAF} = \mathbb{E}_{\Delta_{DT}(X_p,\hat{X}_p))}[||f_{AW}(\Delta_{DT}(X_p - \hat{X}_p))||_1] \tag{2.30}$$

where Δ_{DF} is the differential operator in the frequency axis and Δ_{DT} is the differential operator in the time axis. The instantaneous phase loss (L_{IP}) is the difference between the phase of the clean and enhanced signals. The group delay loss (L_{GD}) and the instantaneous angular frequency loss (L_{IAF}) are used to ensure continuity of the predicted phase along frequency and time axis respectively. The coefficients of MP-SENet generator loss are 0.9, 0.1, 0.2, 0.05 and 0.3 for the $L_{Mag}, L_{RI}, L_{Time}, L_G$ and L_{Phase}.

It should be noted that other models, based on these three ones, have been developed. They also obtain outstanding results. Some examples are the SCP-GAN (Zadorozhnyy et al., 2022) and the SEMamba (Chao et al., 2024) models.

SCPGAN is an improvement of CMGAN. It uses CMGAN as a baseline model and improves it with a Self Correcting (SC) discriminator and Consistency Preservation (CP). The self correcting discriminator fixes the problems associated with the unstable discriminator training. This is done by introducing weights in its loss formula. The consistency preservation tries to overcome the undesirable effects of the ISTFT on the enhanced signal. Losses in the frequency domain, such as the magnitude loss or the complex loss, do not take into account the effect of the ISTFT. They compare the features for the clean and enhanced signals before the ISTFT. SCP-GAN modifies CMGAN architecture and loss formulas to take into account the effects of the ISTFT on the losses of the frequency domain.

SEMamba has a similar architecture to that of MP-SENet. It has an encoder, a latent space and two decoders, one for the magnitude and other for the phase. In the latent space, the Mamba model (Gu and Dao, 2024) is implemented instead of conformers. As in the conformer block, two Mamba models are implemented in each block: the first one for the time domain and the second one for the frequency domain. Mamba blocks do not use attention mechanisms, which are computationally expensive, but selective State Space Models (SSMs). SSMs are based on CNN and RNN. SSMs use linear recurrence and global convolutions. Selective SSM increase the efficiency and flexibility of SSM with input-dependent dynamics. Instead of using the basic Mamba model, some modifications are introduced, obtaining an advanced Mamba. This new model allows bidirectional processing, use the consistency loss of SCP-GAN and perceptual contrast stretching (PCS). PCS is used to stretch the magnitude spectrum as a function of the importance of specific frequency bands for human perception. The more important the band for human perception, the more contrast and stretch is applied to the band.

The architectures explored in this section seem to give raise to multiple models with outstanding results. Exploring these architectures appear to be the future trend in speech enhancement.

Chapter 3

Problem Statement

Speech enhancement refers to methods that improve audio quality and intelligibility. Its main goal is to produce cleaner audios that are more pleasant to listen and understand. As a consequence, the important information from audios can be more easily extracted. Speech enhancement methods are designed to suppress background noises, reverberation, and other forms of interference that degrade the audio quality and intelligibility. It involves several techniques and algorithms that try to distinguish the speech signal from the noises of an audio. Those noises can then be removed in order to obtain a clean version of the input audio.

Deep learning is currently the most effective trend in speech enhancement. Previously, other methods, such as statistical methods, were used. However, deep learning models have surpassed by far other methods used in this area. Neural networks offer high flexibility, are easily adapted to work with high amounts of data and can be used in real time. Deep learning models are also able to easily generalize, enhancing audios quite different from those they are trained on. On top of that, deep learning models can be trained to erase only a specific type of noise, offering a solution for domain specific tasks.

Speech enhancement is crucial for several applications, such as human-machine speech interaction. Virtual assistants controlled by voice need to isolate speech signals from background noises. Therefore, a speech enhancement model may be used to isolate the speaker commands. Hearing aids use speech enhancement models to suppress background noises in real time. Speech enhancement can also be used in telecommunication systems, such as phone or video calls, in which a clearer audio implies better communication. Additionally, speech enhancement can be used in automated transcription services, to allow voice-to-text transcription even in noisy environments. Another crucial application of speech enhancement is to improve the hearing experience of people with hearing impairments.

Currently, there is a growing demand for speech enhancement models that are efficient. High performance models have limited applications if they are not efficient. Small models can be integrated in small devices such as phones, improving their properties with respect to users privacy and models latency. Minimizing memory requirement and computational cost is one of the main goals in most technologies, as it is the way to make the models available to general population for daily-life tasks. Some strategies that have been explored to make speech enhancement models

efficient, include model compression with pruning techniques and knowledge distillations strategies.

The main goal of this project is to develop a speech enhancement model that obtains state of the art results and which is efficient. To measure the performance, the most popular objective and subjective metrics in speech enhancement are computed. To measure the efficiency, the number of parameters, the number of operations, the size of the models and the ability to be used in real time applications are measured. There is a trade-off between performance and efficiency. Therefore, by focusing on model efficiency, a slight degradation on the model metrics with respect to state of the art models is expected. The goal of this project can be divided in three subgoals.

The first subgoal of this project is to develop a framework for speech enhancement models training and evaluation. A data preprocessing pipeline is developed and data augmentation methods are implemented. Data augmentation techniques deal with the overfitting problem. Data from the Valentini database is used, as it is one of the most common benchmarks in speech enhancement. Training and evaluation pipelines are also developed in order to treat different models the same way, obtaining a reliable and consistent comparison of results.

The second subgoal of this project is to design an efficient deep learning algorithm for speech enhancement. Several models are explored for this purpose. The models developed in this project are based on the best state of the art models. They combine different elements of deep learning networks such as conformers, convolutions, autoencoders and generative adversarial networks. The models are developed with low number of parameters and of operations, in order to be efficient. Once the algorithms are defined, the models are trained. Several challenges are addressed in this project. Hardware and time limitations affect vastly to the development of the models. Strategies such as gradient accumulation and automated mixed precision are explored. The instability of generative adversarial networks training, also pose difficulties during the development of the project.

The third subgoal of this project is to validate, evaluate and analyse the results obtained for the developed and trained models. To validate the models, a partition is performed on the Valentini train set to obtain a train and a validation set. This partition ensures some differences between train and validation sets, in order to efficiently detect and avoid overfitting. To evaluate the models, the most common objective and subjective metrics are employed. In addition, the computational efficiency is analysed. The contribution of the different elements of the models is also studied. This is done to find out which elements have a greater weight in the efficiency and the performance of the models.

Chapter 4

Development

The following chapter contains the information related to the development of this project. The decisions made and all the reasons after them are explained. First, the resources for the models development are indicated. After that, the dataset employed and preprocessing steps are shown. A different section is used to explain the data augmentation process. Next, the models architecture and losses are explained. Finally, information about the models training and evaluation processes is indicated. The results obtained by the developed models are shown and discussed in the next chapter.

4.1 Resources

For this project NVIDIA GeForce GTX 1080 Ti GPUs of 10 GB are employed. GPU stands for graphics processing unit. The installed driver version is 550.54.15, and CUDA version 12.4 is employed for GPU-accelerated computing tasks. These GPUs are crucial for the computational tasks, reducing the training time for the models significantly.

The programming language of this project is Python[1], as it allows the use of several tools for speech enhancement and for deep learning. Using Python offers several advantages, such as its high versatility and available libraries. The deep learning models are implemented and trained using PyTorch[2]. PyTorch is selected as it offers great flexibility and is the preferred framework for deep learning with audio data. Its dynamic approach for building models enables easier debugging and customization. Torchinfo[3] has also been employed to illustrate information about the models. This is quite useful for model visualization, indicating the number of parameters that each element of the model involves. To count the number of floating point operations per second (FLOPS) of the models, Thop[4] is used.

For audio data, the library Torchaudio[5] from PyTorch is employed. Toarchaudio can be easily integrated in PyTorch and offers easy preprocessing of the audio data. Tor-

[1]Python - https://www.python.org/
[2]PyTorch - https://pytorch.org/
[3]Torchinfo - https://pypi.org/project/torchinfo/0.0.1/
[4]Thop - https://pypi.org/project/thop/
[5]Torchaudio - https://pytorch.org/audio/stable/index.html

chaudio includes pipelines for the evaluation of deep learning models both with objective and subjective metrics. In this project, SQUIM[6] from Torchaudio is employed. SQUIM is defined as a non-intrusive speech assessment which allows for subjective and objective evaluation. Apart from SQUIM, PESQ and STOI metrics are computed using the pesq[7] and pystoi[8] modules from Python. Librosa[9] is another package of Python used for audio analysis. In the present project, librosa has been employed to compute some of the evaluation metrics. IPhython[10] is used to create audio objects. These objects can be used for interactive computing, showing how audios change before and after their enhancement.

Other libraries have been utilized in this project. To obtain graphical visualization of the data, Matplotlib[11] is selected. Seaborn[12] complements Matplotlib, generating more complex graphs with a higher-level interface. Numpy[13] and Scipy[14] are employed for mathematical and scientific operations. Audios are signals with several features, that require mathematical computations. The models developed in this project are TF models. Hence, the number of operations they require to process the audios is significant. Consequently, the tools used for the computations must be efficient. Pandas[15] is selected to handle the dataset. Scikit-learn (Sklearn)[16] is also employed in this project for some of the hyperparameter selection. The tools that Sklearn offer are based on machine learning algorithms.

4.2 Data and preprocessing

To train and evaluate the models, the database VoiceBank + DEMAND is employed. The dataset is already divided into train and test set. In this project such established partition is preserved in order to obtain results that allow a more reliable comparison with other models from the literature. The dataset includes the clean and noisy wav files of the train and test set. Two train sets are available. One includes twenty eight speakers and other fifty six speakers. The latter is chosen. Preprocessing the data obtained from the Valentini database is a crucial step for this project.

Three tsv files are employed to allocate information about the wav files. A first tsv file provides information about the whole Valentini dataset, including the train set of the fifty six speakers and the test set. This file involves information for each sample about the sample identifier, path of the clean wav file, path of the noisy wav file, audio length, language, transcription, recording device, speaker identifier, type of noise and the database where it comes from. A second tsv file is derived from the first one including only the information about the test set samples. This is done in order to later identify the audios that correspond to the test set partition of the database, and be able to respect such partition. Finally, a third tsv file comprises information about

[6] SQUIM - https://pytorch.org/audio/main/tutorials/squim_tutorial.html
[7] pesq - https://pypi.org/project/pesq/
[8] pystoi - https://pypi.org/project/pystoi/
[9] Librosa - https://librosa.org/
[10] IPhython - https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html
[11] Matplotlib- https://matplotlib.org/stable/
[12] Seaborn - https://seaborn.pydata.org/
[13] Numpy - https://numpy.org/doc/stable/index.html
[14] Scipy - https://scipy.org/
[15] Pandas - https://pandas.pydata.org/
[16] Sklearn- https://scikit-learn.org/stable/

the noises of each sample of the train set. Such file is used for data augmentation techniques, which are explained in the next section.

The overall number of audios of the Valentini dataset corresponds to 23899 noisy audios with their corresponding 23899 clean version audios. The number of unique speakers is sixty. The number of noise types in the whole dataset corresponds to fifteen. Figure 4.1 illustrates the distribution of audios dependent on the audio length. As it can be appreciated, most of the audios last less than four seconds.

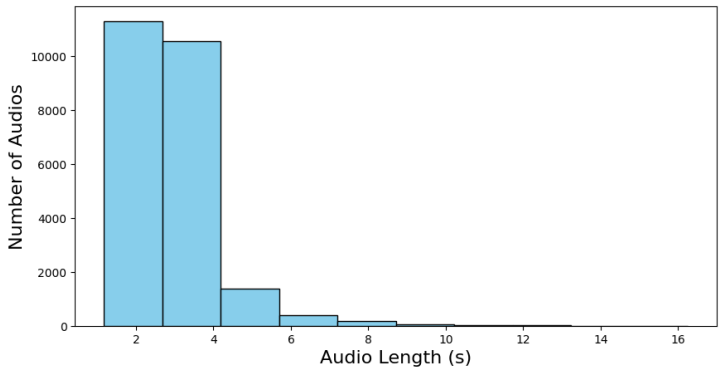

Figure 4.1: Distribution of Valentini audios based on audio length.

Figure 4.2 illustrates the distribution of audios as a function of signal to noise ratio. The audios seem to follow two uniform discrete distributions with different densities. The two distinct distributions are taken into account when performing the partition. A high number of audios present a SNR of 0.0 dB, 5.0 dB, 10.0 dB, and 15.0 dB, while a low number of audios have a SNR of 2.5 dB, 7.5 dB, 12.5 dB, and 17.5 dB. This distribution is possible because the Valentini dataset is a database with synthetic audios that mix clean audios and noises adjusting them to different SNR values. Note that the SNR is measured in dB which means that the SNR is computed in logarithmic scale. Otherwise a SNR level of 0 would not be possible. A SNR of 0 in logarithmic scale indicates the same signal and noise levels.

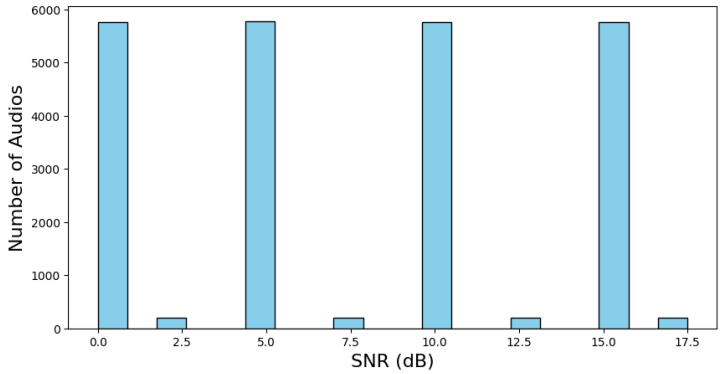

Figure 4.2: Distribution of Valentini audios based on SNR levels.

Figure 4.3 illustrates the distribution of the noises in the Valentini dataset audios, before performing the partition. Ten noises are in high proportion, while five of them are not that common. This is taken into account for the partitions creation. The noises used in this database aim to represent common daily-life noises. Almost all

of them correspond to recordings of real noises. Babble and ssn, which stands for speech-shaped noises, are not from the DEMAND dataset but synthetic noises. They are used to simulate noises derived from people speechs.

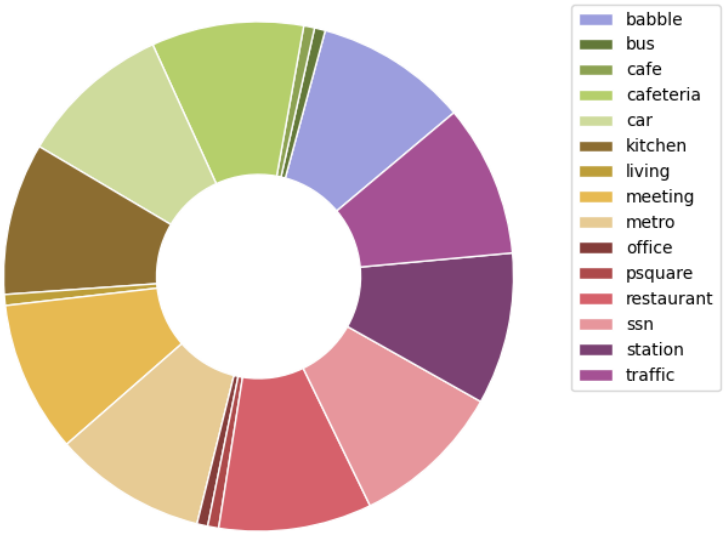

Figure 4.3: Distribution of Valentini audios based on noise type

Using the three tsv files, the partition is performed and new tsv files are created. The resulting files correspond to the train, validation and test partition files. The files contain information of the samples corresponding to each set. The location of each sample is specified in each entry of the tsv file. To obtain the test set, the partition indicated by the creators of the Valentini dataset is respected. The remaining audios are split in a 9:1 proportion into train and validation set. This partition is performed to have different speakers in the train and validation sets, but the same types of noises and SNR in a proportion of 9:1 in the two sets.

Table 4.1 contains information about each set after the partition. As it can be appreciated, the train and validation set include the noises and clean audios that are mixed after the partition. By adding the noises and the clean audios, new noisy audios are obtained. In the case of the test set, the original noisy audios are the ones employed. The clean test audios are used for comparison of the enhanced audios and the target clean audios.

	Noisy audios	Clean audios	Noises audios	Unique speakers	Types of noises
Train set	-	20592	20592	52	10
Validation set	-	2483	2483	6	10
Test set	824	824	-	2	5

Table 4.1: Description of three sets. The train and validation set are defined as a mix of clean audios and noises. For the test set, the original noisy audios are maintained. The speakers are different in the three sets. The types of noises are the same for the train and validation set, but different for the test set.

The number of unique speakers can also be appreciated. As it can be seen, the num-

ber of unique speakers of the train set is much higher than the number of speakers of the validation and test sets. Different speakers are employed in the three sets in order to detect possible overfitting and improve generalization. This approach avoids obtaining a model that learns to enhance audios of specific speakers only.

A similar strategy is employed for the noises. The noise types of the train and validation set are the same. However, in the test set, different noises are employed, so that the results obtained are more reliable. Hence, good results on the test set should indicate that the model is able to generalize and clean different types of noises from those it was trained on. From figure 4.3, it can be appreciated that of the fifteen original noises of the dataset, ten are in higher proportion than five of them. The five less common - bus, cafe, living, office and psquare - correspond to the noises used in the test set. The remaining noises are part of the train and validation set. The same is done for the SNR values. Figure 4.2 shows that some SNR are in much lower proportion than other SNR levels. The audios of SNR levels with higher density correspond to train and validation set audios. The less common SNR levels involve test set audios.

4.3 Data augmentation

Data augmentation techniques are crucial strategies to reduce the overfitting of models. Data augmentation is quite frequent in speech enhancement, as many of the databases employed for this task do not contain a huge number of audios. In this project four data augmentation techniques are implemented: white noise addition, random background noise addition, revecho and bandmasking. Note that this is only applied to the train and validation sets. In the case of the test set, the original noisy audios of the database, without any modification, are the ones employed. This is done in order to have a more reliable comparison of the results with respect to other models of the literature.

After generating the partitions, the train and validation clean signals are separated from the noises. Each clean audio is sliced into windows of size 32000. As the sampling frequency corresponds to 16000, using a window size of 32000 produces fragments of two seconds. These hyperparameters are commonly used in speech enhancement scenarios. Hence, they are also used in this project for consistency purposes. If some audio slice is smaller than the window size, padding strategies to fill the absent audio signals are used.

After that, white noises can be added probabilistically. The probability is set to $p = 0.5$, so that white noises are only added to half of the audio samples. The SNR range between the white noises and the clean audios needs to be adjusted. To cover all the SNR values of the test set, a range of [0, 20] is selected. The noise ranges are bound to sample a uniform distribution. The white noises added in this project are stationary. This means that the noise values are constant during the two seconds of each windowed audio. The majority of the experiments performed in this project do not include white noises, which means that $p = 0$. After that, some experiments that include this strategy are run. The addition of white noise in such experiments is explicitly indicated. If no mention of white noise is done, the experiments are trained without it.

Background noises are generated from the noises entries of the train and validation

sets. Random noises are selected, sliced to the same window as the clean audio, adapted to the SNR level desired and added to the clean signal. The SNR used in this case correspond to the range [5, 30]. This SNR is higher than the white noise SNR for three reasons. First, other data augmentation techniques that degrade the clean signals are employed. Hence, using slightly higher SNR at this point, may be beneficial. Second, it is a more faithful representation of common noisy environments. Third, for robustness analysis, when enhancing test audios with lower SNR. The background noise addition is done for every audio of the train and validation sets.

In the Revecho technique, reverberation is probabilistically applied to the clean audios and noises. It is important to clarify that if the revecho technique takes place, it should be applied before the background noises and clean audios are merged. This is because the reverberated process applied on the noises and on the clean signal is not the same. The noises are modified to take into account their own reverberation and most of the reverberation of the clean signals. The clean signals only incorporate a small proportion of their own reverberation.

The probability used is $q = 0.5$, which means that only half of the audios have reverberation. With the revecho technique, reverberation or echo can be introduced into the samples so that the model also learns to clean it. To control the reverberation, several hyperparameters are used. The echo is controlled by its maximum initial amplitude with respect to the original sound, the time it takes to reach a threshold value (0.001), the time that it takes for the first delay to happen and the number of times an echo is replayed. The two time parameters are not given a fixed value but a random value within a time range. This is done to increase the variability of the echos. With the same purpose, the jitter is defined as the proportion of variations that are added to make each reverberation slightly different. Finally, the keep clean hyperparameter refers to the proportion of the reverberation of the clean audio that is applied to the clean audio. The rest of the clean audio reverberation is applied to the noise.

Bandmasking involves masking some frequency bands to focus on others, which generally correspond to low frequencies. This is because, human speech and perception concentrate in low frequency ranges. The number of frequency bands in which the input is segmented and the maximum bandwidth to be removed are some of the bandmasking hyperparameters. Using these two hyperparameters, the bandwidth can be computed. Taking into account the number of bands and the computed bandwidth, the low and midlow frequencies are defined. A filter is used to eliminate midlow frequencies, focusing on lower frequencies. All of the hyperparameters used for data augmentation are shown in table 4.2. Most of the hyperparameters values chosen correspond to the standard practices in speech enhancement.

In this project, data augmentation is performed on-the-fly. This means that every time a batch is loaded, the data augmentation techniques are applied. The number of audios employed in each epoch correspond to the original number of clean audios of the train and validation sets (20592 and 2483 respectively according to table 4.1). However, in each epoch, the data augmentation techniques generate different audios from the same clean audio. Overall, the model is exposed to different transformed audios in each epoch. Using on-the-fly data augmentation techniques, poses benefits as not having to store the newly generated audios and generating a vastly diversity of audios. Figure 4.4 indicates the data augmentation process that takes place every time an audio is loaded as input for the model. For ease of understanding, clean au-

Hyperparameter	Value
Sampling frequency	16000
Window size	32000
SNR white noise	[0, 20]
Probability of white noise (p)	0.5
SNR background noise	[5, 30]
Probability of RevEcho (q)	0.5
Maximum amplitude of first echo	0.3
Time for 0.001 amplitude (s)	(0.3, 1.3)
First delay (s)	(0.01, 0.03)
Number of echo repetitions	3
Jitter	0.1
Keep clean	0.1
Number of frequency bands	120
Maximum bandwidth to be removed	0.2

Table 4.2: Data augmentation hyperparameters.

dios and background noises are indicated in turquoise and orange color respectively.

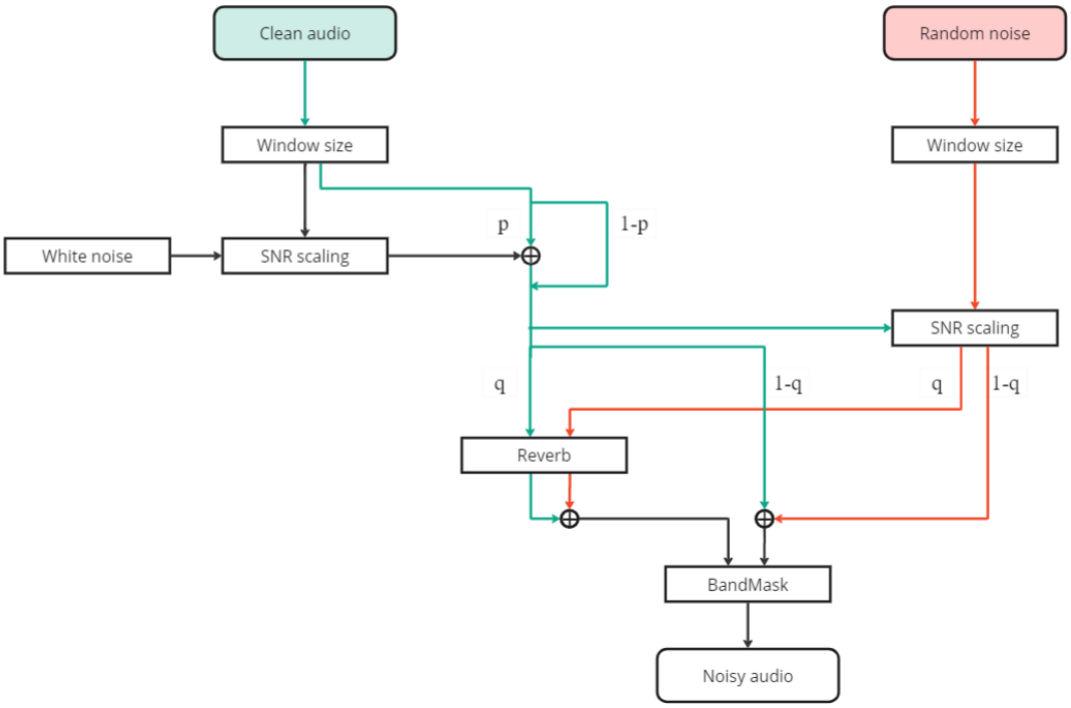

Figure 4.4: Data augmentation process. Clean audios (turquoise) and random background noises (orange) are adjusted to a window length. White noises can be added probabilistically to clean audios. The SNR value is adjusted between the clean audios and white noises. The SNR values between the clean audios, with or without the white noises, and the background noises are also adjusted before being merged. The reverb addition is done probabilistically and in parallel to the background noise addition. Finally, the frequency mask filter is applied.

Figure 4.5 illustrates the waveform and spectrogram of an audio during the data aug-

mentation process. The first step is to select a window of two seconds. Then, white noise can be added. The noisier signal can be easily appreciated by looking at the waveform which has higher amplitude across the whole window. The spectrogram also indicates that the signal has become noisier with more blurred signals. After that, background noise is added. As it can be seen, after applying the background noise, the signal gets even noisier. Note the differences in the waveform of the signal before and after applying background noises. Reverberation can be added probabilistically. The reverberation is easily detected by looking at the waveform graph. Certain peaks of sound seem to reappear and propagate in time and gradually fade away. The masking of some frequency bands can be easily appreciated in the last spectrogram.

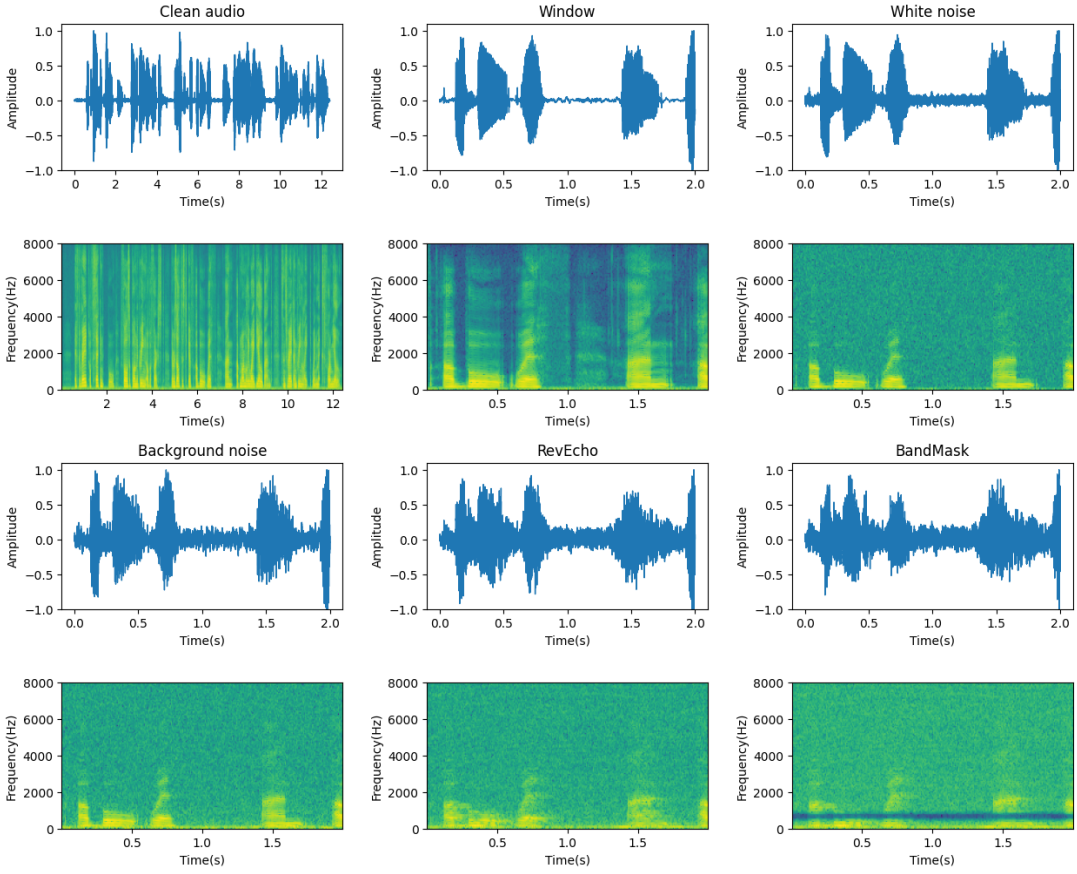

Figure 4.5: Waveform and spectrogram of one audio of the train set during the process of data augmentation. The process starts on the top left corner and continues along the top row. First, the window size is adjusted, then white noise is probabilistically added. After that, it follows the bottom row with the background noise and the probabilistically reverberation addition (RevEcho). It ends at the bottom right corner with the frequency masking process (BandMask).

In figure 4.5, the white noise addition and reverberation process are shown, although they do not always take place. On top of that, the hyperparameters used for the reverberation and bandmasking techniques are higher than those shown in table 4.2. This is done to appreciate visually the effect of each technique on the signal.

4.4 Models development

In this project three models are developed: LitGAN, MidGAN and MiniGAN. In this section the development process of the three models is explained, along with their structure and losses equations.

The models of this project are based on the best state of the art models for speech enhancement. Before generating the models architectures, the CMGAN, TPTGAN and MP-SENet models are compared and implemented using the same building blocks. This is done because the three models share most of the elements. Hence, the comparison between their scripts becomes an easier task when merged. The initial approach is to train from scratch the models, but given the hardware limitations it has not been possible.

The three models developed are, as the models on which they are based, generative adversarial networks, conformed by a generator and a discriminator. Each generator has an encoder-decoder structure. In the latent space attention mechanisms are employed. The losses employed by the models are again multiterm. Several contributions are made in this project, defining new architectures based on state of the art models. Such models are modified in order to obtain more efficient ones. The contributions of this project are explained in what follows.

First, a modification of the encoder-decoder structure is developed. In LitGAN, the encoder compromises five convolution blocks. Each block has a convolution layer, instance normalization and PReLU activation. The initial number of channels is two: one for the magnitude and the other for the phase of the signal. In each convolution layer, the number of channels is doubled. This is done for the extraction of more complex and abstract features from the input data. The encoder learns to hierarchically capture higher-level features in each convolution block. Note that this is a main difference with respect to CMGAN, TPTGAN and MP-SENet models. Those models increase the number of channels in the first convolution block of the encoder and then maintain the number of channels throughout the next convolution blocks. In LitGAN, the number of channels increases in each block, until reaching the attention mechanisms in the latent space, which uses sixty-four channels.

Another difference in the encoder is that the frequency downsampling is not performed. This is because the complexity reduction is not necessary in this case, as the network is simplified with other strategies. In LitGAN, dilations are employed from the second to the last block in the encoder. The dilation rates correspond to 1, 2, 4 and 8. The 3x3 kernel is applied over the input data with gaps between its values, controlled by the dilation rates. Figure 4.6 shows the structure of this encoder.

A third modification comes in the latent space. An important simplification of the state of the art models takes place at this point. In the latent space, only one conformer block is employed. It has a conformer for time domain and a conformer for frequency domain. Before each conformer, the data is reshaped so that the conformer focuses on the respective dependencies. The conformer block is exactly the same as the one of CMGAN and MP-SENet. It is selected over the transformer block because of the superior performance of CMGAN and MP-SENet with respect to TPTGAN. Its structure can be seen in figure 2.10.

In the decoder, the opposite operation to the encoder is performed, reducing the

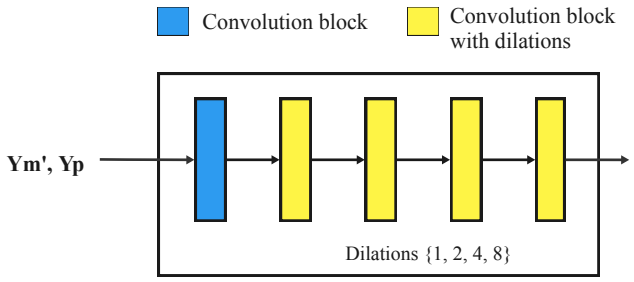

Figure 4.6: LitGAN encoder structure. Ym is the magnitude of the noisy signal. Yp is the phase of the noisy signal. The apostrophe indicates compressed feature.

number of channels from sixty four to one in each decoder. The decoders and encoder are symmetrical. Dilations are used from the first to fourth block with the same rates used in the encoder but in inverse order. The decoders are composed of five convolution blocks in which the last block does not halve the number of channels but reduce it to a fourth. This is because the encoder input is of two features and, as two decoders are used separately, the output of each decoder is just one feature.

The LitGAN model employs a magnitude decoder and a phase decoder. The magnitude decoder uses a a learnable sigmoid activation instead of PReLU in the last convolution block. This is because the MP-SENet model shows in the ablation study, that this practice improves the model notably. In the phase decoder, the last convolution is performed separately on real and imaginary parts. Then, the arcotangent function is applied to obtain the phase of the enhanced signal. Figure 4.7 and 4.8 show the magnitude mask decoder and phase decoder structures of LitGAN.

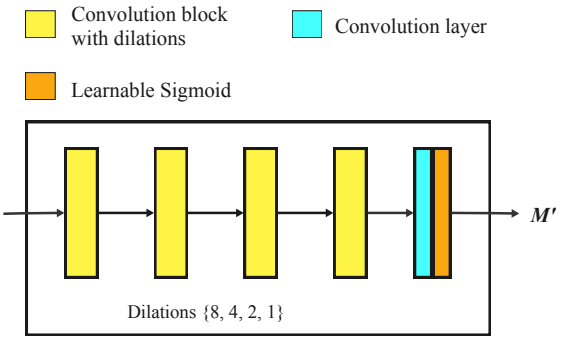

Figure 4.7: Magnitude mask decoder of LitGAN. M is the enhanced magnitude mask. The apostrophe indicates compressed feature.

The loss of the generator is defined as a three term loss:

$$L_{G_{LitGAN}} = L_{TF} + L_{Time} + L_G \tag{4.1}$$

where L_{TF} is the loss in the TF domain L_{Time} the loss in time domain and L_G the loss with respect to the discriminator output. The loss in the time-frequency domain is defined as:

$$L_{TF} = L_{Mag} + L_{Phase} \tag{4.2}$$

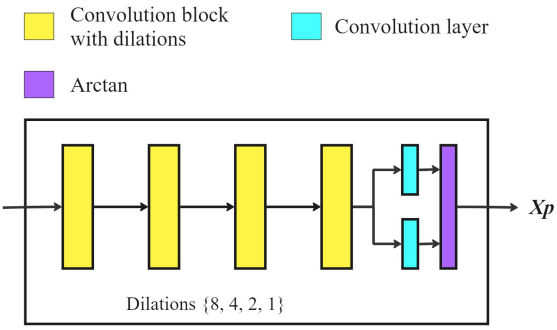

Figure 4.8: Phase decoder of LitGAN. *Xp* is the phase of the enhanced signal.

where L_{Mag} is the loss in the magnitude space and L_{Phase} is the loss in the phase space. The magnitude and phase losses can be expressed as:

$$L_{Mag} = \mathbb{E}_{X_m,\hat{X}_m}[||X_m - \hat{X}_m||^2] \tag{4.3}$$

$$L_{Phase} = L_{IP} + L_{GD} + L_{IAF} \tag{4.4}$$

where L_{IP}, L_{GD} and L_{IAF} are the instantaneous phase loss, group delay loss, and instantaneous angular frequency loss respectively. They can be expressed as:

$$L_{IP} = \mathbb{E}_{X_p,\hat{X}_p}[||f_{AW}(X_p - \hat{X}_p)||_1] \tag{4.5}$$

$$L_{GD} = \mathbb{E}_{\Delta_{DF}(X_p,\hat{X}_p))}[||f_{AW}(\Delta_{DF}(X_p - \hat{X}_p))||_1] \tag{4.6}$$

$$L_{IAF} = \mathbb{E}_{\Delta_{DT}(X_p,\hat{X}_p))}[||f_{AW}(\Delta_{DT}(X_p - \hat{X}_p))||_1] \tag{4.7}$$

where f_{AW} is the antiwrapping function, Δ_{DF} the differential operator in the frequency axis and Δ_{DT} the differential operator in the time axis. The antiwrapping function is used to avoid the error expansion caused by phase wrapping, ensuring that the angle x is within the range $[0, 2\pi]$. The differentials are used to ensure continuity of the predicted phase along frequency and time axis. The time domain loss, L_{Time}, is expressed as:

$$L_{Time} = \mathbb{E}_{x,\hat{x}}[||x - \hat{x}||_1] \tag{4.8}$$

where x refers to the waveform of the signals. Finally, the loss of the generator ability to fool the discriminator is defined as:

$$L_G = \mathbb{E}_{X_m,\hat{X}_m}[||D(X_m, \hat{X}_m) - 1||^2] \tag{4.9}$$

As it can be appreciated, the multiterm loss of the generator in LitGAN is similar to the MP-SENet loss. The main difference arises from the absence of the real and imaginary term loss shown in equation 2.19. This is because the ablation study of the MP-SENet shows little effect of this term and because using a phase loss and a real and imaginary loss can become redundant. Eliminating this term reduces the number of operations and seems to have little impact on the performance. This defines another contribution of this project.

The discriminator, which uses the magnitude of the signals as input, compromises three convolutional blocks. Then, an adaptative maxpool layer is used to reduce the

dimensionality. Finally, two linear layers are employed to classify real clean audios and fake, or enhanced by the generator, audios. The first linear layer uses PReLU activation function and drop out. The activation function of the last linear layer corresponds to a learnable sigmoid. The discriminator is almost the same as in the CMGAN, TPTGAN and MPSE-Net models, shown in figure 2.14. The main difference is that it includes one less convolutional block, to improve its efficiency.

The loss of the discriminator is simplified vastly. This is done because calculating the normalized PESQ, $\mathcal{Q}_{\text{PESQ}}$, makes the speech enhancement task exceedingly time consuming. Therefore, training the discriminator to output a value as close as possible to the $\mathcal{Q}_{\text{PESQ}}$, increases the training time notably. In LitGAN, the discriminator is trained to output a one is the audio corresponds to a clean audio and zero if the audio is an enhanced audio by the generator. This reduces the training time to approximately one third. Removing the metric discriminator is a central modification of this work. The loss of the discriminator can then be defined as:

$$L_{D_{LITGAN}} = \mathbb{E}_{X_m}[||D(X_m, X_m) - 1||^2] + \mathbb{E}_{X_m, \hat{X}_m}[||D(X_m, \hat{X}_m) - 0||^2] \tag{4.10}$$

As in the best state of the art models, all the terms of the generator loss are multiplied by specific coefficients. The discriminator loss terms are not multiplied by any coefficient. These hyperparameters are presented in the next section. Figure 4.9 shows the whole structure of the LitGAN model.

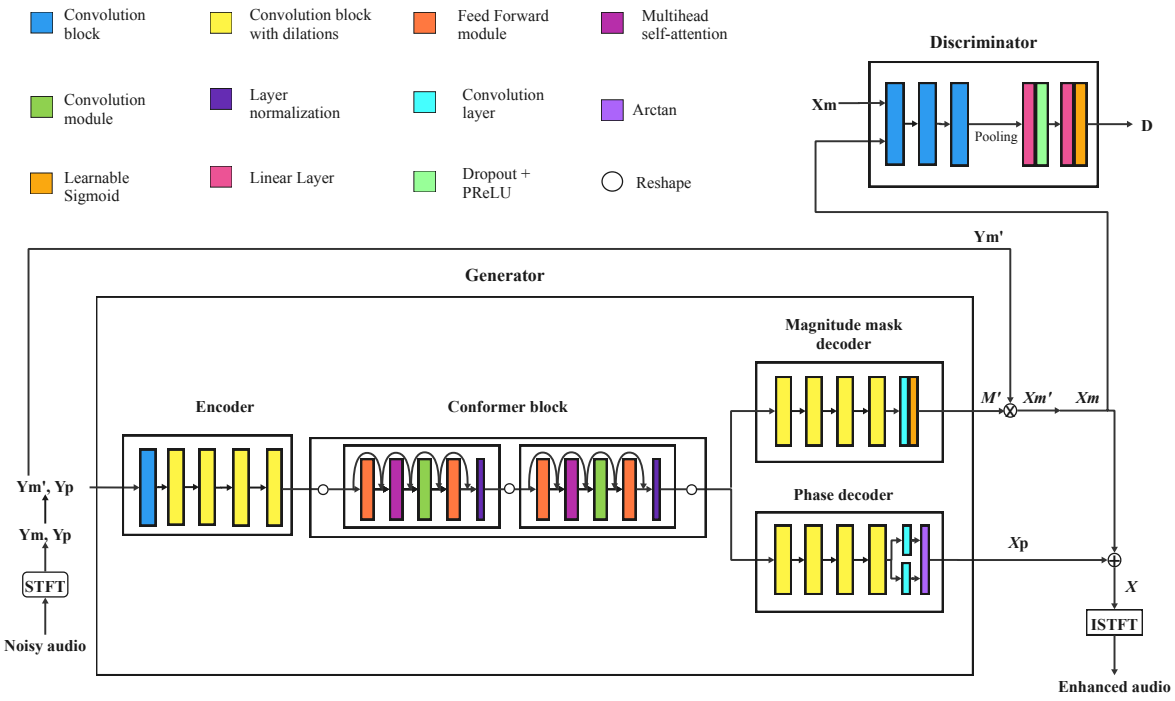

Figure 4.9: LitGAN structure. Y is the noisy signal, X the clean signal and X the enhanced signal. M is the enhanced mask of the magnitude. The subscripts m and p refer to the magnitude and phase. The apostrophe indicates compressed feature. LitGAN is conformed by a generator and a discriminator. The generator uses an encoder, a conformer block with the time and frequency conformers and two decoders.

From LitGAN, two other architectures are derived: MidGAN and MiniGAN. They are quite similar models. The difference between MidGAN and LitGAN is that the first employs two conformer blocks in the latent space instead of one. The second block is implemented after the first block, as in CMGAN and MP-SENet which employ four blocks in series. The multiterm loss for the generator and discriminator of MidGAN are the same as for LitGAN. Figure 4.10 illustrates the MidGAN structure.

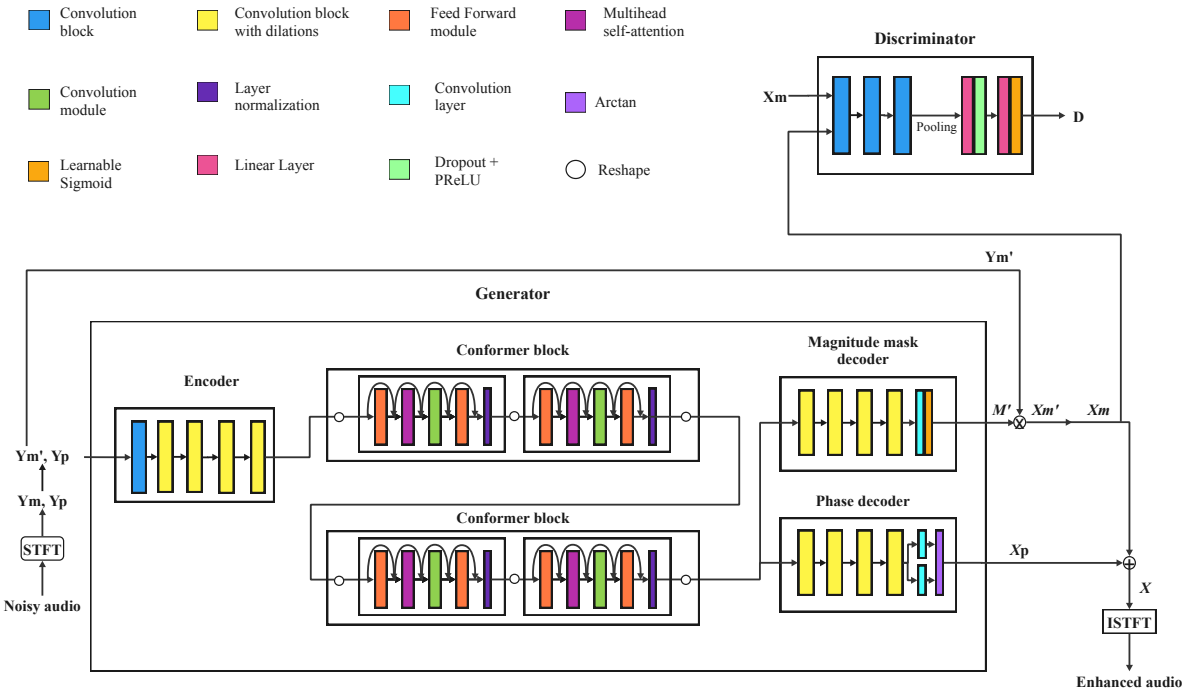

Figure 4.10: MidGAN structure. Y is the noisy signal, X the clean signal and X the enhanced signal. M is the enhanced mask of the magnitude. The subscripts m and p refer to the magnitude and phase. The apostrophe indicates compressed feature. MidGAN is conformed by a generator and a discriminator. The generator uses an encoder, two conformer blocks with the time and frequency conformers in each block and two decoders.

MiniGAN is more different. It differs in the encoder and decoder structure, being more similar to MP-SENet. Instead of using an autoencoder structure in which the convolution channels increase in the encoder and decrease in the decoder, MiniGAN employs the dilated dense net with residual connections of CMGAN, TPTGAN and MP-SENet. In this case, the number of initial channels correspond to two: one for the magnitude and other for the phase. In the first convolution block of the encoder, the number of channels is increased to thirty two channels. The dilated dense net is then employed with dilation rates of 1, 2, 4 and 8, and residual connections. The residual connections increase the number of input channels of each convolution block by thirty two. The number of output channels is always thirty two. After the dilated dense net, a last convolution block in the encoder is used to downsample the frequency dimension, in order to reduce complexity of the attention mechanisms.

In the latent space a conformer block is placed with the time and frequency conformers. The conformer block is exactly the same as the other models. The only difference

is that in MiniGAN, the number of channels used in the latent space is thirty two and the frequency is downsampled. Hence, the complexity of the attention mechanisms is remarkably reduced.

The decoders have a dilated dense net with the same structure as the encoder. Then, a convolution block is used to upsample the frequency dimension to its original one. Finally, a convolution layer is used in the magnitude mask decoder, followed by a learnable sigmoid. In the phase decoder two convolution layers are utilized to obtain the real and imaginary parts of the signal. The phase is computed as the arcotangent of both of them. The multiterm loss for the generator and discriminator of MiniGAN are the same as for LitGAN. Figure 4.11 shows the overall structure of MiniGAN. The encoder and decoders are shown in figures 2.9, 2.12 and 2.13, as they are based on the best state of the art models.

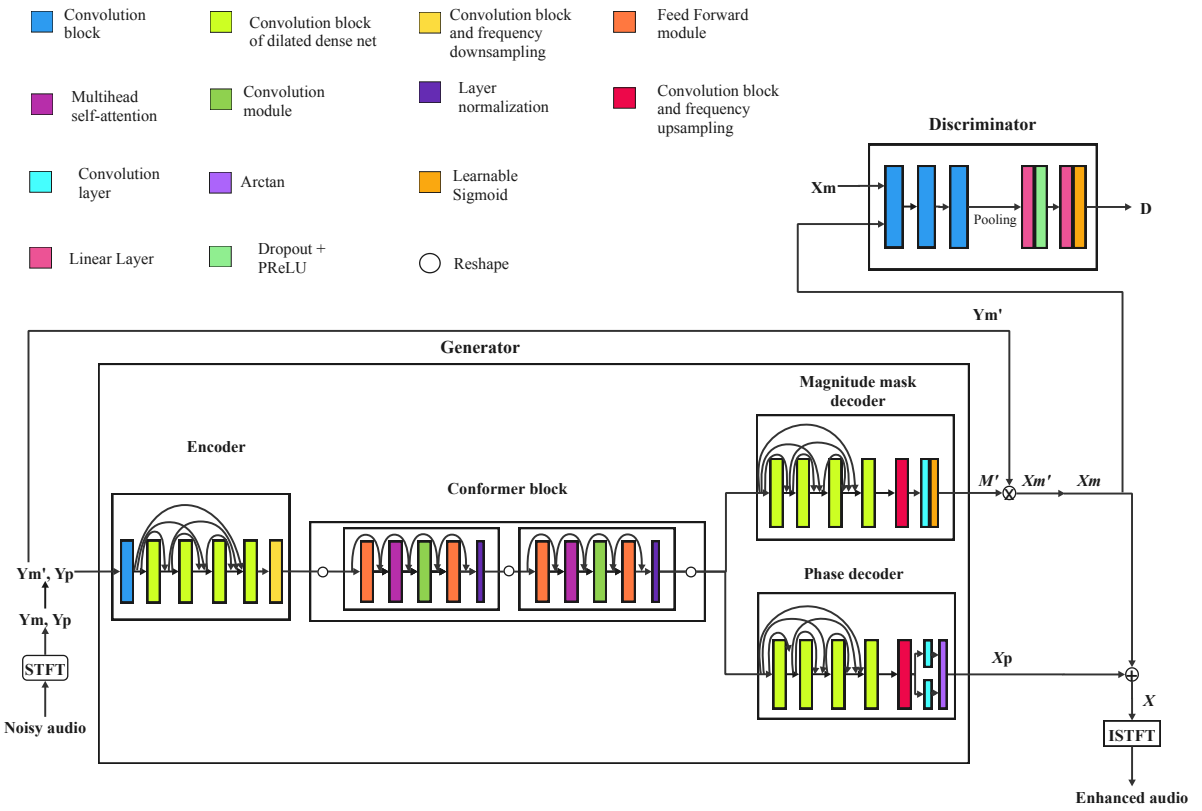

Figure 4.11: MiniGAN structure. Y is the noisy signal, X the clean signal and X the enhanced signal. M is the enhanced mask of the magnitude. The subscripts m and p refer to the magnitude and phase. The apostrophe indicates compressed feature. MiniGAN is conformed by a generator and a discriminator. The generator uses an encoder with a dilated dense net, a conformer block with the time and frequency conformers and two decoders which also employ a dilated dense net.

The three models work in TF domain. From the audio signal in time domain, the magnitude and phase of the signal are obtained trough the STFT. The magnitude is compressed. The power-law compression is performed to give the same importance to louder and quieter noises. The compressed magnitude and the phase of the noisy sig-

nal are concatenated and used as input of the encoder. After being processed by the attention mechanisms of the latent space, the magnitude and phase are separately processed by the decoders.

The first decoder obtains the magnitude mask. Such mask is used to multiply the magnitude of the noisy signal, obtaining the magnitude of the enhanced signal. A decompression of the enhanced magnitude is required to reverse the compression previously performed. The phase decoder obtains the phase of the enhanced signal as the arcotangent of the real and imaginary part of the signals. The phase and magnitude of the enhanced signal are used to obtain the enhanced signal in time domain, through the ISTFT. The magnitude of the enhanced signal is used as input for the discriminator. The discriminator also uses the magnitude of the clean signal as input. It discriminates between magnitude of clean and enhanced signals.

As it can be appreciated, the architectures developed during this project are much more smaller than current speech enhancement models. This can be reflected on the number of trainable parameters. Table 4.3 shows this information for each of the components of the three models.

Components		Parameters LitGAN	Parameters MidGAN	Parameters MiniGAN
Generator	Encoder	24.984	24.984	96.064
	Conformer block	224.896	449.792	187.072
	Magnitude decoder	24.982	24.982	96.071
	Phase decoder	24.730	24.730	95.938
	Total for generator	299.592	524.488	347.273
Discriminator		25.810	25.810	25.810
Total		325.402	550.298	373.083

Table 4.3: Number of trainable parameters of LitGAN, MidGAN and MiniGAN. The parameters are shown for each of the elements of the three models.

4.5 Models training

This project involves a huge number of hyperparameters. The data augmentation hyperparameters are not the only ones that need to be adjusted. Parameters related to the audio processing and to the training process are required. In this section, the remaining hyperparameters and the training pipeline are shown. The reasons after each decision are explained.

The three GANs work in the TF domain, which means that the audio signal must suffer a STFT before being fed to the model and an ISTFT after being output from the model. These transformations have several parameters. The Number of points for the Fast Fourier Transform (NFFT) is the number of points used in each segment, to perform the transformation. The hop parameter is the number of samples between each successive transformation. The NFFT and hop parameters employed correspond to 512 and 100 respectively, which are the standard parameters in speech enhancement. After performing the STFT, the magnitude and the phase of the signal are extracted. These are concatenated and used as input of the network.

The magnitude of the noisy signal is used to multiply the estimated mask by the mag-

nitude mask decoder. The output of this operation is the magnitude of the enhanced signal. Note that a power-law compression is performed on the magnitude. This is done to assign similar importance to loud and quiet sounds. The compression factor used in this project is 0.3, as the state of the art tendency. The hyperparameters related to the audio processing are indicated in table 4.4.

Hyperparameter	Value
NFFT	512
Hop size	100
Compression factor	0.3

Table 4.4: Audio data hyperparameters.

The main limitation encountered during this project is the training time. The number of epochs of the training process is 120. Most speech enhancement models use 100 epochs. A slightly higher number of epochs is chosen to ensure convergence. The models developed in this project are optimized so that they take the least time to train as possible. Despite that, the training time is not low. The training times for LitGAN, MidGAN and MiniGAN are approximately seven, twelve and four days, respectively. Note that all training is performed with GPU. Yet, the time required is remarkable. Parallel programming is discarded as several experiments are run at the same time in all the available GPUs.

Strategies such as automatic mixed precision have also been explored. Automatic mixed precision allows working with data types of 16 bits instead of always working with 32 bits. This reduces the computational time for some of the operations. The main problem of working with 16 bits instead of 32 bits is that the gradient vanishes more easily. To prevent that, a gradient scaler can be used. However, even when using the gradient scaler, the vanishing gradient problem occurs. Therefore, this strategy is also discarded.

Another problem in this project arises from the instability of the discriminator in the early epochs. Some times such instability leads to exploding gradients. To solve that, gradient clipping is used. Gradient clipping computes the gradients, normalize them and use a maximum value as threshold. Hence, gradients cannot grow vastly, causing problems in the training process. The gradient threshold is set to 0.5. Gradient clipping is only used in the experiments in which the training process becomes extremely unstable.

With the same purpose, initialization of the parameters is employed in the unstable experiments. Initializers set the initial values of the parameters to determined values. In this project uniform Kaiming or He initializers are selected, as they are the preferred initializers to work with the PReLU activation function used in the models. Patience is used during the training of the models of this project. Patience allows for the model to stop training after a determined number of epochs, if no improvement has occurred. As GANs are quite unstable, a patience of 75 epochs is used.

The batch size is set to 1 and gradient accumulation is employed. This is because the memory requirements are too high to allow a higher batch size. The gradient accumulation is employed to update the parameters every 16 batches. As the batch is of size 1, the gradient accumulation strategy serves as an alternative to simulate using a minibatch size of 16. Hence, the gradient accumulation enables using small

batch sizes but not having unstable learning curves as stochastic learning.

The optimizer employed in this project is the AdamW optimizer. AdamW is an improvement of Adam optimizer in which the learning rate and weight decay are optimized separately, conversely to what Adam does. Instead of applying weight decay directly to the loss, AdamW applies the weight decay directly to the weights updates. This generally leads to better generalization and improves the training process.

The initial learning rate is set to 0.001 and halves every 30 epochs for both the generator and the discriminator. These parameters are chosen because of the good results obtained in other deep learning models for speech enhancement. Other models employ smaller learning rates for the generator. However, due to slow convergence, higher rates are selected. In exceptional occasions, some of the experiments show extremely high instability. A learning rate of 0.0005 for the generator is used in such cases. The models do not have problems converging even when using a smaller learning rate as enough epochs are used to train the models.

As the loss of the generator and discriminator are multi-term, the weights of each loss term are other hyperparameters that need to be adjusted. CMGAN employs the same weights for each of the terms. MP-SENet indicates a clear improvement when adjusting the weights. They offer a set of weights which gives the best results for their architecture. However, the set of weights they propose, does not yield the best results in the models developed in this project. As parameter optimization, using a metaheuristic for example, is discarded because of the training time requirements, other strategies to select the losses weights are studied.

A parameter grid strategy to analyse the importance of each term during the early epochs is used. A study of the correlation between the PESQ and the loss terms of the generator is conducted. Linear regression is used over each loss term with the PESQ. The magnitude loss and the first term of the phase loss seem the most correlated terms to the PESQ. Hence, they obtain the highest weights. The term which refers to the output of the discriminator shows little relation with the PESQ. Hence, it obtains the lowest weight.

TPTGAN and MP-SENet include a study of the loss term weights. Although the weights they propose do not yield the best results in the models of this project, the relative importance they give to the magnitude term and the discriminator term is in agreement. In the models described in this section, the phase loss seems to also have huge relevance. The weights of the generator loss are set to 0.2, 0.2, 0.1, 0.1, 0.1 and 0.05 for the magnitude loss, the instantaneous phase loss, the group delay loss, the instantaneous angular frequency loss, the time loss and the generator loss with respect to the discriminator respectively. That is, in equation 4.11:

$$L_{Generator} = \alpha_1 L_{Mag} + \alpha_2 L_{IP} + \alpha_3 L_{GD} + \alpha_4 L_{IAF} + \alpha_5 L_{Time} + \alpha_6 L_G \tag{4.11}$$

$\alpha_1 = 0.2, \alpha_2 = 0.2, \alpha_3 = 0.1, \alpha_4 = 0.1, \alpha_5 = 0.1, \alpha_6 = 0.05$. The two terms of the discriminator loss are given the same weight.

Table 4.5 shows the hyperparameters chosen for the training process of the models. Note that LitGAN, MidGAN and MiniGAN have different architectures. In order to have a more reliable comparison between them, the hyperparameters used are the same. The hyperparameters marked with * are only used in extremely unstable experiments.

Hyperparameter	Value
Training epochs	120
Gradient clipping factor*	0.5
Patience epochs	75
Batch size	1
Gradient accumulation steps	16
Initial learning rate	0.001
Initial generator learning rate*	0.0005
Decay factor	0.5
Epochs for decay	30
Generator loss terms weights $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$	(0.2, 0.2, 0.1, 0.1, 0.1, 0.05)

Table 4.5: Training hyperparameters. The hyperparameters marked with * are only used in extremely unstable experiments. The explanation of each α can be found on equation 4.11.

4.6 Models evaluation

Apart from keeping track of the training progress of the three models, the enhanced audios are analysed and evaluated. The trained models are evaluated using the test set. The audios of the test set are not preprocessed. They do not suffer any transformation, such as data augmentation techniques. The audios are also not adjusted to a two second window. Whole audios are used to test the models. Some of the audios are too long to be processed by the generative adversarial networks. Those audios are identified, cut to the half and processed as two separated audios in the test step. This is taken into account when computing the metrics.

To compare the results obtained for the three architectures of this project, three other models are trained. The models are trained using the same hyperparameters and under the same conditions as LitGAN, MidGAN and MiniGAN, for consistency purposes. The models are called CRN, GRUSE and RESSE. It is important to train different models under the same conditions, in order to carry out a more reliable comparison of results. Although many studies do not include this process, it is considered a crucial element in this project.

CRN, GRUSE and RESSE present an encoder-decoder structure with convolution blocks in the encoder and the decoder, that increase the number of channels and reduce them respectively. The major difference between them is within the latent space. The CRN employs a LSTM in the latent space, GRUSE employs GRU and RESSE residual blocks with convolutions. As they do not use attention mechanisms in the latent space, the number of channels can be significantly higher, without creating a training time bottleneck.

The three models have one input channel in the encoder and one output channel in the decoder. The encoder increases the number of channels up to two hundred fifty six. This is the number of channels used by the elements of the latent space. The decoders then reduce the number of channels to one, starting from such high number of channels. CRN, GRUSE and RESSE are in time domain, which also reduces the training time vastly. This is because the STFT and the ISTFT are not required.

In order to perform a reliable comparison of the results, the same pipeline is used to compute the objective and subjective metrics of all the models. Objective and subjective metrics are computed separately. In both cases, the metrics for the clean and noisy audios of the test set are also computed. Hence, the enhanced audios results can be better analysed.

The objective metrics used include the PESQ, STOI, SSNR, CBAK, CSIG and COVL. They are computed using the pesq, pystoi and librosa packages of python. Sometimes objective metrics fail to reproduce human perception. Thus, obtaining subjective metrics is interesting. The main problem of subjective metrics is that they are expensive to obtain, as an study with several listeners is required.

Algorithms have been developed to obtain an approximation of those subjective measures. To compute the subjective metrics, SQUIM is employed. SQUIM uses a clean non reference match, which is given a MOS value of five. Using it as reference, SQUIM computes the predicted MOS for the enhanced audios. The non reference match used in this project is a random clean audio from the train set. This is done to ensure that the selected audio is a NMR for all the audios in the test set. Note that even though SQUIM offers a great method to predict subjective metrics, there is high dependence on the NMR audio selected. Therefore, similar results, but not exactly the same, would be obtained if other NMR audio is selected.

Apart from measuring the objective and subjective metrics, an analysis of the models efficiency is performed. In this case, the results of the three models are compared to the MP-SENet results. This is because, the three models are mainly inspired by MP-SENet and the main goal of this project is to obtain more efficient models than it. To illustrate the efficiency of the models, four variables are measured. First, the number of parameters of the models is measured. Second, the number of Floating point Operations Per Second (FLOPS) is computed. The model sizes in Mega Bytes (MB) is also obtained. Finally, the real time factor is computed. For a model to be used in real time, its RTF has to be lower than the unit. The results show which of the models developed in this project can be used in real time.

After the performance and the efficiency studies, it is clear that one of the models developed in this project is both better in performance and in efficiency. Hence, variation and ablation studies with some modifications of the best model are performed. The same pipelines used for the training and evaluation of the models are employed in these studies. The objective metrics are computed for the modifications of the best model.

Chapter 5

Results

In the following chapter, the results of this project are shown. The first section refers to the training losses of the generator and discriminator. Then, an example of enhanced signal is shown and analysed. The objective and subjective metrics obtained are also shown and discussed. The results of an efficiency analysis are illustrated. Finally, a modification of the network and an ablation study on MiniGAN is performed.

5.1 Training progress overview

All the models developed in this project are generative adversarial networks. The generators and discriminators are trained using different losses. Such losses are shown in figures 5.1, 5.2 and 5.3 for LitGAN, MidGAN and MiniGAN respectively. The figures illustrate the train and validation set losses. The losses history are shown separately for the generator and discriminator, because of the difference in scale.

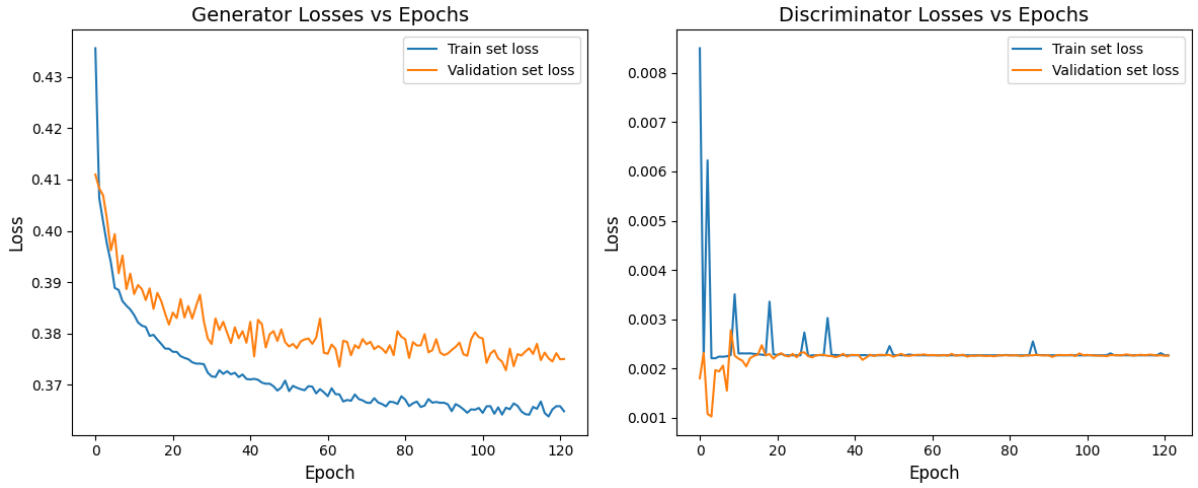

Figure 5.1: LitGAN losses history for the generator (left) and discriminator (right). The losses for the train set are shown in blue and the losses for the validation set are shown in orange.

It can be appreciated in the three models that the generator losses decrease drastically in the first twenty epochs approximately. From that moment, the losses are

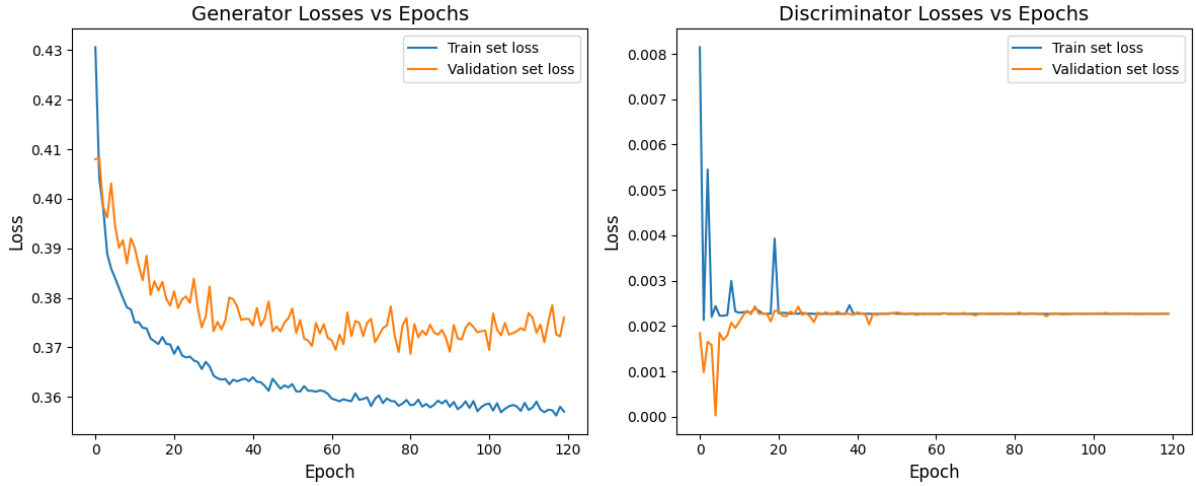

Figure 5.2: MidGAN losses history for the generator (left) and discriminator (right). The losses for the train set are shown in blue and the losses for the validation set are shown in orange.

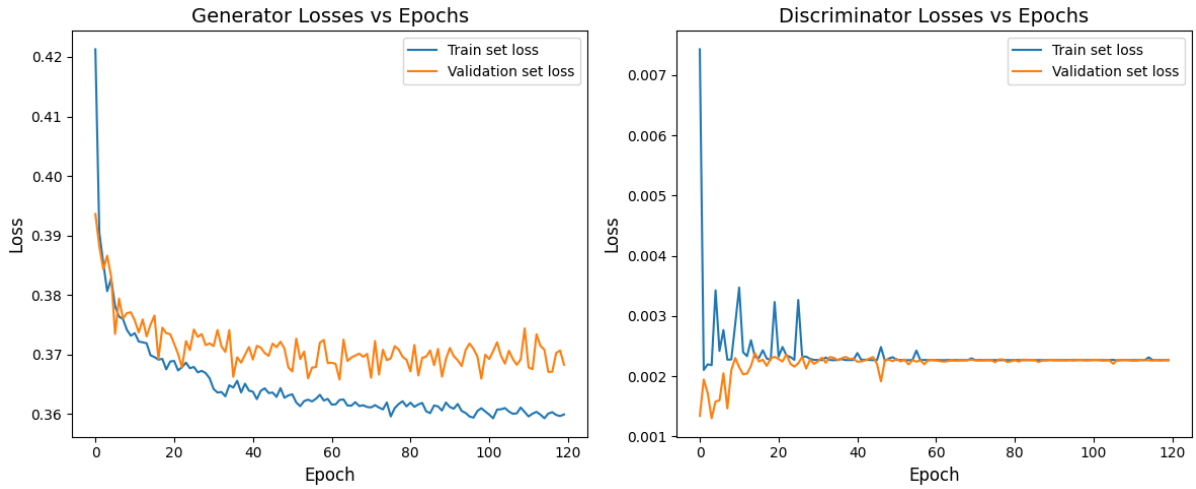

Figure 5.3: MiniGAN losses history for the generator (left) and discriminator (right). The losses for the train set are shown in blue and the losses for the validation set are shown in orange.

slightly reduced. In LitGAN, the generator loss converges around 0.37 for the train set and 0.38 for the validation set. In MidGAN, the generator loss converges around 0.36 for the train set and 0.38 for the validation set. In the last epochs of MidGAN training, the generator loss of the train set seems to slightly decrease while the generator loss for the validation set slightly increases. This could indicate a possible overfitting of the model. Note that this model is also the one that has the highest difference between the train and validation generator losses. Finally, in the case of MiniGAN, the generator loss converges around 0.36 for the train set and 0.37 for the validation set.

In the discriminator, the losses of the train and validation set for the three models converge around 0.002. Initially, the losses values are quite unstable. This can

be due to the rapid learning of the discriminator. Initially, the generator does not produce enhanced signals of good quality. Thus, the discriminator rapidly learns to distinguish between the clean and enhanced signals. In the three models it can be appreciated how the loss of the validation set is lower in the first epochs. This can be due to the overfitting of the generator. At the beginning of the training, the generator enhances significantly better signals from the train set, than from the validation set. The enhanced and clean signals of the validation set are more easily distinguishable. Its loss is therefore lower than for the train set in the early epochs of the discriminator. Overall it can be said that the generator has more difficulties in learning how to enhance the noisy signals, than the discriminator learning how to discriminate between clean and enhanced signals.

Perhaps, by not using the Q_{PESQ} for the computation of the discriminator loss, the discriminator converges very fast. Although this may seem beneficial, if the discriminator converges so fast and at such a low value, it means that the generator-enhanced samples are easily distinguishable from the clean samples. Nevertheless, the generator learns even after the discriminator converges, as the generator loss is multiterm, and only one term refers to the output of the discriminator. On top of that, the term of the generator loss that refers to the discriminator, is the one that obtains the lowest weight.

Both in the generator and discriminator, the number of epochs for training seems enough to reach convergence. A higher number of epochs may lead to overfitting, as illustrated in the MidGAN generator losses. As figures 5.1, 5.2 and 5.3 show, the generator losses curves, principally for the validation set, seem quite unstable. This is due to the low minibatch size. The batch used in this project is of one. However, through the gradient accumulation strategy a minibatch of sixteen is simulated. Using higher minibatch sizes may lead to smoother learning curves.

The learning rate is another important hyperparameter for the learning process of the three models. In the case of the generator, it seems that the learning rate allows for an initial rapid learning and convergence afterwards. For the discriminator of the three models, the learning rate may affect to the initial instability. A lower initial learning rate for the discriminator may be a better choice. All the loss curves show for both the train set and validation set consistent convergence. AdamW optimizer improves generalization by adjusting the learning rate and updating the weights separately. In this scenario, where there is little difference between the losses of the train set and validation set, it seems that AdamW optimizer is an appropriate choice. Nevertheless, there is still some difference between the generator losses of the validation and train sets.

In the training process of LitGAN, MidGAN and MiniGAN no specific initializers are used. The use of initializers may have produced a more stable learning process. However, their absence does not seem to affect significantly the results. The activation functions, excluding the last layer of the discriminator and magnitude decoder, correspond to PReLU. PReLU is a highly flexible activation function, which may have influence in the relatively smooth learning process of the generator. In the discriminator, spectral normalization is most likely one of the responsible factors for the stability of the discriminator learning process, after the first epochs. Instance normalization also influences the learning process of these models. It helps stabilizing the learning curves treating each instance separately. Instance normalization is not

affected by small batch sizes.

The bias-variance trade-off indicates that there is a balance between how well the model adjusts to the data and their generalization ability. In these models, the discriminators are poorly adjusted to the data at the beginning of their training process. However, they stabilize after some epochs and both the discriminators and generators learn to efficiently adjust to the data. This is a result of highly complex models, high number of iterations and appropriate optimization algorithms. With respect to the model ability to generalize, it can be seen that the discriminators show no difference in the losses of the train and validation sets after some epochs. However, in the case of the generator, the validation set always present a higher loss than the train set. Even though data augmentation has been applied, more data may be required to cope with the overfitting problem. Also, some regularization techniques could be added to the generator encoder and decoders to cope with this problem. The only difference in the train and validation set is the speakers. Although the difference is not vast, it should be noted that the models enhance better the audios that include the speakers they have been trained on, than audios with different speakers.

5.2 Waveform and spectrogram of an enhanced audio.

In this section, the waveform and spectrogram of an audio of the test set are analysed. Figure 5.4 shows both the waveform in time domain and the spectrogram in TF domain of an audio of the test set. Those representations are illustrated for the noisy audio, the enhanced audio by each of the three models and the clean audio.

The clean audio in figure 5.4 shows the desired output of the models for both the waveform and the spectrogram. The noisy audio, which is used as input of the model, exhibits significant noise. The differences in the waveform and spectrogram of the noisy audio and clean audio arises from the presence of noise in the noisy audio. As it can be seen the amplitude in the waveform is higher for the noisy audio at many points of time. Those points are where noise is present. In addition, the noisy audio spectrogram seems more blurry. This is because, as more noise is present in the audio, the energy spreads across different frequencies. Note that as human voice has low frequency, higher amplitudes, shown in yellow, are in low frequency regions.

The enhanced audio by LitGAN seems cleaner in its waveform but still presents some irregularities and noises. The same happens for MidGAN. The spectrogram shows how for most of the frequencies during the whole audio the amplitude is quite high. The speech components do not seem as clear as in the clean audio. Both models imply noise reduction, as can be easily seen in the time domain. However, some noises remain, as the waveforms of LitGAN and MidGAN differ significantly to the clean audio waveform.

The irregularities, which are easily identified in the LitGAN and MidGAN waveforms, are called artifacts. Artifacts refer to unwanted distortions or noises that occur due to the processing of the signals. From their presence and the unexpected high amplitude at some regions of the spectrogram, it can be said that LitGAN and MidGAN introduce distortions that foul the enhanced signals. They can indicate that the models are not complex enough to enhance audios or that they do not adapt well to the audio used as an example. Artifacts can appear when the models encounter sounds they are not familiar with. Note that test set audios are quite different to those of the train

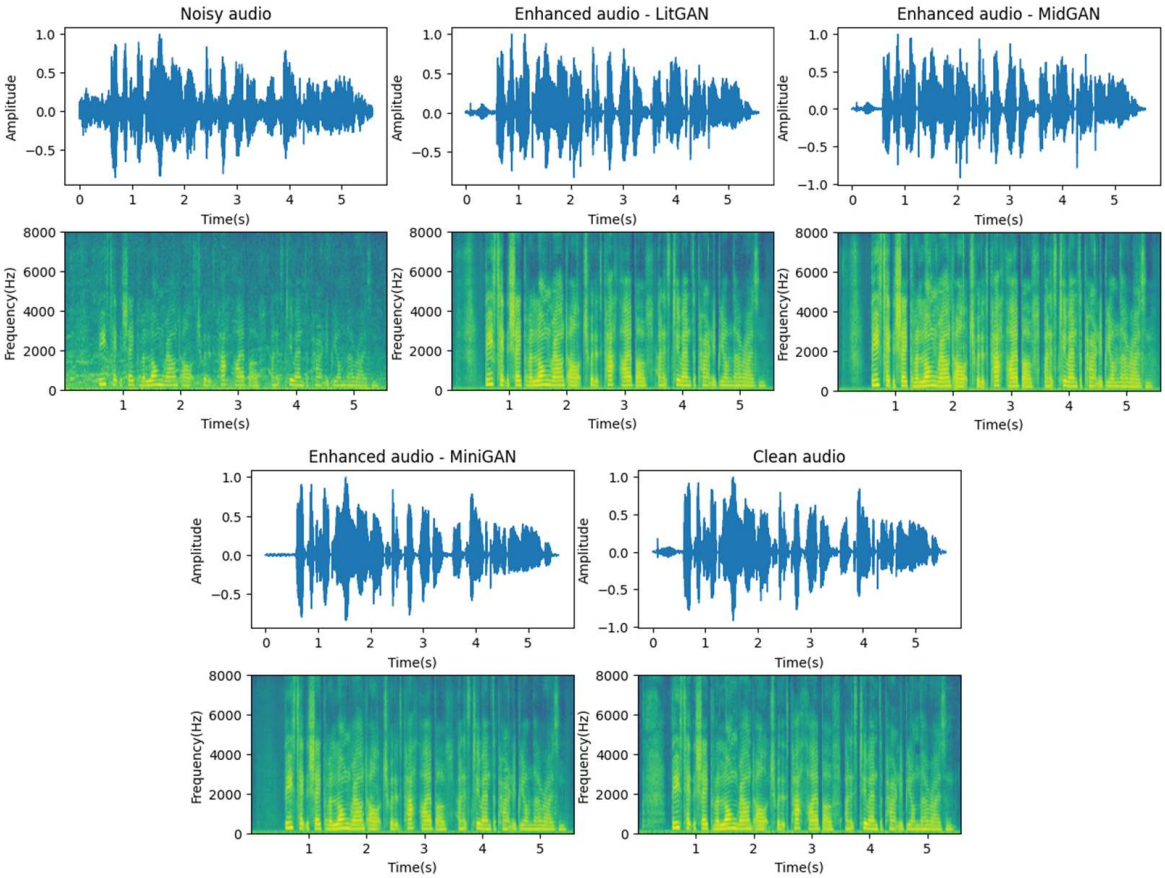

Figure 5.4: Waveform and spectrogram images of an audio of the test set in its noisy form (top left), enhanced by LitGAN (top middle), enhanced by MidGAN (top right), enhanced by MiniGAN (bottom left) and clean form (bottom right).

and validation sets. It is possible that LitGAN and MidGAN do not learn to properly generalize.

Finally, MiniGAN, seems to efficiently enhance the audio signal. As it can be appreciated the enhanced audio resembles the clean audio both in waveform and spectrogram. The noise is clearly reduced and the clean signal, which corresponds to the speaker voice, is easily distinguished. Overall MiniGAN seems the most effective model for noise reduction, offering the best enhanced signal from the models developed. However, no conclusions can be drawn from this section, as it only illustrates the enhancement of one of the test set audios. An analysis of the whole test set is required. The results of such study are shown in the following sections.

5.3 Objective metrics

The metrics used in speech enhancement can be divided in objective and subjective metrics. The former are covered in this section, while next section focuses on the latter. In both cases, the metrics are computed using the audios of the test set. The objective metrics commonly used involve the PESQ, the computed SIG, BAK and OVL, the segmental SNR and the STOI.

Table 5.1 illustrates the metrics obtained for several models when enhancing the audios of the test set. First, it shows the results for the noisy audios without being enhanced. This is done to obtain a baseline of the metrics. Then, the results obtained by other state of the art models are indicated. A horizontal line separates the state of the art models from the models trained in this project. Models trained but not designed in this project are marked with *. Such models are used for comparison purposes. The last three entries correspond to the three models developed in this work. Apart from the objective metrics, the number of parameters and the input form are shown for each model. Bold letters refer to best results in the state of the art models and in the models trained.

Model	Input	Param. (M)	PESQ↑	CSIG↑	CBAK↑	COVL↑	SSNR↑	STOI↑
Noisy	-	-	1.97	3.48	2.55	2.74	1.68	0.92
SEGAN	W	97.47	2.16	3.48	2.94	2.80	7.73	0.92
MetricGAN	M	-	2.86	3.99	3.18	3.42	-	-
DEMUCS	W	18.87	3.07	4.31	3.40	3.63	-	0.95
MANNER	W	-	3.21	4.53	3.65	3.91	-	0.95
DB-AIAT	M+C	2.81	3.31	4.61	3.75	3.96	10.79	**0.96**
DPT-FSNet	C	**0.91**	3.33	4.58	3.72	4.00	-	**0.96**
CMGAN	M+C	1.83	3.41	4.63	3.94	4.12	11.10	**0.96**
TPTGAN	M+C	1.03	3.35	4.59	3.83	4.02	**11.63**	-
MP-SENet	M+P	2.05	**3.50**	**4.73**	**3.95**	**4.22**	10.64	**0.96**
CRN*	W	1.58	2.39	4.01	2.97	3.23	4.37	0.92
GRUSE*	W	1.31	2.34	3.97	2.97	3.19	4.52	0.93
RESSE*	W	2.45	2.24	3.93	2.90	3.11	4.38	0.92
LitGAN	M+P	**0.325**	2.05	3.64	2.64	2.88	1.50	0.93
MidGAN	M+P	0.550	2.07	3.71	2.73	2.91	1.53	0.93
MiniGAN	M+P	0.373	**2.95**	**4.36**	**3.46**	**3.72**	**7.49**	**0.94**

Table 5.1: Objective metrics obtained for different models on the test set. Horizontal lines separate the metrics obtained for the noisy audios of the test set, the metrics obtained by other state of the art models, and the metrics obtained for the models used for comparison (*) and developed in this project. Bold numbers indicate the best results of each category. The input form and number of parameters, in millions, are also shown. W stands for waveform, M for magnitude, C for complex and P for phase.

The input column indicates the form of the input. It can be the waveform (W) in time domain or magnitude (M), complex (C), magnitude and complex (M+C) and magnitude and phase (M+P) in the TF domain. The three models used for comparison are in time domain, while the three models developed in this project are in TF domain. The current trend is to use TF domain combining the magnitude with the phase or complex characteristics of the audio signals. The numbers of parameters of the six trained models are remarkably low. Note that LitGAN, MidGAN and MiniGAN employ numbers of parameters much lower than those used by popular state of the art models. The lower the number of parameters, the better as the goal of this project is to obtain an efficient model. For the objective metrics the higher the value, the better, as they define the performance of the models.

LitGAN and MidGAN show little improvement with respect to to the noisy audios of the test set. LitGAN and MidGAN do not generalize well. Thus, when the models encounter the noisy audios of the test set, which are quite different, both in noise type and speakers, to those used to train and validate the model, the results are far

from desirable. This can be due to the small number of channels that they use in the first layers of the autoencoder. Using such small number may lead to information loss which translates into poor enhancement. It is clear, given that MidGAN uses the double number of conformers than LitGAN but obtains similar results, that the limitation in their results must not be present on the latent space.

CRN, GRUSE and RESSE obtain better results but have a higher number of parameters. MiniGAN obtains the best results for all the objective metrics. This indicates that the audios enhanced by MiniGAN are those of better quality and intelligibility. MiniGAN has a slightly higher number of parameters than LitGAN. MiniGAN is still a smaller model than those used in the state of the art by far. The difference in the objective metrics of MiniGAN and LitGAN, suggest that increasing the number of channels in the first convolution block and using residual connections is a better strategy than using an autoencoder structure per-se. This conclusion is reached as LitGAN and MiniGAN only differ in their encoder and decoders structures.

The metrics obtained by other state of the art models are significantly better than those obtained in this project. It would be desirable to obtain higher metrics. However, the main goal of this project is to develop a speech enhancement model more efficient than those of the state of the art. An improvement in the efficiency is accompanied by a worsening of the performance as expected. Nevertheless, considering the small size of MiniGAN, outstanding performance results are still obtained for it. Table 5.1 indicates the objective metrics means for each of the models. That information is valuable but limited as the variance in the computations is not known. Figure 5.5 shows the objective metrics means and standard deviations in a bar-plot form.

In the bar-plot, the higher the height of the bars the better score. Smaller error bars are desired. Note that the different metrics are represented in different scales so that the graphs analysis becomes an easier task. The metrics for the noisy audios and enhanced audios of the six models are shown. The metrics for the clean audios are not shown, as they simply take the maximum values of each metric. Such values correspond to 4.5 for the PESQ, 5 for the CSIG, CBAK and COVL, 35 for the SSNR and 1 for the STOI. The standard deviations are almost none existent in such scenario. In addition to taking expected and known values, in some of the metrics, such as the SSNR, plotting the metrics of clean audios only hinders the graphs interpretation. This is because of the difference in scale.

Figure 5.5 shows that the variance of the metrics in each of the six models is quite high in comparison to the metric scale. This indicates that some of the enhanced audios of the test set obtain much better scores than others. The error bars in all the models are of similar sizes, indicating that all models suffer from this. Note that LitGAN and MidGAN have smaller variation errors in the CBAK and the SSNR. This is because they introduce distortions that increase the noise in the enhanced audios, obtaining for most of the audios low values of CBAK and SSNR. It is also remarkable that the SSNR standard deviations are quite high for the rest of the enhancement models and the noisy audios. This, and the fact that the SSNR of the clean audios correspond to 35, indicates that speech enhancement models are yet not able to completely subtract in an effective way the noises from noisy signals. Note that even the best state of the art models do not have a SSNR above 12, as table 5.1 indicates.

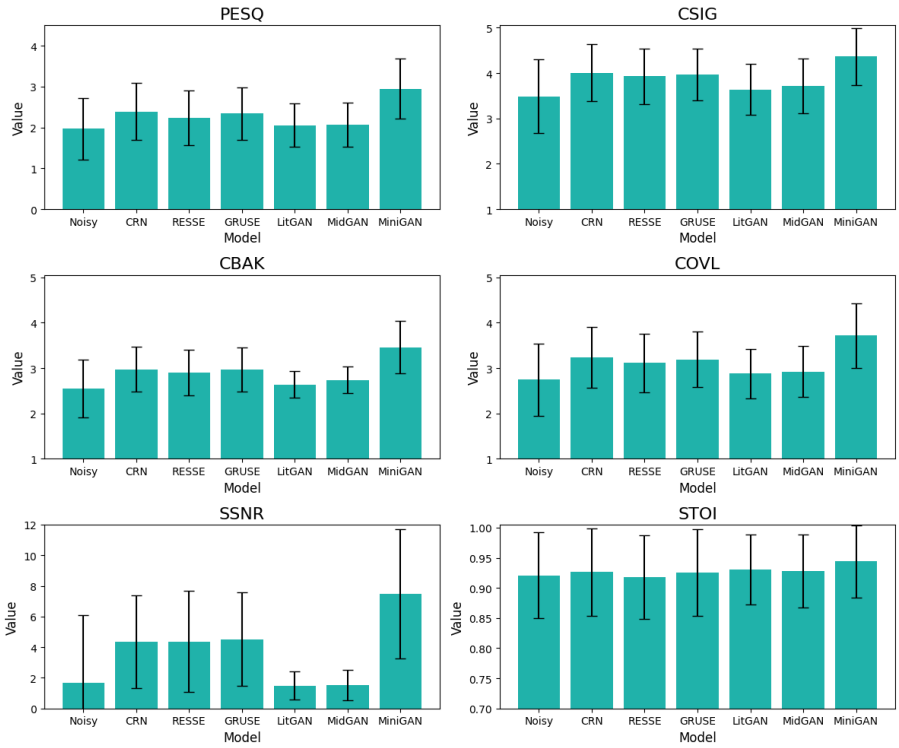

Figure 5.5: Objective metrics for the noisy audios and the six models trained on this project. The bar-plot height indicates the mean of the objective metrics obtained on the test set. The error bars refer to the standard deviation for each of the metrics in each model.

5.4 Subjective metrics

Figure 5.6 shows the computed Mean Opinion Score (MOS) obtained for the noisy and clean audios of the test sets, the models developed for this project, and the other models trained for performance comparison purposes. MOS values range from 1 to 5. Higher values indicate better opinion about the audio. It is important to emphasise that in this work no subjective study with people as listeners is included. Instead, an algorithm is used to obtain the computed MOS.

Violin plots, as in figure 5.6, give different information. The coloured shape indicates the density of the audios that have a given computed MOS (y axis). The black box inside each blue shape indicates the values between the 25th and 75th percentile (or Q1 and Q3). The interquartile range (IQR), which is defined as the difference between Q3 and Q1, should be as small as possible. The horizontal white line inside the coloured box indicates the median. The higher the median, the better. The thinner vertical lines out of the box indicate all the range of values of computed MOS that the samples take. The maximum and minimum values should be as high as possible. Note that the computed MOS take values, at most, between two and five. The blue shapes bellow two or above five do not indicate that there are audios that take these scores. It is merely a question of the design of the graphs that are intended to indicate the trend of the data.

Although the 25th, 50th (median) and 75th percentiles are different for each model,

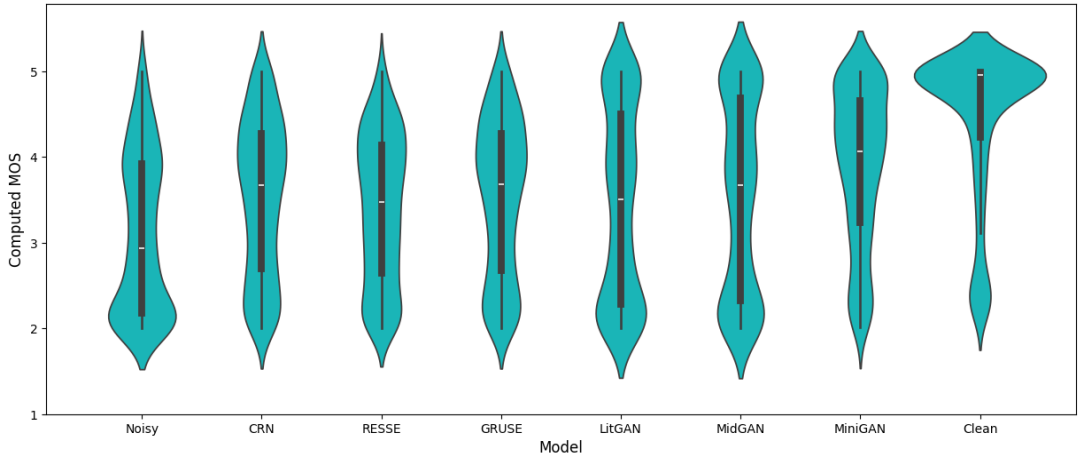

Figure 5.6: Computed Mean Opinion Score values for the noisy audios, enhanced audios by several models and the clean audios of the Valentini test set.

the minimum and maximum values, approximately two and five respectively, seem to be similar for the noisy and enhanced audios. This arises from two reasons. First, in each of the cases, some audios seem to be difficult to clean, while others are easily enhanced. These audios are not the same for the different models, as each model learning process is different. For some of the models, some audios are easy to clean, while for others it becomes a difficult task. Note that this also affects the objective metric scores and can be one of the reasons why figure 5.5 shows high variance in the metrics. Second, SQUIM computes subjective metrics. It makes sense that even some of the noisy audios get high MOS values and some of the clean audios get low MOS values. This is because SQUIM attempts to represent the values that would be obtained from a subjective study of audio quality and intelligibility. The listeners used in such studies are not professionals, which sometimes implies that the values assigned by them are not consistent.

In the case of the noisy audios, the median is around 3 and the 25th and 75th percentile correspond to approximately 2.2 and 3.8 respectively. The violin plot shows how the region with more density of audios is around 2. The CRN median is above 3.5 which indicates an improvement. Also, its 25th and 75th percentile are around 2.7 and 4.2. It can be seen how its most common metric score is around 4. RESSE and GRUSSE show similar values. RESSE has a slightly lower median.

The audios enhanced by LitGAN and MidGAN have wider IQR than the enhanced audios by the rest of the models. The medians of the two models are quite similar to those of CRN, RESSE and GRUSE. Even tough the 75th percentile is higher for the audios enhanced by these two GANs than for the audios enhanced by CRN, RESSE and GRUSE, the 25th percentile is lower. From the violin shape, it can be seen that some audios get MOS scores of almost 5, while some of them are around 2. From this it can be derived that LitGAN and MidGAN do not generalize well.

The audios enhanced by MiniGAN present a median above 4, which is the higher value for all the models studied in this project. The IQR is more similar to those of CRN, RESSE and GRUSSE than to LitGAN and MidGAN. Most of the enhanced audios obtain computed MOS metrics above 3.5. MiniGAN obtains better subjective results and generalize better than the other two models developed in this project. Note that

the shape of the MiniGAN model widens at the top and narrows at the bottom. This is desirable, as the aim is to make its shape as similar as possible to that of the clean audios.

The clean audios have a median of almost 5 which makes sense, considering that the audios are clean and of good quality. Most of the clean audios have a computed MOS score of almost 5. Clean audios have the smallest IQR indicating their high and consistent quality. Several outliers are in the region between MOS values of 2 and 3. This is known because the violin shape widens in the region around 2.5, while the minimum value, indicated by a thin vertical line, is above such region.

5.5 Efficiency study

Apart from measuring the objective and subjective metrics, an analysis of the models efficiency is performed. The results for the three GANs developed and MP-SENet are provided in table 5.2. MP-SENet is used for comparison in this case, as the goal of this project is to develop a model based on such model but much more efficient. To illustrate the efficiency of the models, four variables are measured: the number of parameters in millions, the number of floating point operations per second (FLOPS) in millions, the model size in Mega Bytes (MB) and the real time factor (RTF). The RTF has no units as it is defined as a ratio between the time taken to process one audio and the audio duration. Again, bold letters indicate the best results of each variable. The RTF is computed on CPU.

Model	Parameters (M) ↓	FLOPS (M) ↓	Size (MB) ↓	RTF ↓
MP-SENet	2.05	47489.35	8.227	1.1529
LitGAN	**0.325**	10958.02	**1.246**	0.5572
MidGAN	0.550	18832.60	2.106	1.0822
MiniGAN	0.373	**9022.93**	1.427	**0.2404**

Table 5.2: Efficiency comparison of MP-SENet and the models developed in this project. Four parameters are shown: the number of parameters, the number of loating point operations (FLOPS), the size of the models and the real time factor (RTF).

The number of parameters of the model ought to be as low as possible. In that matter, LitGAN shows the best values. MiniGAN closely follows LitGAN and MidGAN also has a quite small number of parameters compared to current state of the art models. LitGAN and MidGAN have a small number of parameters in their encoder and decoders, as table 4.3 shows. MiniGAN has a higher number of parameters in the encoder and decoders as the number of channels employed in them is higher. However, the number of parameters used in the latent space by the conformers of MiniGAN is lower. This is because the number of channels in the conformers of MiniGAN is the half of those of LitGAN and MidGAN. MP-SENet has more than six times the parameters of LitGAN, but is still quite small in number of parameters for being one of the best state of the art models.

The three models of this project significantly reduce the number of FLOPS with respect to MP-SENet. Even though LitGAN is smaller than MiniGAN in number of parameters, it implies more floating point operations per second. This may arise from the downsampling of the frequency and the use of a lower number of channels in the

latent space. Conformers are complex architectures that involve feed forward layers, multi-head self-attention and convolution blocks. It is reasonable that if LitGAN and MidGAN have conformers with higher number of channels, the number of operations increases with respect to MiniGAN.

The size of the models, measured in MB, does not only take into account the number of parameters and their size but also the buffer elements. Buffers are tensors that are not considered parameters but that are linked to the model. They are similar to parameters but are not learned though backpropagation. They can have fixed values or statistically computed values. They are useful to obtain information about the state of the model. As the size of the model is related to the number of parameters, it is reasonable that LitGAN obtains the best values. MiniGAN follows closely. MidGAN has a slightly higher model size, but is still quite small. As with the number of parameters, MP-SENet is bigger, but still quite efficient for its performance.

Finally, the real time factor is measured. For a model to be used in real time applications, its RTF must be equal or lower than 1, as that means that the time required to process an audio is equal or lower than the audio length. LitGAN and MidGAN have a lower RTF than MP-SENet. That does not come as a surprise, as MP-SENet is a quite complex model with a total of eighth conformers. On top of that, LitGAN can be used in real time.

Although LitGAN and MidGAN obtain better results in that matter, the RTF does not improve much with respect to MP-SENet. A reason for this could be that LitGAN and MidGAN do not perform frequency downsampling. However, the number of conformers is much lower with respect to MP-SENet. Further explanation can be found in the number of parameters of the encoder and decoder. Higher number of parameters, as in MP-SENet and MiniGAN, can lead to more efficient feature extraction. Therefore, the workload in subsequent layers is reduced.

LitGAN and MidGAN perform a poor feature extraction in early layers of the encoder as the number of channels is drastically low. This may explain why, although the number of parameters of the models is remarkably low, the models are not as efficient as MiniGAN and do not obtain good scores in the objective or subjective metrics analysis. Artifacts, as the ones shown in figure 5.4, can also be explained by this inefficient feature extraction. This poor behaviour does not only degrade their efficiency but also their enhancement and generalization abilities.

The best RTF is obtained for MiniGAN. This means that MiniGAN is the model that enhances the audios the fastest. MiniGAN can be used in real time. Figure 5.7 shows the PESQ values of MP-SENet and the three models of this project against the RTF. The higher the PESQ and the lower the RTF, the better.

It can be seen how MP-SENet obtains the best performance but has the highest RTF. MiniGAN obtains the second to best values in relation to the PESQ and has the lowest RTF. It is also noticeable that LitGAN and MidGAN obtain almost the same PESQ but very different values for the RTF, making LitGAN a model that can be used in real time convesely to MidGAN. Two conclusions can be drawn from this.

First, the difference in the architecture, which is the number of conformers, significantly affects the efficiency. The RTF of MidGAN is about twice that of LitGAN. MidGAN has twice as many conformers as LitGAN. It follows that the low efficiency of MP-SENet is indeed conditioned by its large number of conformers. However, con-

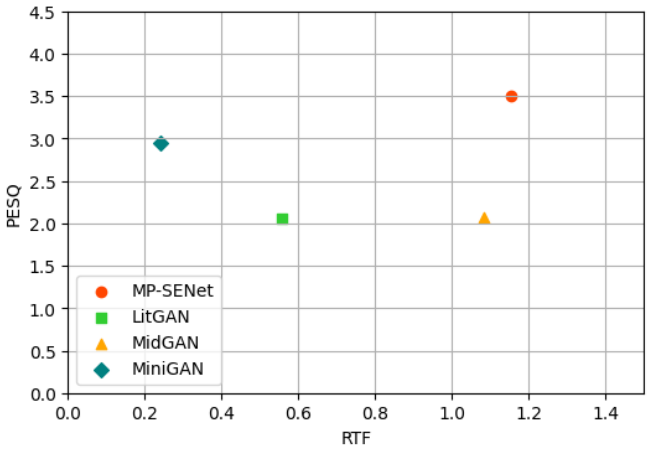

Figure 5.7: PESQ against real time factor (RTF) for the three models developed in this project and MP-SENet.

formers are not the only responsible for the models efficiency. MP-SENet has twice as many conformers as MidGAN and its RTF is very similar to that of MidGAN. This indicates that the frequency downsampling and the use of the dilated dense net, by MP-SENet and MiniGAN, improves the efficiency of the models remarkably.

Second, the fact of LitGAN and MidGAN obtaining almost the same values in PESQ confirms that using more conformers does not lead to a direct improvement in the quality of the enhanced signal. The main limitation in their architectures is the encoder and decoders structures, which does not lead to to proper feature extraction. Therefore it limits their enhancement abilities. This is in agreement with the conclusions drawn from results shown in previous sections.

5.6 Network variations and ablation study

According to the results shown in tables 5.1, 5.2 and in figures 5.5, 5.6, 5.7, MiniGAN is the best model, from those developed in this project, both in objective metrics, subjective metrics and efficiency. This model is selected for a further analysis. A network variation study and an ablation study are performed to analyse which of the elements of such small model are responsible for its extraordinary results.

First, a variation of MiniGAN with the order of the conformers inverted, MiniGAN-FT, is developed. The frequency conformer is placed before the time conformer. Second, a variation of MiniGAN with forty eight channels instead of thirty two, MiniGAN-48, is defined. Note that the number of channels affects both the encoder, the decoders and the conformers. After that, two modifications to see the effect of the data augmentation process are obtained. A modification of MiniGAN without data augmentation, MiniGAN-NDA, is studied. Finally, a modification of MiniGAN trained with white noise, on top of the other data augmentation techniques, MiniGAN-DA+WN, is developed.

Table 5.3 shows the objective metrics for MiniGAN and its two variations in the network structure: MinGAN-FT and MiniGAN-48. Table 5.4 shows the objective metrics for MiniGAN and its two variations in relation to the data augmentation: MiniGAN-

NDA and MiniGAN-DA+WN. For the white noise addition, initializers, clipping gradients and lower learning rates for the generator are employed. This is because the instability of the learning process increases when processing highly noisy audios. The convergence of the losses is ensured in order to obtain a more reliable comparison of results. Values marked in bold indicate the best values for each metric. The scores obtained are practically the same in all the cases. The reasons for this are explained in what follows.

Model	PESQ↑	CSIG↑	CBAK↑	COVL↑	SSNR↑	STOI↑
MiniGAN	**2.95**	**4.36**	3.46	**3.72**	7.49	0.94
MiniGAN-FT	2.93	**4.36**	**3.52**	3.71	**8.43**	0.94
MiniGAN-48	**2.95**	4.34	3.46	3.71	7.62	**0.95**

Table 5.3: Objective metrics for MiniGAN and its two network variations. MiniGAN-FT has the frequency conformer before the time conformer. MiniGAN-48 has forty eight channels through all the network instead of thirty two.

The MiniGAN in which the order of the conformers is altered, MiniGAN-FT, shows an improvement in the CBAK and in the SSNR. Given that the CBAK focuses on the background noise and the SSNR measures the segmental signal to noise ratio, it makes sense that an improvement in one of them is related to an improvement in the other. This improvement may happen because the frequency conformer learns the most important features of the signal to be enhanced better than the time conformer. However, the difference in the metrics are not noteworthy enough to reach a conclusion.

The MiniGAN with forty eight channels, MiniGAN-48, only shows an improvement in the STOI metric. The scores for the objective metrics are in general very similar to those of MiniGAN. This argues that the results obtained for MiniGAN are not because of its size but the complexity of the overall architecture. Hence, an efficient model as the one developed in this project is possible.

Model	PESQ↑	CSIG↑	CBAK↑	COVL↑	SSNR↑	STOI↑
MiniGAN-NDA	**2.96**	4.31	3.43	3.70	7.06	0.94
MiniGAN-DA	2.95	**4.36**	3.46	**3.72**	7.49	0.94
MiniGAN-DA+WN	2.82	4.27	**3.47**	3.59	**8.48**	0.94

Table 5.4: Objective metrics for MiniGAN and its two variations of the ablation study. MiniGAN without data augmentation. MiniGAN-NDA, only adds background noises to clean audios. The original MiniGAN, MiniGAN-DA, also uses reverberation and bandmasking. MiniGAN with data augmentation including white noise, MiniGAN-DA+WN, also adds white noises.

The MiniGAN trained without data augmentation, MiniGAN-NDA, shows a slightly improvement in the PESQ metric. The results obtained by the MiniGAN without data augmentation, indicate that the data augmentation techniques applied in this project do not cause a noteworthy improvement in the test results. They avoid overfitting, but do not cause an improvement in the metrics. This raises the question of whether more data augmentation strategies are required. To reach conclusions about the model and the audios of the test set, a last experiment with the addition of white noise is performed.

The MiniGAN trained with data augmentation including white noise, MiniGAN-DA+WN,

shows the best results in the CBAK and SSNR. As the model is trained using noisier signals, it comes to no surprise that it extracts the noise more efficiently, obtaining enhanced signals with higher CBAK and SSNR. However, the PESQ results are the worst of all the experiments. The STOI values obtained for the three models of table 5.4 are the same.

A further analysis on the effect of data augmentation is performed. Figure 5.8 shows the PESQ obtained for the validation set during the training process of MiniGAN without data augmentation (orange), with data augmentation (blue) and with data augmentation including white noise (green). The PESQ values obtained for the validation set, increase in the first sixty epochs and then seem to stabilize. The dashed lines indicate the PESQ obtained for the test set in the three cases.

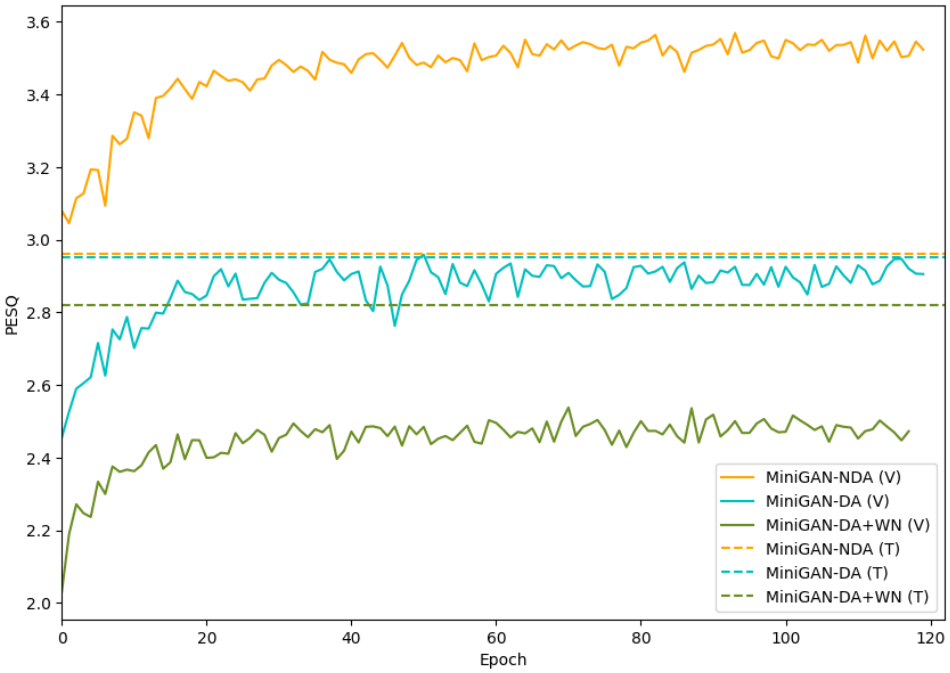

Figure 5.8: PESQ values for the validation sets (continuous lines) and test sets (dashed lines) of three different MiniGAN models. The orange model does not include data augmentation (MiniGAN-NDA). The blue values correspond to the trained model with data augmentation or original MiniGAN (MiniGAN-DA). The green model includes white noise in addition to the data augmentation techniques (MiniGAN-DA+WN).

The model trained without data augmentation, MiniGAN-NDA, obtains much higher values in the validation set that in the test set. This indicates that overfitting takes place if no data augmentation process is performed. The model trained with data augmentation, MiniGAN-DA, which correspond to the original MiniGAN, obtains similar values of PESQ for the validation and test set. The model trained with data augmentation including white noise MiniGAN-DA+WN, obtains worse PESQ values in the validation set than in the test set. Remember that the data augmentation techniques are only applied in the train and validation sets. Hence, it makes sense that validation sets get lower values in the PESQ than test sets, in those cases in which data augmentation has been applied (green and blue). However, the difference in the test set results (dashed lines) indicate that using white noises in the training and validation process may lead to local minima during the learning process. Hence, the

PESQ obtained in the test set for this case is significantly lower than for the other two cases.

Using background noises, reverberation and bandmasking is the most consistent scenario to obtain similar results between the validation and test set. This is the practice used for the original MiniGAN. In such case, the PESQ of the validation set is slightly lower than the PESQ of the test set. From the information shown in figure 5.8, it can be deduced that data augmentation techniques do not improve results of these models, but they clearly avoid overfitting.

Another conclusion can be drawn looking at the results obtained in this section. The train and validation set are not different enough to analyse the model robustness and ability to generalize. Note that the train and validation set compromise different audios with different speakers but the same type of noises. The test set includes audios with different speakers and different types of noises. One of the main problems of speech enhancement models arises from the variety of noises. Most models are trained with a limited number of noise types. Hence, they have difficulties enhancing noisy audios with other types of noises. A possible solution for this problem would be to split the train and validation set so that they compromise different types of noises as well of different speakers.

The results of this section indicate that there is not much difference between the variations of MiniGAN. This can be due to (i) the variations tested not being significantly different to the original model and (ii) some of the audios of the test set causing a stagnation in results. Both reasons are responsible for the results. For MiniGAN and all its variations, specific audios received the best and worst scores in all the cases. This arises because the models trained here are not different enough to show a huge difference in the metrics.

In the ablation studies of MP-SENet and CMGAN, modifications that change drastically the model architecture are analysed. For example, they eliminate one of the decoders, the attention mechanisms or the discriminator. They also train models without one or more loss terms, in order to see the difference in the learning process. Seeing the virtually non existent differences in the study performed here, other severe modifications to the MiniGAN model could shed light on the most important elements of the model.

Chapter 6

Conclusions and future work

In this work, several efficient speech enhancement models are developed. In the following chapter, the main conclusions drawn from this project are set out. Future work lines are also indicated.

6.1 Conclusions

The main goal of this work is to develop an efficient and high performance speech enhancement model. For that, a state of the art review on the current methods and tendencies of speech enhancement is performed. Three models are designed using deep learning elements, including convolutions, attention mechanisms, autoencoders and generative adversarial networks. As speech datasets are not big, data augmentation techniques are implemented to simulate more data.

Several difficulties have been present along the whole project. To begin with, because of hardware limitations, only a batch size of one can be used. The generative adversarial networks of this project take long to train, which also limits the number of experiments and the parameter optimization process. The training process of some of the experiments becomes very unstable in the early epochs due to exploding gradients. Many strategies are studied and developed to deal with these setbacks.

The three models built for this project are very efficient. However, two of them, LitGAN and MidGAN, do not obtain suitable results and are not able to generalize. The third model, MiniGAN, obtains outstanding results, both in performance and in efficiency. It is clear that elements that distinguish MiniGAN, such as the dilated dense net with residual connections or the frequency downsampling, have a huge impact on its performance and efficiency.

MiniGAN effectively generalizes and is a robust model that can enhance noisy audios with different speakers and noises from those it is trained on. MiniGAN is smaller than state of the art models and can be used in real time. The balance between performance and efficiency proves that MiniGAN is a model with extraordinary properties. The purposes of this work have been achieved thanks to this model.

The framework of this project is based on some of the best state of the art models. Approaches on how to make such high performance models more efficient are pursued. The opposite process can be followed from MiniGAN. Multiple performance-

enhancing improvements can be made to obtain an extremely efficient model with even higher performance metrics. This work does not only contribute to the current state of the art, but also opens the way to further research. MiniGAN has practical implications. Due to its small size and its ability to work in real time, it can be used in a broad variety of daily life applications.

6.2 Future work

Three main approaches can define the future lines of work on the basis of the results obtained.

1. The first line would be to optimize the model. Because of time limitations, hyperparameter optimization is not possible. A future line of work is to design or employ existent metaheuristic algorithms to perform model optimization. For example, the use of genetic algorithms or simulated annealing are suitable strategies to perform hyperparameter optimization. On top of that, prunning strategies could be applied to MiniGAN to analyse the relative importance of each architecture element. Removing one of the decoders or one of the conformers are some of the examples to be analysed. In order to improve MiniGAN performance, once the relative importance of each element is defined, least significant elements could be removed and higher complexity could be added to the most important elements.

2. An alternative line of work would be to explore other deep learning elements that can replace certain elements of MiniGAN. Particularly, an interesting option is to explore attention mechanisms that could be used in the latent space instead of conformers. There is currently a strong focus on research into attention-based architectures that promote efficiency, such as longformers and fastformers. They can replace the conformers. As they are efficient, a higher number of those elements could be used. Consequently, MiniGAN performance can be improved without severely compromising its efficiency. State Space Models have also shown outstanding results. Their use instead of attention mechanisms should also be analysed.

3. Finally, two paths could be followed regarding MiniGAN generalization and adaptability. On the one hand, it could be interesting to improve MiniGAN generalization capacity by testing the model on more diverse datasets. A different partition between the train and validation set is recommended, in order to include different types of noises. Therefore, when training MiniGAN variations, their ability to generalize can be more faithfully followed. In this scenario, more regularization techniques may be required. On the other hand, adapting MiniGAN to domain specific tasks pose a striking alternative. MiniGAN could be adapted for people with hearing and speech impairment. This defines an interesting alternative to improve accessibility and the quality of life of these people.

Bibliography

Abayomi-Alli, O. O., Damaševičius, R., Qazi, A., Adedoyin-Olowe, M., & Misra, S. (2022). Data augmentation and deep learning methods in sound classification: A systematic review. *Electronics*, *11*(22), 3795.

Abdulatif, S., Armanious, K., Sajeev, J. T., Guirguis, K., & Yang, B. (2021). Investigating cross-domain losses for speech enhancement.

Ai, Y., & Ling, Z.-H. (2023). Neural speech phase prediction based on parallel estimation architecture and anti-wrapping losses.

Alex, A., Wang, L., Gastaldo, P., & Cavallaro, A. (2023). Data augmentation for speech separation. *Speech Communication*.

Arias, F., Zambrano Nuñez, M., Guerra-Adames, A., Tejedor, N., & Vargas-Lombardo, M. (2022). Sentiment analysis of public social media as a tool for health -related topics. *IEEE Access*, *10*, 1–1. https://doi.org/10.1109/ACCESS.2022.3187406

Bäckström, T., Räsänen, O., Zewoudie, A., Zarazaga, P. P., Koivusalo, L., Das, S., Mellado, E. G., Mansali, M. B., Ramos, D., Kadiri, S., & Alku, P. (2022). *Introduction to speech processing* (2nd ed.). https://doi.org/10.5281/zenodo.6821775

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

Bello, I., Zoph, B., Vaswani, A., Shlens, J., & Le, Q. V. (2019). Attention augmented convolutional networks. *Proceedings of the IEEE/CVF international conference on computer vision*, 3286–3295.

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Benesty, J., Makino, S., & Chen, J. (2006). *Speech enhancement*. Springer Science & Business Media.

Braun, S., & Tashev, I. (2020). Data augmentation and loss normalization for deep noise suppression. *International Conference on Speech and Computer*, 79–86.

Braun, S., & Tashev, I. (2021). A consolidated view of loss functions for supervised deep learning-based speech enhancement. *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 72–76.

Cámbara, G., López, F., Bonet, D., Gómez, P., Segura, C., Farrús, M., & Luque, J. (2022). TASE: Task-aware speech enhancement for wake-up word detection in voice assistants. *Applied Sciences*, *12*(4). https://doi.org/10.3390/app12041974

Cao, R., Abdulatif, S., & Yang, B. (2022). CMGAN: Conformer-based metric gan for speech enhancement. *arXiv preprint arXiv:2203.15149*.

Chao, R., Cheng, W.-H., Quatra, M. L., Siniscalchi, S. M., Yang, C.-H. H., Fu, S.-W., & Tsao, Y. (2024). An investigation of incorporating mamba for speech enhancement.

Chen, S., Wu, Y., Chen, Z., Wu, J., Yoshioka, T., Liu, S., Li, J., & Yu, X. (2022). Ultra fast speech separation model with teacher student learning. *arXiv preprint arXiv:2204.12777*.

Chen, Z., Watanabe, S., Erdogan, H., & Hershey, J. R. (2015). Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks. *Proc. Interspeech 2015*, 3274–3278. https://doi.org/10.21437/Interspeech.2015-659

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Dang, F., Chen, H., & Zhang, P. (2022). DPT-FSNet: Dual-path transformer based full-band and sub-band fusion network for speech enhancement. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6857–6861.

Defossez, A., Synnaeve, G., & Adi, Y. (2020). Real time speech enhancement in the waveform domain. *arXiv preprint arXiv:2006.12847*.

de Oliveira, D., Welker, S., Richter, J., & Gerkmann, T. (2024). The pesqetarian: On the relevance of goodhart's law for speech enhancement. *arXiv preprint arXiv:2406.03460*.

Ding, H., Lee, T., Soon, Y., Yeo, C. K., Dai, P., & Dan, G. (2015). Objective measures for quality assessment of noise-suppressed speech. *Speech Communication*, *71*, 62–73.

Donley, J., Tourbabin, V., Lee, J.-S., Broyles, M., Jiang, H., Shen, J., Pantic, M., Ithapu, V. K., & Mehra, R. (2021). Easycom: An augmented reality dataset to support algorithms for easy communication in noisy environments. *arXiv preprint arXiv:2107.04174*.

Ferreira-Paiva, L., Alfaro-Espinoza, E., Almeida, V. M., Felix, L. B., & Neves, R. V. (2022). A survey of data augmentation for audio classification. *Congresso Brasileiro de Automática-CBA*, *3*(1).

Flores, S. (2019). *Variational autoencoders are beautiful*. Retrieved April 15, 2024, from https://www.compthree.com/blog/autoencoder/

Fu, S.-W., Hu, T.-y., Tsao, Y., & Lu, X. (2017). Complex spectrogram enhancement by convolutional neural network with multi-metrics learning. *2017 IEEE 27th international workshop on machine learning for signal processing (MLSP)*, 1–6.

Fu, S.-W., Liao, C.-F., Tsao, Y., & Lin, S.-D. (2019). MetricGAN: Generative adversarial networks based black-box metric scores optimization for speech enhancement. *International Conference on Machine Learning*, 2031–2041.

Fu, S.-W., Yu, C., Hsieh, T.-A., Plantinga, P., Ravanelli, M., Lu, X., & Tsao, Y. (2021). MetricGAN+: An improved version of metricgan for speech enhancement.

Fu, S.-W., Yu, C., Hung, K.-H., Ravanelli, M., & Tsao, Y. (2022). MetricGAN-U: Unsupervised speech enhancement/dereverberation based only on

noisy/reverberated speech. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7412–7416.

Fu, Y., Liu, Y., Li, J., Luo, D., Lv, S., Jv, Y., & Xie, L. (2022). Uformer: A Unet based dilated complex real dual-path conformer network for simultaneous speech enhancement and dereverberation.

Gelderblom, F. B. (2023). Evaluating performance metrics for deep neural network-based speech enhancement systems.

Gomes, H., Molholm, S., Christodoulou, C., Ritter, W., & Cowan, N. (2000). The development of auditory attention in children. *FBL*, *5*(3).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures [IJCNN 2005]. *Neural Networks*, *18*(5), 602–610. https://doi.org/https://doi.org/10.1016/j.neunet.2005.06.042

Gu, A., & Dao, T. (2024). Mamba: Linear-time sequence modeling with selective state spaces.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. (2020). Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Guo, H., Jian, H., Wang, Y., Wang, H., Zhao, X., Zhu, W., & Cheng, Q. (2023). MAMGAN: Multiscale attention metric gan for monaural speech enhancement in the time domain. *Applied Acoustics*, *209*, 109385.

Hansen, J. H. L., & Pellom, B. L. (1998). An effective quality evaluation protocol for speech enhancement algorithms. *5th International Conference on Spoken Language Processing (ICSLP 1998)*. https://api.semanticscholar.org/CorpusID:7794824

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. https://arxiv.org/abs/1502.01852

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, Y., & Loizou, P. C. (2006). Evaluation of objective measures for speech enhancement. *Ninth international conference on spoken language processing*.

ITU. (1996, August). *Itu-t recommendation p.800 - methods for subjective determination of transmission quality* (tech. rep. No. T-REC-P.800-199608-I). International Telecommunication Union. https://www.itu.int/rec/T-REC-P.800-199608-I/en

ITU. (2003). *Recommendation itu-r bs.1534 - method for the subjective assessment of intermediate quality levels of coding systems* (ITU-R Recommendation No. BS.1534). International Telecommunication Union. https://www.itu.int/rec/R-REC-BS.1534/en

Kawanaka, M., Koizumi, Y., Miyazaki, R., & Yatabe, K. (2020). Stable training of dnn for speech enhancement based on perceptually-motivated black-box cost function.

Keren, G., & Schuller, B. (2017). Convolutional RNN: An enhanced model for extracting features from sequential data.

Kierszbaum, S. (2020). *Masking in transformers' self-attention mechanism*. Retrieved April 17, 2024, from https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

Kim, E., & Seo, H. (2021). SE-Conformer: Time-domain speech enhancement using conformer. *Interspeech*, 2736–2740.

Kim, K., Wu, F., Peng, Y., Pan, J., Sridhar, P., Han, K. J., & Watanabe, S. (2022). E-branchformer: Branchformer with enhanced merging for speech recognition. https://arxiv.org/abs/2210.00077

Kingma, D. P., & Welling, M. (2022). Auto-encoding variational bayes.

Kinoshita, K., Delcroix, M., Yoshioka, T., Nakatani, T., Habets, E., Haeb-Umbach, R., Leutnant, V., Sehr, A., Kellermann, W., Maas, R., et al. (2013). The ¨REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech. *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1–4.

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kong, Z., Ping, W., Dantrey, A., & Catanzaro, B. (2022). Speech denoising in the waveform domain with self-attention. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7867–7871.

Kosar, V. (2022). *Cross-attention in transformer architecture*. Retrieved April 17, 2024, from https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, *37*(2), 233–243.

Kumar, A., Tan, K., Ni, Z., Manocha, P., Zhang, X., Henderson, E., & Xu, B. (2023). Torchaudio-squim: Reference-less speech quality and intelligibility measures in torchaudio.

Le, X., Chen, H., Chen, K., & Lu, J. (2021). DPCRN: Dual-path convolution recurrent network for single channel speech enhancement.

Le Roux, J., Wisdom, S., Erdogan, H., & Hershey, J. R. (2019). SDR–half-baked or well done? *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 626–630.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

Li, A., Zheng, C., Peng, R., & Li, X. (2020). Two heads are better than one: A two-stage approach for monaural noise reduction in the complex domain.

Li, Y., Sun, Y., Wang, W., & Naqvi, S. M. (2023). U-shaped transformer with frequency-band aware attention for speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Lin, J., van Wijngaarden, A. J. d. L., Wang, K.-C., & Smith, M. C. (2021). Speech enhancement using multi-stage self-attentive temporal convolutional networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*, 3440–3450.

Liu, G., Gong, K., Liang, X., & Chen, Z. (2020). CP-GAN: Context pyramid generative adversarial network for speech enhancement. *ICASSP 2020-2020*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6624–6628.

Liu, Y. C., Han, E., Lee, C., & Stolcke, A. (2021). End-to-end neural diarization: From transformer to conformer. *Interspeech 2021*. https://doi.org/10.21437/interspeech.2021-1909

Liu, Z., Jiang, Z., Luo, W., Fan, Z., Di, H., Long, Y., & Wang, H. (2024). TPT-GAN: Two-path transformer-based generative adversarial network using joint magnitude masking and complex spectral mapping for speech enhancement. In B. Luo, L. Cheng, Z.-G. Wu, H. Li, & C. Li (Eds.), *Neural information processing* (pp. 48–61). Springer Nature Singapore.

Loo, L. (2020). *Speech data augmentation: [audio/speech processing part 2]*. Retrieved June 17, 2024, from https://leonardloo.medium.com/part-2-audio-speech-processing-speech-data-augmentation-463dac823a8d

Lu, X., Tsao, Y., Matsuda, S., & Hori, C. (2013). Speech enhancement based on deep denoising autoencoder. *Interspeech*, *2013*, 436–440.

Lu, Y.-X., Ai, Y., & Ling, Z.-H. (2023). MP-SENet: A speech enhancement model with parallel denoising of magnitude and phase spectra. *INTERSPEECH 2023*. https://doi.org/10.21437/interspeech.2023-1441

Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., & Liu, T.-Y. (2019). Understanding and improving transformer from a multi-particle dynamic system point of view.

Luo, J., Wang, J., Cheng, N., Xiao, E., Zhang, X., & Xiao, J. (2022). Tiny-sepformer: A tiny time-domain transformer network for speech separation. *arXiv preprint arXiv:2206.13689*.

Macartney, C., & Weyde, T. (2018). Improved speech enhancement with the wave-U-net.

Maciejewski, M., Wichern, G., McQuinn, E., & Le Roux, J. (2020). WHAMR!: Noisy and reverberant single-channel speech separation. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 696–700.

Malinverno, M. (2023). *What is a spectrogram?* Retrieved June 17, 2024, from https://splice.com/blog/what-is-a-spectrogram/

Manocha, P., & Kumar, A. (2022). Speech quality assessment through mos using non-matching references.

McGuire, H. (2005). *Librivox*. Retrieved February 8, 2024, from https://librivox.org/

Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., & Poria, S. (2023). A review of deep learning techniques for speech processing. *Information Fusion*, 101869.

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks.

Music Technology Group of Universitat Pompeu Fabra, S., Barcelona. (2005). *Freesound*. Retrieved February 8, 2024, from https://freesound.org/

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

Ochieng, P. (2023). Deep neural network techniques for monaural speech enhancement and separation: State of the art analysis. *Artificial Intelligence Review*, *56*(Suppl 3), 3651–3703.

Oostermeijer, K., Wang, Q., & Du, J. (2021). Lightweight causal transformer with local self-attention for real-time speech enhancement. *Interspeech*, 2831–2835.

O'Shaughnessy, D. (2024). Speech enhancement—a review of modern methods. *IEEE Transactions on Human-Machine Systems*, *54*(1). https://doi.org/10.1109/THMS.2023.3339663

Pandey, A., & Wang, D. (2019). TCNN: Temporal convolutional neural network for real-time speech enhancement in the time domain. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6875–6879. https://doi.org/10.1109/ICASSP.2019.8683634

Pandey, A., & Wang, D. (2020). Densely connected neural network with dilated convolutions for real-time speech enhancement in the time domain. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6629–6633. https://doi.org/10.1109/ICASSP40776.2020.9054536

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*. https://doi.org/10.21437/interspeech.2019-2680

Park, H. J., Kang, B. H., Shin, W., Kim, J. S., & Han, S. W. (2022). MANNER: Multi-view attention network for noise erasure. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7842–7846. https://doi.org/10.1109/ICASSP43922.2022.9747120

Pascual, S., Bonafonte, A., & Serra, J. (2017). SEGAN: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*.

Pascual, S., Serra, J., & Bonafonte, A. (2019). Time-domain speech enhancement using generative adversarial networks. *Speech communication*, *114*, 10–21.

Peng, Y., Dalmia, S., Lane, I., & Watanabe, S. (2022). Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding. https://arxiv.org/abs/2207.02971

Phan, H., Le Nguyen, H., Chén, O. Y., Koch, P., Duong, N. Q., McLoughlin, I., & Mertins, A. (2021). Self-attention generative adversarial network for speech enhancement. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7103–7107.

Pingel, J., & Patel, S. (2017). *What are convolutional neural networks? | introduction to deep learning*. Retrieved April 15, 2024, from https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Pinheiro, P. H. O., & Collobert, R. (2013). Recurrent convolutional neural networks for scene parsing.

Pratap, V., Xu, Q., Kahn, J., Avidov, G., Likhomanenko, T., Hannun, A., Liptchinsky, V., Synnaeve, G., & Collobert, R. (2020). Scaling up online speech recognition using convnets.

Puckette, M. (2014, November). Acoustics for musicians and artists [DRAFT]. https://creativecommons.org/licenses/by-sa/3.0/

Reddy, C., Beyrami, E., Dubey, H., Gopal, V., Cheng, R., Cutler, R., Matusevych, S., Aichner, R., Aazami, A., Braun, S., et al. (2020). The interspeech 2020

deep noise suppression challenge: Datasets, subjective speech quality and testing framework. arxiv 2020. *arXiv preprint arXiv:2001.08662*.

Rix, A. W., Beerends, J. G., Hollier, M. P., & Hekstra, A. P. (2001). Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, *2*, 749–752.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Rosenblatt, F. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books. https://books.google.es/books?id=7FhRAAAAMAAJ

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, *45*, 2673–2681. https://doi.org/10.1109/78.650093

Shelf. (2024). *Why recurrent neural networks (rnns) dominate sequential data analysis*. Retrieved July 2, 2024, from https://shelf.io/blog/recurrent-neural-networks/#:~:text=First%2C%20RNNs%20process%20data%20sequentially,and%20require%20significant%20memory%20resources.

Skariah, D., & Thomas, J. (2023). Review of speech enhancement methods using generative adversarial networks. *2023 International Conference on Control, Communication and Computing (ICCC)*, 1–4.

Soni, M. H., Shah, N., & Patil, H. A. (2018). Time-frequency masking-based speech enhancement using generative adversarial network. *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 5039–5043.

Sound & video understanding teams at Google. (2017). *Audioset*. Retrieved February 8, 2024, from https://research.google.com/audioset/

Streijl, R. C., Winkler, S., & Hands, D. S. (2016). Mean opinion score (MOS) revisited: Methods and applications, limitations and alternatives. *Multimedia Systems*, *22*(2), 213–227.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., & Zhou, D. (2020). MobileBERT: A compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2011). An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(7), 2125–2136.

Tan, K., & Wang, D. (2018). A Convolutional Recurrent Neural Network for Real-Time Speech Enhancement. *Proc. Interspeech 2018*, 3229–3233. https://doi.org/10.21437/Interspeech.2018-1405

Tan, K., & Wang, D. (2019). Complex spectral mapping with a convolutional recurrent network for monaural speech enhancement. *ICASSP 2019 - 2019*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6865–6869. https://doi.org/10.1109/ICASSP.2019.8682834

Tan, K., & Wang, D. (2020). Learning complex spectral mapping with gated convolutional recurrent networks for monaural speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 380–390. https://doi.org/10.1109/TASLP.2019.2955276

Tang, H., Hsu, W.-N., Grondin, F., & Glass, J. (2018). A study of enhancement, augmentation, and autoencoder methods for domain adaptation in distant speech recognition. *arXiv preprint arXiv:1806.04841*.

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2022). Efficient transformers: A survey. *55*(6). https://doi.org/10.1145/3530811

Thiemann, J., Ito, N., & Vincent, E. (2013). DEMAND: A collection of multi-channel recordings of acoustic noise in diverse environments. *Proc. Meetings Acoust*, 1–6.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2017). Instance normalization: The missing ingredient for fast stylization. https://arxiv.org/abs/1607.08022

Valentini-Botinhao, C. (2017). Noisy speech database for training speech enhancement algorithms and tts models. *University of Edinburgh. School of Informatics. Centre for Speech Technology Research (CSTR)*.

Valin, J.-M. (2018). A hybrid DSP/deep learning approach to real-time full-band speech enhancement.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Veaux, C., Yamagishi, J., MacDonald, K., et al. (2016). Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit.

Verma, A. (2019). *Generative adversial networks*. Retrieved April 15, 2024, from https : / / www . linkedin . com / pulse / generative - adversarial - network - abhishek-verma/

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., & Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, *11*(12).

Wang, C., Yogatama, D., Coates, A., Han, T. X., Hannun, A. Y., & Xiao, B. (2016). Lookahead convolution layer for unidirectional recurrent neural networks. https://api.semanticscholar.org/CorpusID:126402114

Wang, K., He, B., & Zhu, W.-P. (2021). TSTNN: Two-stage transformer based neural network for speech enhancement in the time domain. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7098–7102.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wang, Y., & Wang, D. (2012). Cocktail party processing via structured prediction. *Advances in Neural Information Processing Systems*, *25*.

Wei, S., Zou, S., Liao, F., & weimin lang. (2020). A comparison on data augmentation methods based on deep learning for audio classification. *Journal of Physics: Conference Series*, *1453*(1), 012085. https://doi.org/10.1088/1742-6596/1453/1/012085

Weninger, F., Erdogan, H., Watanabe, S., Vincent, E., Le Roux, J., Hershey, J., & Schuller, B. (2015). Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR. *9237*. https://doi.org/10.1007/978-3-319-22482-4_11

Wichern, G., Antognini, J., Flynn, M., Zhu, L. R., McQuinn, E., Crow, D., Manilow, E., & Roux, J. L. (2019). WHAM!: Extending speech separation to noisy environments. *arXiv preprint arXiv:1907.01160*.

Wichern, G., & Lukin, A. (2017). Low-latency approximation of bidirectional recurrent networks for speech denoising, 66–70. https://doi.org/10.1109/WASPAA.2017.8169996

Williamson, D. S., Wang, Y., & Wang, D. (2016). Complex ratio masking for joint enhancement of magnitude and phase. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5220–5224.

Wilson, K., Chinen, M., Thorpe, J., Patton, B., Hershey, J., Saurous, R. A., Skoglund, J., & Lyon, R. F. (2018). Exploring tradeoffs in models for low-latency speech enhancement.

Wisam, E. (2022). *Why do rnns have short-term memory?* Retrieved July 12, 2024, from https://towardsdatascience.com/a-true-story-of-a-gradient-that-vanished-in-an-rnn-56437c1eea45

Wu, C., Wu, F., Qi, T., Huang, Y., & Xie, X. (2021). Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*.

Wu, Z., Liu, Z., Lin, J., Lin, Y., & Han, S. (2020). Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*.

Yin, D., Luo, C., Xiong, Z., & Zeng, W. (2019). PHASEN: A phase-and-harmonics-aware speech enhancement network.

Yu, C., Zezario, R. E., Wang, S.-S., Sherman, J., Hsieh, Y.-Y., Lu, X., Wang, H.-M., & Tsao, Y. (2020). Speech enhancement based on denoising autoencoder with multi-branched encoders.

Yu, F., & Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions.

Yu, G., Li, A., Zheng, C., Guo, Y., Wang, Y., & Wang, H. (2022). Dual-branch attention-in-attention transformer for single-channel speech enhancement. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7847–7851.

Yu, W., Zhou, J., Wang, H., & Tao, L. (2022). SETransformer: Speech enhancement transformer. *Cognitive Computation*, 1–7.

Zadorozhnyy, V., Ye, Q., & Koishida, K. (2022). SCP-GAN: Self-correcting discriminator optimization for training consistency preserving metric gan on speech enhancement tasks. *arXiv preprint arXiv:2210.14474*.

Zell, A. (1994). Simulation neuronaler netze. https://api.semanticscholar.org/CorpusID:47558162

Zhang, Z., Deng, C., Shen, Y., Williamson, D. S., Sha, Y., Zhang, Y., Song, H., & Li, X. (2020). On loss functions and recurrency training for gan-based speech enhancement systems. *arXiv preprint arXiv:2007.14974*.

Zheng, C., Zhang, H., Liu, W., Luo, X., Li, A., Li, X., & Moore, B. C. (2023). Sixty years of frequency-domain monaural speech enhancement: From traditional to deep learning methods. *Trends in Hearing*, *27*, 23312165231209913.

Appendix A: Legal Assessment

The speech enhancement models of this project are developed for research purposes. Hence, the Artificial Intelligence Regulation does not apply. However a legal assessment is conducted in this appendix.

The speech enhancement models could be classified as a minimal risk or limited risk system, depending on their use. These two categories have the same transparency obligations. If these models were commercialized, users should be informed of the presence of the AI systems behind them. Also, the scope and limitations of the systems should be disclosed. The systems are trained so that the enhanced signal resembles as much as possible the clean signal. As a consequence, users receive more clear information, which could translate into making more informed decision and preserving human autonomy.

The data used in this project is obtained from a public database, called VoiceBank + DEMAND. To avoid possible biases, an analysis of the dataset distribution is studied, previous to the development of the model. It should be noted that people over seventy years old is underrepresented, as well as unemployed people or people from working class. The female-male ratio is approximately the same and there is no bias with respect to different english accents. Although there is no explicit mention of it in the Valentini dataset report, one should also take into account the digital bias. People from upper classes have access to modern technology with better quality microphones. This implies that the audios from people from the upper social classes have better quality, which can affect the speech enhancement models learning.

Apart from biases regarding people, the models developed in this project are trained using specific types of noises. One should take into account that these models may not be appropriate for other types of noises. Also, speech enhancement models could be used to suppress audio which is not noise. This should be carefully supervised as these models could pose a risk in speech communications between machines and humans. For example, a malicious application could be designed to obtain tainted consent from customers. This could be done by suppressing audio with information to be consented to, violating the principle of informed consent. It could also fail to properly clean up a human being's voice in a recording, in order to misinterpret their words.

The data is anonymised by the public dataset in order to protect personal data. There is no sensitive data that would require special treatment within the dataset. Human oversight and monitoring have been present during the whole development of these systems. Several reviews and updates took place during the implementation of the models. Technical documentation is obtained during the model training and several

records are kept for security reasons and for the analysis of the systems traceability.

Overall, in this project several principles are taken into account to give raise to a trustworthy AI. These principles include privacy protection, bias and fairness analysis, performance assessment, transparency and security.

Appendix B: Waveforms and spectrograms of various audio

This appendix illustrates the waveforms and spectrograms of noisy and enhanced audios. The audios are not synthetic but recorded specifically for this project. The same version of the audios, with the same speaker and in the same environment is recorded. The speaker speaks in english and spanish. The noisy and enhanced audios in english and spanish are shown in figures 1 and 2 respectively. The models have been trained using only english audio. However, there is no significant difference when enhancing spanish or english audios. The background noises correspond to traffic noises, instrumental music and lyrical music. The models have been trained with traffic background sounds. Music is harder to clean. Lyrical music causes even more difficulties in the enhancing process, as the noise includes speech.

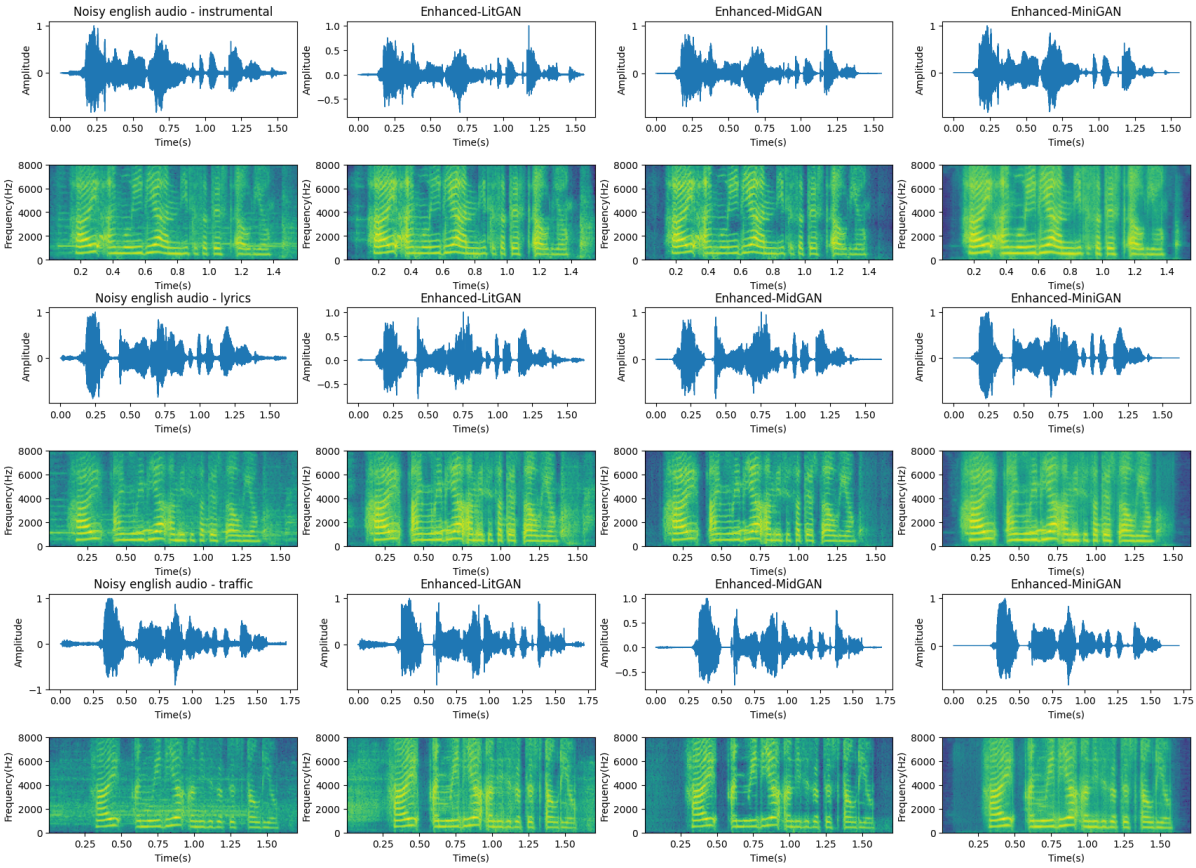

Figure 1: Waveforms and spectrograms of english audios. The first column refers to noisy audios. The second, third and fourth column correspond to the enhanced audios by LitGAN, MidGAN and MiniGAN models respectively. The first row of waveforms refers to audios with instrumental music as noises, the second one to lyrical music and the third one to traffic noises. The speaker is always the same.

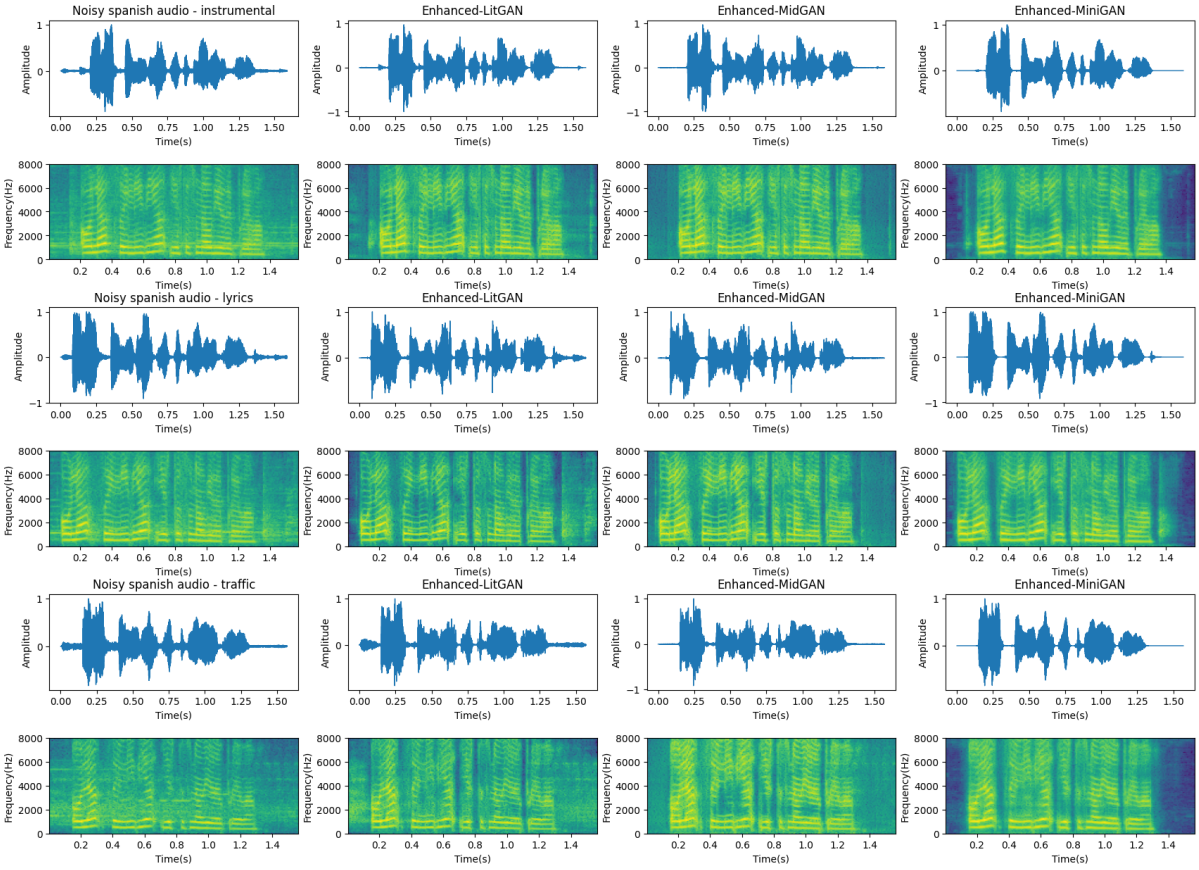

Figure 2: Waveforms and spectrograms of spanish audios. The first column refers to noisy audios. The second, third and fourth column correspond to the enhanced audios by LitGAN, MidGAN and MiniGAN models respectively. The first row of waveforms refesr to audios with instrumental music as noises, the second one to lyrical music and the third one to traffic noises. The speaker is always the same.