UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

Apprentissage automatique pour le codage cognitif de la parole

Machine learning for cognitive speech coding

Thèse de doctorat
Specialité: génie électrique

Reza Lotfidereshgi

Sherbrooke (Québec) Canada

Août 2022

JURY MEMBERS

Roch Lefebvre

Supervisor

Philippe Gournay

Co-supervisor

Tom Bäckström

Examiner

Eric Plourde

Examiner

Jean Rouat

Examiner

RÉSUMÉ

Depuis les années 80, les codecs vocaux reposent sur des stratégies de codage à court terme qui fonctionnent au niveau de la sous-trame ou de la trame (généralement 5 à 20 ms). Les chercheurs ont essentiellement ajusté et combiné un nombre limité de technologies disponibles (transformation, prédiction linéaire, quantification) et de stratégies (suivi de forme d'onde, mise en forme du bruit) pour construire des architectures de codage de plus en plus complexes.

Dans cette thèse, plutôt que de s'appuyer sur des stratégies de codage à court terme, nous développons un cadre alternatif pour la compression de la parole en codant les attributs de la parole qui sont des caractéristiques perceptuellement importantes des signaux vocaux. Afin d'atteindre cet objectif, nous résolvons trois problèmes de complexité croissante, à savoir la classification, la prédiction et l'apprentissage des représentations. La classification est un élément courant dans les conceptions de codecs modernes. Dans un premier temps, nous concevons un classifieur pour identifier les émotions, qui sont parmi les attributs à long terme les plus complexes de la parole. Dans une deuxième étape, nous concevons un prédicteur d'échantillon de parole, qui est un autre élément commun dans les conceptions de codecs modernes, pour mettre en évidence les avantages du traitement du signal de parole à long terme et non linéaire. Ensuite, nous explorons les variables latentes, un espace de représentations de la parole, pour coder les attributs de la parole à court et à long terme. Enfin, nous proposons un réseau décodeur pour synthétiser les signaux de parole à partir de ces représentations, ce qui constitue notre dernière étape vers la construction d'une méthode complète de compression de la parole basée sur l'apprentissage automatique de bout en bout.

Bien que chaque étape de développement proposée dans cette thèse puisse faire partie d'un codec à elle seule, chaque étape fournit également des informations et une base pour la prochaine étape de développement jusqu'à ce qu'un codec entièrement basé sur l'apprentissage automatique soit atteint.

Les deux premières étapes, la classification et la prédiction, fournissent de nouveaux outils qui pourraient remplacer et améliorer des éléments des codecs existants. Dans la première étape, nous utilisons une combinaison de modèle source-filtre et de machine à état liquide (LSM), pour démontrer que les caractéristiques liées aux émotions peuvent être facilement extraites et classées à l'aide d'un simple classificateur. Dans la deuxième étape, un seul réseau de bout en bout utilisant une longue mémoire à court terme (LSTM) est utilisé pour produire des trames vocales avec une qualité subjective élevée pour les applications de masquage de perte de paquets (PLC).

Dans les dernières étapes, nous nous appuyons sur les résultats des étapes précédentes pour concevoir un codec entièrement basé sur l'apprentissage automatique. un réseau d'encodage, formulé à l'aide d'un réseau neuronal profond (DNN) et entraîné sur plusieurs bases de données publiques, extrait et encode les représentations de la parole en utilisant la prédiction dans un espace latent. Une approche d'apprentissage non supervisé basée sur plusieurs principes de cognition est proposée pour extraire des représentations à partir de trames de parole courtes et longues en utilisant l'information mutuelle et la perte contrastive. La capacité de ces représentations apprises à capturer divers attributs de la parole à court et à long terme est démontrée.

Enfin, une structure de décodage est proposée pour synthétiser des signaux de parole à partir de ces représentations. L'entraînement contradictoire est utilisé comme une approximation des mesures subjectives de la qualité de la parole afin de synthétiser des échantillons de parole à consonance naturelle. La haute qualité perceptuelle de la parole synthétisée ainsi obtenue prouve que les représentations extraites sont efficaces pour préserver toutes sortes d'attributs de la parole et donc qu'une méthode de compression complète est démontrée avec l'approche proposée.

Mots-clés : codage de la parole, codage cognitif, apprentissage automatique, apprentissage des représentations, apprentissage non supervisé, réseaux de neurones.

ABSTRACT

Since the 80s, speech codecs have relied on short-term coding strategies that operate at the subframe or frame level (typically 5 to 20ms). Researchers essentially adjusted and combined a limited number of available technologies (transform, linear prediction, quantization) and strategies (waveform matching, noise shaping) to build increasingly complex coding architectures.

In this thesis, rather than relying on short-term coding strategies, we develop an alternative framework for speech compression by encoding speech attributes that are perceptually important characteristics of speech signals. In order to achieve this objective, we solve three problems of increasing complexity, namely classification, prediction and representation learning. Classification is a common element in modern codec designs. In a first step, we design a classifier to identify emotions, which are among the most complex long-term speech attributes. In a second step, we design a speech sample predictor, which is another common element in modern codec designs, to highlight the benefits of long-term and non-linear speech signal processing. Then, we explore latent variables, a space of speech representations, to encode both short-term and long-term speech attributes. Lastly, we propose a decoder network to synthesize speech signals from these representations, which constitutes our final step towards building a complete, end-to-end machine-learning based speech compression method.

The first two steps, classification and prediction, provide new tools that could replace and improve elements of existing codecs. In the first step, we use a combination of source-filter model and liquid state machine (LSM), to demonstrate that features related to emotions can be easily extracted and classified using a simple classifier. In the second step, a single end-to-end network using long short-term memory (LSTM) is shown to produce speech frames with high subjective quality for packet loss concealment (PLC) applications.

In the last steps, we build upon the results of previous steps to design a fully machine learning-based codec. An encoder network, formulated using a deep neural network (DNN) and trained on multiple public databases, extracts and encodes speech representations using prediction in a latent space. An unsupervised learning approach based on several principles of cognition is proposed to extract representations from both short and long frames of data using mutual information and contrastive loss. The ability of these learned representations to capture various short- and long-term speech attributes is demonstrated.

Finally, a decoder structure is proposed to synthesize speech signals from these representations. Adversarial training is used as an approximation to subjective speech quality measures in order to synthesize natural-sounding speech samples. The high perceptual quality of synthesized speech thus achieved proves that the extracted representations are efficient at preserving all sorts of speech attributes and therefore that a complete compression method is demonstrated with the proposed approach.

Keywords: speech coding, cognitive coding, machine learning, representation learning, unsupervised learning, neural networks.

ACKNOWLEDGEMENTS

I am extremely grateful for the multitude of factors that have given me the freedom to pursue this doctorate. I thank my supervisors, Profs Philippe Gournay and Roch Lefebvre, for all the opportunities and the confidence they gave me. Their academic discussions are always appreciated. I express my gratitude to my colleagues in the GRPA research group, several of whom participated in subjective evaluations carried out for this thesis. I thank all of the people who contributed to my financial support during my studies in Université de Sherbrooke. I would like to thank the jury, Profs Tom Bäckström, Eric Plourde and Jean Rouat for the evaluation of this thesis. Their generosity benefits the scientific community. Finally, I thank my family and my close friends, who stood by my side during the ups and downs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronyme	**Définition**
3GPP	3rd Generation Partnership Project
ACELP	Algebraic Code Excited Linear Prediction
AVQ	Algebraic Vector Quantizer
ANNS	Artificial Neural Networks
AC	Audio Coding
BWE	Bandwidth Extension
CN	Cochlear Nucleus
CC	Cognitive Coding
CNG	Comfort Noise Generation
CNNS	Convolutional Neural Networks
CPC	Contrastive Predictive Coding
DAES	Denoising Autoencoders
DFS	Discrete Fourier Series
EVS	Enhanced Voice Services
ERB	Equivalent Rectangular Bandwidth
BP	Error Backpropagation
EC	Error Concealment
FFT	Fast Fourier Transform
FFNNS	Feed-Forward Neural Networks
GAN	Generative Adversarial Network
GSC	Generic Audio Signal Coding Mode
GC	Generic Coding
IC	Inactive Coding
JBM	Jitter Buffer Management
LSF	Line Spectral Frequencies
LSP	Line Spectral Pairs
LP	Linear Prediction
LSM	Liquid State Machine
LSTM	Long Short-Term Memory
LTP	Long-Term Prediction
MOS	Mean Opinion Score
MSE	Mean Squared Error
MGB	Medial Geniculate Nucleus
MDCT	Modified Discrete Cosine Transform
NELP	Noise-Excited Linear Prediction
OL	Open-Loop
PESQ	Perceptual Evaluation Of Speech Quality
PSP	Postsynaptic Potential
PCA	Principal Components Analysis
PPP	Prototype Pitch Period

Acronyme	**Définition**
PSTN	Public Switched Telephone Networks
GRU	Gated Recurrent Unit
RNNs	Recurrent Neural Networks
ResNets	Residual Neural Networks
SNR	Signal To Noise Ratio
SC-VBR	Source-Controlled Variable Bit-Rate
STDP	Spike Timing Dependent Plasticity
SNNs	Spiking Neural Networks
SGD	Stochastic Gradient Descent
SON	Superior Olivary Complex
SVM	Support Vector Machine
TCX	Transform Coded Excitation
TC	Transition Coding
UC	Unvoiced Coding
VAE	Variational Autoencoder
VQ-VAE	Vector-Quantized Variational Autoencoder
VAD	Voice Activity Detection
SAD	Voice/Sound Activity Detector
VC	Voiced Coding

CHAPTER 1

Introduction

1.1 Problem statement

For decades, speech has been coded using classical signal processing methods such as linear adaptive filters, linear predictors, analysis-by-synthesis search of an optimal excitation signal, and waveform matching in a perceptually weighted domain. In the range of medium bit rates such as the one used in mobile telephony (around 13 kbits/s), the primary objective and main strategy was to synthesize a speech signal that is as close as possible to the original one using objective measures such as the signal-to-noise ratio (SNR) measured on short segments of speech signal. Most of the time, a weighted SNR is used so that coding noise is properly shaped according to the properties of human perception, which is the second main strategy used in this range of bit rates.

It is a known fact that objective measures do not always correlate well with perceived speech quality. The short-term waveform-matching strategy does not correspond to what is known about the biological mechanism of hearing, and speech is processed very differently in the human auditory system compared to conventional signal-processing based coding methods. Biological speech processing includes a preprocessing of speech in the auditory periphery, hierarchical processing in several auditory nuclei, and a primary processing in the auditory cortex. The combination of data-driven acoustic features (bottom-up knowledge) with context-driven perceptual conclusions (top-down knowledge) in the auditory system results in a long-term processing of the speech signal. Although classical speech coders use only a fraction of what is known about human auditory perception, they are already extremely efficient. Perceptual transform coders such as MP3 [1], AAC [2] and HE-AAC [3] use even more, but not all, of this knowledge. However, they are general-audio codecs that normally require a higher bit rate than speech-specific coders to achieve the same level of subjective quality on speech signals.

Long-term processing of the speech signal, similar to that done in the human auditory system, is a complex task. On a large temporal scale, speech signals are not stationary. Traditional manually-engineered approaches do not scale well for processing of large chunks of signal, either because their complexity grows very rapidly with size of the chunk, or because the objective can not be clearly defined for a non-stationary signal. Alternatively,

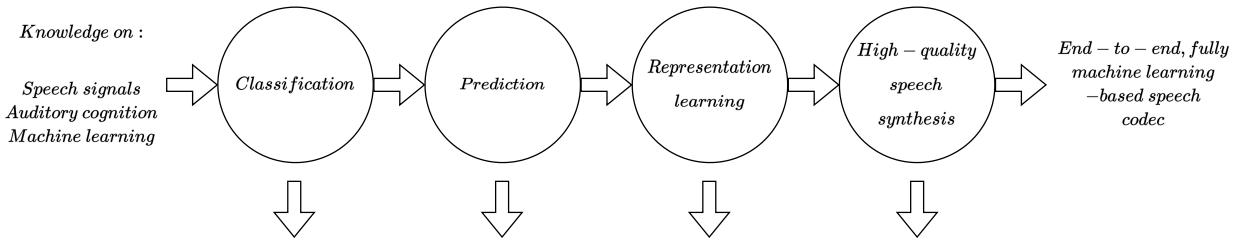
machine learning tools such as artificial neural networks have recently proven to be very successful in dealing with all sorts of complex signals and have been used in many speech-related applications. Many speech databases have also been created for training machine learning tools. In the field of speech coding, large sections of machine learning knowledge and tools have simply not been used in the past, either because of a lack of processing power, or simply because this knowledge is incompatible with established strategies of speech coding. There is still much to be learned and there is a potential to improve speech and audio coders.

1.2 Opportunity for contributions

Conventional strategies such as waveform matching operate on short (normally less than 30ms) frames of audio signal, parametrize each frame of signal independently of the others, and reconstruct it in such a way that it minimizes some objective measures such as the SNR. Although conventional speech coding approaches are very good in capturing details of the speech signal along with reducing the bitrate, recent evidence from the Speech and Audio Research Group at the Université de Sherbrooke and other research groups showed that some higher-level speech information, like speaker identity or emotion, can be degraded [4].

Due to the long-term nature of such lost information, we postulate that some long-term attributes of the audio signal are not completely captured and preserved by current techniques. Processing the speech signal in the long-term to extract and code its long-term attributes is a new task which includes the selection or design of the necessary tools and strategies. Tools and techniques developed in other fields of speech processing such as speech recognition have been shown to be efficient at capturing some of the desired long-term details of speech. These methods have the potential to be introduced into current coders in order to preserve the aforementioned attributes. Artificial neural networks are very successful in many speech-related applications and seem to be good candidates to recognize, extract and code the attributes not adequately captured by current codecs. Finally, many biological and perceptual theories have been proposed that were either not computationally implementable until recently or just neglected by speech coding research groups because speech coding methods, biological models and perceptual theories were developed separately. These relevant and complementary disciplines are a source of inspiration for the improvements of speech codecs. In conclusion, in recent years, several new tools and methods emerged that lead to successful results in many signal processing areas. The research questions addressed by this project is therefore: **How can machine learning tools be used to extract a compact representation of all short and**

long-term perceptually relevant attributes of the speech signal and to reconstruct a natural-sounding signal from this representation? To test these new emerging methods in the field of speech coding, we identified the tools with the most potential for the purpose of speech coding. We evaluated their ability to capture the speech information that current codecs miss. Hence, the contributions of this study are: first, extracting both short-term and long-term speech information and preserving them during the encoding and the decoding process; second, introducing new tools from machine learning and statistics in the structure of speech coders.



Objective	$p(C\|x')$	$p(y\|x)$	$p(C_s, C_l\|x)$	$p(x\|\hat{C}_s, \hat{C}_l)$
Tools	*SNNs*	*RNNs*	*CNNs* *RNNs*	*CNNs* *ResNet*
Results	*classifier*	*speech sample predictor*	*speech encoder*	*speech decoder*

Figure 1.1 Illustration of the tasks performed in this thesis where four tasks are performed. Each task further extends the objective towards the development of fully machine learning-based. Simultaneously, we explore different machine learning tools and design different key elements in a speech codec.

1.3 Proposed approach

After considering the nature of the problem and in order to answer the research question, we performed several tasks, each touching different key elements in a speech codec. This includes classification, prediction and speech representations. Although the results of each of these tasks can be a part of a codec on its own, each task provides insights and basis for the next and adds more elements from the field of machine learning towards the development of a fully machine learning-based codec. Fig. 1.1 illustrates the tasks performed in this thesis.

For the first task, we designed a speech classifier. The classification task can be performed on any of the types of information that we plan to preserve in the speech coding pro-

cess. We could choose to design a voice activity detector, a voiced/unvoiced classifier, a speech/music classifier, or any other type of classifier commonly found in speech coders. Instead, we decided to design an automatic emotion classifier. We focused on extracting a set of unsupervised features x' from an input signal x followed by a simple classifier to test if long-term attributes C are preserved and easily identifiable. To extract features x', a two stage design is used, including a biologically-inspired preprocessing and a processing stage using spiking neural networks (SNNs). This task is interesting because it is complex. Also, it deals with one of the long-term speech attributes which are the focus of the study.

For the second task, we designed a speech sample predictor. Prediction deals with a more complicated objective compared to classification and it involves producing a waveform rather than class labels in the previous task. Prediction of speech signal samples is central in many speech coders [1]. Improvement in sample prediction will affect performance of these coders. This task does not focus on decomposition and reconstruction of speech signals information, instead it focuses on decorrelation of the past information from each speech signal sample. Recurrent neural networks (RNNs) are used for capturing all sorts of correlations between speech samples. We propose the prediction task as a way to improve prediction-based speech coders. In addition, development of a machine learning-based predictor is a core element in the final step of the proposed approach i.e. development of fully machine learning-based codec.

For the third task, we designed and trained a neural network to recognize the building blocks of speech signals as representations in its layers. Perceptual theories of the human auditory system introduce a way to see the speech signal as building blocks (such as vocal tract movements, phones, phonemes,...) that are joined together to form speech signals. These theories, specially the ones that consider long-term attributes such as speech perception theories, can improve the performance of speech processing systems. Encoding fundamental building blocks of speech offers a great potential for further compression and enhanced speech quality that can not be achieved with conventional methods. In this task, we designed an encoder to extract two sets of representations C_s and C_l which encode short-term and long-term speech attributes respectively using convolutional neural networks (CNNs) and RNNs.

Finally, by designing a decoder, we verified the fact that the extracted representations are complete. While representations can be tested using labeled data for their capacity in preserving speech attributes, only resynthesizing speech signals with high perceptual quality proves that all sorts of speech attributes are preserved in representations. We used adversarial training, as an arbitrarily complex method to approximate speech subjective

quality measures, to synthesize speech signals from representations. A two stage design is used to resynthesize natural-sounding speech signals x from the quantized representations $\hat{C}_s$ and $\hat{C}_l$ extracted by the encoder using CNNs and residual neural networks (ResNet). Designing a decoder is the last step in the development of a fully machine learning-based codec and towards the completion of the objective of this project.

To summarize, the first two tasks explore useful design concepts for a speech codec with intermediate results which can be used as a building block of a codec. The last two tasks are the encoder and the decoder structure for a fully machine learning-based codec. An overview of the experiments is presented in Chapter 4 including further details about each step, a summary of experimental results, further insights and links between the development steps.

1.4 Overview of the thesis

This document is organized as follows. In Chapter 2 and 3, we briefly review the literature relevant to this project. In Chapter 2, speech and codec attributes are presented to clarify what are the speech signal characteristics and speech codec criteria to successfully encode and decode speech signals. An example of a modern codec, specifically the 3GPP enhanced voice services (EVS) codec [5, 6], is given to show how current technology deals with the dynamic nature of audio signals in general and how this technology can be compared with the proposed method. In Chapter 3, state-of-the-art machine learning techniques relevant to this project are explained to facilitate understanding of experimental results. An overview of the experiments is presented in Chapter 4 to explain insights and links between experiments. Chapters 5-8 are published papers presenting the experimental results of this project in the fields of speech classification, prediction, representations and machine learning-based compression, respectively. Finally, the thesis concludes with a comparison of the proposed approaches with existing approaches, a summary of the contributions, and the prospects for future research.

CHAPTER 2

Speech coding

2.1 Introduction

This chapter is the first of two chapters about the state of the art relevant to the research question. The general topics about speech and audio codecs are presented in this chapter. An expert researcher in the field of speech and audio coding can study other chapters independently and skip this chapter.

In Section 2.2 of this chapter, speech attributes are discussed and relevant theories for their understanding are introduced. Some examples from current codecs that utilize those theories are given. In Section 2.3, codec attributes and their trade-offs are considered. Next, in Section 2.4, building blocks of enhanced voice services (EVS) codec, which is representative of most modern speech codecs, are discussed to give a review of modern signal processing techniques used in the field of speech and audio coding. EVS exemplifies a common trend in modern standardized codec design, which is to increase the number of sub-blocks to deal with the diversity of contents in the speech signals and audio signals in general. This trend increasingly complicates the design of codecs, an issue that we address with the uniform codec design presented in Chapters 7 and 8.

2.2 Speech attributes

Speech attributes refer to any kind of characteristics in the speech signal that create a sort of sensation when it is perceived by a listener. Speech attributes can be divided into short-term and long-term attributes based on the time scale at which they can be identified. Short-term attributes are extracted from a short frame of speech signal (typicaly 10 to 30 ms). Long-term attributes, on the other hand, are present for a long period of time and in some cases for the whole duration of a speech signal. Long-term attributes are the result of within-speaker variability, between-speakers variability, and speaker's environmental properties. They are much more diverse than short-term attributes [7, 8]. These two categories of attributes are not independent, and to some extent, long-term attributes arise from short-term attributes over a longer period of time.

In this section, speech attributes are discussed based on their relationship with the physiological structure of the human speech production and auditory systems as well as some auditory perceptual theories.

2.2.1 Human speech production system

Speech is a non-stationary signal which is produced by air pressure passing through an acoustic filter. For unvoiced speech, air pressure is a noise-like turbulence, while for voiced speech, it also includes a quasi periodic vibration produced by vocal folds. The frequency of vibrations is called fundamental frequency or pitch. The vocal tract acoustic filter is mainly characterized by the peaks of its frequency response which are called formants.

The speech production system can be modeled using a source-filter model. The linear source-filter model of speech production, commonly used in the field of speech coding, assumes that the source of the speech sounds is independent of the filter [9]. In this model speech is decomposed into source and filter components using short windows of speech signal. When using a source-filter model, the characteristics of the source and the filter are examples of short-term attributes. More elaborate models of speech production consider nonlinear effects of source and filter interactions [10].

2.2.2 Auditory system

The human auditory system consists of several subsystems. There is first the auditory periphery, then millions of neurons in various auditory nuclei that perform various intermediate auditory stages, and finally the auditory cortex.

The peripheral auditory system includes the outer ear, the middle ear and the inner ear. The outer ear can be seen as an acoustic resonant cavity which modifies the sound pressure in the frequency range of 2-7 khz. The middle ear provides some impedance matching and also amplifies the frequencies above 100 hz with a maximum amplification around 1 khz. The inner ear (cochlea) is the organ of signal transduction. It performs a frequency-to-place mapping. Finally, hair cells in the cochlea convert mechanical vibration into electrical pulses.

Signals produced by hair cells are sent to the intermediate auditory stages. These stages include multiple nuclei [1] which all play a different part in auditory processing. The functionalities of these nuclei are not all known but some facts about them are well understood. For example, it is known that they perform tasks such as encoding binaural cues for source localization and preprocessing complex sounds. The auditory cortex, including AI and

1. The cochlear nucleus (CN), the superior olivary complex (SON), the inferior colliculus (IC) and the medial geniculate nucleus (MGB)

AII, is the final stage of the auditory system. The auditory cortex includes neurons which receive projections from auditory nuclei and other regions of the auditory cortex [11].

The properties of the peripheral auditory system, and especially those of the cochlea, are the most commonly used in speech coders. Logarithmic frequency scales including mel, Bark and ERBs correspond to the properties of the human ear and are used in perceptual codecs such as AAC and MP3. These codecs model the peripheral auditory system by analysing short frames of signal. Therefore, it can be considered that their strategy for extracting and encoding parameters aims at preserving short-term attributes in the spectral domain.

Identifying speech attributes that result from neural processes have also been studied to some extent and the evidence of phonetic features [12] and representations of multi-talker speech perception [13] are identified in the human cortex. In parallel with studies of the human brain, machine learning tools are also developed with the ability to process information in comparable ways. Tools from artificial neural networks (ANNs) such as convolutional neural networks (CNNs) and recurrent neural networks(RNNs) are often compared to the way that the brain achieves different types of processing. Non-linearities and a long receptive field makes ANNs capable of processing long-term attributes. In recent years, techniques such as GANs [14], Wavenet [15] and WaveRNN [16] with receptive fields in the range of hundreds of milliseconds have been used for speech synthesis and bandwidth extension and VAE's [17] have been employed to extract speech representations. While speech processing methods based on these approaches show improvements in perceptual quality of speech [18], ANN-based approaches have not been used in standardized speech codecs yet. These methods will be discussed in detail in Chapter 3 of this thesis.

2.2.3 Auditory perception

The functionality of the auditory system can be understood not only from its physiological organization, but also from auditory perceptual theories. Auditory perception is the process by which people sense, organize and interpret auditory information. Theories of auditory perception can be divided into two groups. One group addresses speech attributes such as pitch perception, localization and speech perception. Another group focuses on examining variations in speech signals that do not disturb the perceived attributes in a significant way or are not perceived by a listener at all. Examples of these theories are hearing thresholds and masking phenomenon. We first present the first group of perceptual theories. We then discuss the second group which played a much more significant role in the development of current speech and audio codecs.

Pitch is a perceived quantity relating to tones and complex sounds. In the case of a single tone, the perceived pitch can be related to the operation of the peripheral auditory system, which operates on a logarithmic (mel) scale. For complex tones, things are quite different. For example, some combinations of high-frequency tones can be perceived as low-pitch signals. Such examples suggest the involvement of structures beyond the peripheral auditory system for pitch perception [19]. In the case of speech, pitch perception is related to fundamental frequency but still has a long-term nature, and it is believed that it contributes to the identification of speaker, age, gender and emotions [11].

Speech perception involves many types of processes from ear to brain. It starts with the extraction of acoustic cues, such as formants, and continues with the formation of phonemes, syllables, words and sentences. Speech motor theory is one of the traditional speech perception theories which relates speech perception to movements of the speech production system. It also indicates that motor commands involved in speech production also help in speech perception [20]. Another theory in speech perception is the trace model which decomposes speech perception into three levels of features, phonemes and words. In addition to this bottom-up process of assembling parts, this theory puts emphasis on interactions with a top-down process (prior knowledge) [21].

Sound localization is the result of the way the auditory system processes spectral cues as well as binaural cues such as interaural time difference and interaural level distance. Qualitative characteristics depend on the sound source characteristics as well as environmental properties. Source characteristics include harmonic content, temporal envelope and spectral envelope (these types of characteristics are also called timbre). Properties of the environment, which can be characterized by methods such as room impulse response, also influence the sound quality. Finally, auditory scene perception consists in grouping and segregating the different sound sources based on their location, pitch, timbre, etc. [11].

The majority of speech attributes discussed in this section have a long-term nature, hence extraction of such attributes were not considered in standardized speech codecs due to the common short-term processing methods. Instead, subjective tests are used during the development phase of these codecs to measure the extent to which such attributes are preserved. However, the second group of perceptual theories have been frequently used in classical codecs over the last few decades since those theories are applicable for short-term processing of speech signals. As a result, instead of extracting long-term features, codecs attempt to encode and decode speech signals while limiting the short-term artifacts in forms that can not be perceived by a listener. These perceptual theories describe

limitations of the auditory system and the most notable ones are hearing thresholds, temporal and spectral resolution, and masking phenomena.

Hearing threshold and resolution

Hearing threshold is a minimum threshold under which an audio signal can not be heard; it is a function of the signal's frequency and duration. A pure tone is used to measure auditory threshold. The lowest threshold happens between 1 and 3 kHz and the threshold increases for both lower and higher frequency tones. If the duration of the tone is less than 500 ms, the hearing threshold depends on the duration of the tone and is higher for shorter tones. Above hearing threshold, the temporal resolution of hearing is variable and depends on two main processes: the time pattern in each frequency channel and the time patterns across multiple frequency channels. Overall, the resolution decreases as the sound intensity increases [22]. Spectral resolution also depends on signal intensity and frequency. In general, changes in signal intensity have a stronger effect at low frequencies [23].

In audio coding, the hearing threshold information can be used to define the acceptable noise floor for a system. Spectral and temporal resolution are also useful in perceptual codecs. Audio quality at low bitrates can be improved significantly if spectral and temporal resolution of input signal is considered.

Time and frequency masking

Presence of other sounds can affect the threshold of hearing for each sound. This phenomenon is called masking and it depends on relative time, frequency and intensity characteristics of sound components.

In frequency masking, the energy content of the signal in one frequency band can affect the hearing threshold in other energy bands. Frequency masking is a result of the behavior of the basilar membrane. Excitation of the auditory system happens in a frequency range wider than the sound's bandwidth. A similar phenomenon can be observed in the time domain. It is called temporal masking and is a result of the temporal inertia of the basilar membrane. Before and after an intense sound, the human auditory threshold changes for a limited duration of time because of temporal masking. Temporal masking can persist up to 200ms depending on the masking signal's strength.

The excitation patterns in frequency domain are used in codecs to calculate a curve of masking which determines the level of quantization noise that can be tolerated in each frequency band. The shape of auditory filters are also determined by these patterns. Simplified forms of auditory filters, known as critical bands and equivalent rectangular bandwidth (ERB), are used for audio coding. [23]. Temporal masking is also used in speech coding to adjust the block size and quantization in time domain [24, 25].

2.3 Codec attributes

The main characteristics of codecs are complexity, delay, bit rate, quality and robustness. Codecs aim for a trade-off between these measures for some target applications. In this section, the main characteristics of speech codecs are explained.

2.3.1 Algorithmic complexity

The complexity of a codec is related to memory and processing power usage. The complexity is a major factor of hardware cost and energy usage. The monetary cost is important in all applications. The energy consumption is also important in some applications, especially for devices powered by batteries, and some devices have very limited computing power.

2.3.2 Delay

The algorithmic delay is the maximum delay caused by the coding algorithm between the entry of a speech sample into the encoder and its output from the decoder. In real world applications, there are other sources of delay. The data transmission time on the network and the calculation time of devices also should be added to the delay time. Delay is an important factor in real-time conversations since more than 300 ms round-trip can cause difficulties in communication between people [26].

2.3.3 Bitrate

Bitrate is determined by channel capacity for different applications. When transmission bandwidth is limited, it results in a trade-off between all other codec attributes. Some elements of codecs (such as entropy coding) produce variable bitrates which are not suitable for a channel with fixed bitrate. To mitigate this problem, some codecs have an increased algorithmic delay and spread information between multiple frames of speech to produce constant bitrate [27]. Codecs with multiple bitrates are also common since they can achieve the best quality for available channel capacity. Embedded (scalable) codecs are a family of codecs designed to have a bitstream arranged in layers, and some information in the bitstream can be discarded by communication chain components depending on the available capacity [26].

2.3.4 Quality

While speech quality depends on many complex factors, signal bandwidth is a major factor which is easily quantifiable . The narrowband telephony bandwidth is approximately from 300Hz to 3.4 kHz. This bandwidth is used in public switched telephone networks (PSTN) for wired or wireless communication. Most older low-bit-rate codecs aimed to code this audio bandwidth. To achieve higher quality, the next extension of bandwidth are wideband

(approximately 50-7000 Hz), superwideband (approximately 50-14000 Hz) and full band (approximately 20-20000 Hz).

Quality of speech signals can be measured based on objective or subjective measures. The most common objective measure is the signal to noise ratio (SNR), normally expressed in dB. Objective measures can be advantageous since their measurements do not depend on human interaction and the results are easily reproducible. On the other hand, they do not correlate well with subjective speech quality. Subjective measures rely on listeners to evaluate perceived quality of speech signals in a blinded test. One common subjective measure for quality is the mean opinion score (MOS) which has a scale between 1-5 with 5 being the maximum possible score [28]. There are alternatives to MOS such as MUSHRA [29] and AB test [30].

To evaluate new algorithms, subjective testing is normally favored over objective measures. However, time and costs required by these tests can be problematic since normally multiple measurements are required during the development phase of a codec. Therefore, objective quality measurements have been developed to approximate subjective quality measures. Normally, a model of the auditory system that includes modeling of the external ear, internal ear and basic brain functions is used to predict the subjective quality. An example of such measurement method is perceptual evaluation of speech quality (PESQ) [31].

2.3.5 Robustness

Communication networks can introduce unwanted effects on a coded data stream. For example, errors can happen for individual bits, or a packet of data can be lost completely. Normally, a packet loss concealment method is added to decoder to deal with lost data. In some networks, packet delays are unpredictable and many codecs have limited tolerance for delays. A delayed packet could be considered lost or in some cases a decoder can still use a delayed packet to update its internal status and minimize error propagation [32].

2.4 Modern standardized speech and audio codecs

To address the variety of needs and requirements in the market, many different speech and audio codecs have been developed. Most of the techniques used in their development is a combination of mathematical tools and strategies such as quantization, prediction, transformation, noise shaping and entropy coding [33]. This section explores some of the important methods used in modern speech coding. Enhanced voice services (EVS) [5, 6], a recently standardized codec, is chosen as an example of modern codecs to explain the technologies currently in use. Moreover, signal processing methods used in EVS provide further insights into the nature of speech signals.

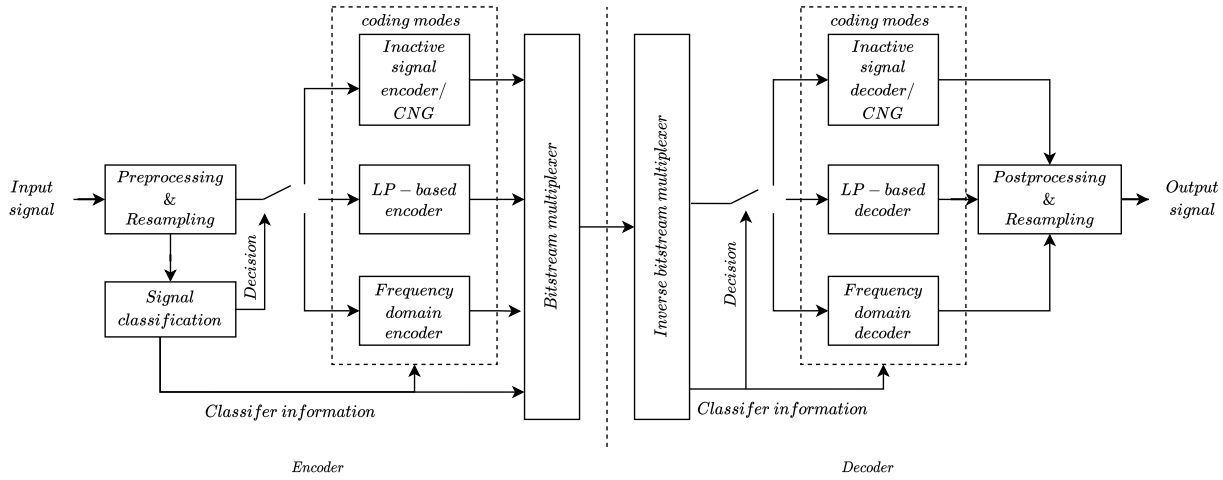


Figure 2.1 Overview of EVS codec.

2.4.1 General description of EVS

EVS was standardized by the 3rd generation partnership project (3GPP) in 2014. It is the successor of the mobile HD voice codec AMR-WB. It conforms to the low latency requirement of real-time communication and supports multiple bitrates for different bandwidths. Supported sampling rates are 8000, 16000, 32000 and 48000 samples/s with 16-bit uniform PCM format for single channel audio. Coding of stereo signals is also supported by means of coding two mono channels. The audio samples forming 20 ms input blocks are mapped to encoded blocks of bits, which may result in several possible choice of bitrates ranging between 5.9 kbps to 128 kbps depending on the configuration of the encoder. The combined algorithmic delay for the encoder and the decoder is 32 ms.

Using a built-in classifier, EVS chooses different strategies for coding of speech, music, mixed and noisy contents. It uses algebraic code excited linear prediction (ACELP), which is a variant of CELP [1], and modified discrete cosine transform (MDCT) for speech and audio compression, respectively. An overview of the EVS codec is illustrated in Fig. 2.1. Some other notable features present in EVS are source-controlled variable bit-rate (SC-VBR), voice/sound activity detector (SAD), comfort noise generation (CNG), error concealment (EC) for packet loss in communication networks, channel-aware mode to improve frame/packet error resiliency, and jitter buffer management (JBM) [34].

2.4.2 Preprocessing and resampling

Encoding speech signals starts with a set of common pre-processing steps. The purpose of preprocessing is to prepare input signals for the codec's functions as well as extracting some parameters for frame classification to decide which subset of functions should be used for

encoding each frame of speech signal. Preprocessing steps have been tuned over generations of standard codecs and have become common practices and were further developed and tuned in EVS. It should be noted that there is no clearly defined borderline between preprocessing steps and other functions in the codec. Nevertheless, preliminary functions, which also can be commonly found in other speech and audio codecs, are presented here as preprocessing steps.

High-pass Filtering

Very low frequency components of speech signals do not carry useful information and they can interfere with codec functions. As a result, the input signal is processed to eliminate undesirable low frequency components below 20Hz with the following transfer function in EVS:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.1}$$

The coefficients in Equation 2.1 are calculated for a -3 dB gain at the 20Hz cut-off frequency for different signal bandwidths.

Sub-band analysis

All input signals are upsampled to 48kHz and passed through a low delay filter bank with filters of 400 Hz bandwidth to decompose the input signal. The output of the filter bank is further analyzed to estimate parameters such as sub-band energies to be used in following operations such as classification of frame contents.

Resampling

Regardless of the input sampling rate, many internal blocks such as the linear predictive (LP) analysis, the long-term prediction (LTP) and the voice activity detection (VAD) algorithm only operate at 12.8 kHz (or 16kHz in some cases). Using specific bandwidth not only simplifies the optimization of these functions, it also reduces complexity of the codec by avoiding processing upper bands which do not carry significant information for functionality of these blocks. For the majority of encoder configurations, the input signal is converted from the input sampling frequency to 12.8 kHz, while an internal 16 kHz sampling frequency is only used when the input sampling frequency is 32kHz or 48kHz at bitrates higher than 13.2 kbps.

Pre-emphasis

Due to the wide dynamic range between low and high frequencies, the spectral tilt of speech signals is pronounced. To eliminate this problem, a pre-emphasis filter given by Equation 2.2 is used at the input to enhance the high frequency content of the signal. The

pre-emphasis factor b is tuned based on the bitrate, the bandwidth of the input signal and the selected coding mode.

$$H(z) = 1 - bz^{-1} \tag{2.2}$$

Spectral analysis

EVS extracts a 256-point fast Fourier transform (FFT) for signal classification functions and signal activity detection (SAD). Several energy-related parameters, such as average energy per critical band (which is calculated using a subset of frequency bins), energy per frequency bin and average total energy in 20 ms frames, are also calculated. Another set of parameters are calculated based on the history of extracted parameters to derive long-term features. Examples of such parameters are long-term active signal energy and relative frame energy.

Linear prediction analysis

Using linear prediction (LP), the encoder determines the coefficients of the synthesis filter of the CELP model. For each frame, two sets of sixteen LP coefficients are extracted, one using a 25 ms symmetric mid-frame analysis window and the other using a 25 ms asymmetric frame-end window. Next, to extend bandwidth, an adaptive lag window is applied to autocorrelations according to Equation 2.3. The lag window decays exponentially and is also a function of the fundamental frequency extracted with the open-loop pitch analysis (see Section 2.4.2).

$$W(i) = exp[-\frac{1}{2}(\frac{2\pi f_0 i}{f_s})^2], \;\; i = 1, 2, ..., 16 \tag{2.3}$$

Then, for stability purposes, some autocorrelation coefficients are modified, for example a white noise correction factor is added to the first coefficient. Finally, the Levinson-Durbin algorithm is used to convert the autocorrelations to LP coefficients. For quantization and interpolation purposes, the LP coefficients are transformed to line spectral pairs (LSP) and then to line spectral frequencies (LSF). The interpolated quantized coefficients are converted back to LP coefficients to construct the synthesis and weighting filters.

Open-loop pitch analysis

EVS performs an open-loop (OL) pitch analysis to calculate three estimates of the pitch for each frame. OL estimates are used to smoothen the pitch evolution contour and to confine the closed-loop pitch search (Section 2.4.4) to a small number of OL estimates.

The first two estimates of the pitch value are based on 10-ms segments of the current frame and the third one corresponds to the look-ahead window which is a 8.75 ms segment.

To calculate pitch values, the input signal is filtered by a pre-emphasis filter (Section 2.4.2) to eliminate the pronounced spectral tilt of the speech signal. Then the signal is filtered by a perceptual weighting filter which is derived from the LP filter transfer function A shown in Equation 2.4. The denominators have coefficients equal to the pre-emphasis filter and act as a de-emphasis filter. γ value is equal to 0.92 and is used to shape the LP transfer function. Such a combination of filters decouples the weighting in formant regions from the spectral tilt.

$$W(z) = \frac{A(\frac{z}{\gamma})}{1 - bz^{-1}} \tag{2.4}$$

Following perceptual weighting, the following steps are performed to calculate pitch values. Further details about these steps can be found in EVS Codec detailed algorithmic description (3GPP TS 26.445).

- Autocorrelation of decimated weighted signal is calculated.
- Correlation is reinforced with past pitch values using a triangular window.
- Normalized correlation is computed.
- In order to avoid selecting pitch multiples within each pitch value range, the lower section is further emphasized.
- Initial pitch value is determined by finding maximum values of correlation.
- Fractional open-loop pitch estimate is calculated for each subframe by interpolating correlation function.

Background noise energy estimation

To further improve the effectiveness of LP-based coding, EVS measures the level of background noise. Followings are examples of additional measures taken when speech over background noise is detected: enhancement of formants, use of dedicated cores for coding the background noise including a variant of generic audio signal coding mode (GSC) or the MDCT-based transform coded excitation (TCX) core and comfort noise generation (CNG) core.

Equation 2.5 describes the recursive process to update the background noise estimate per critical band in which $E_{CB}^{(-1)}$ and $E_{CB}^{(0)}$ stand for energy per critical band in the previous

and current frame, respectively. The noise energy in each critical band is initialized to 0.0035 dB.

$$N(i) = 0.9N_{CB}^{(-1)}(i) + 0.1\left[0.25E_{CB}^{(-1)}(i) + 0.75\left(E_{CB}^{(0)}(i) + 0.5E_{CB}^{(1)}(i)\right)\right] \tag{2.5}$$

The amount of background noise energy in EVS is estimated in two stages. In the first stage, noise energy estimated for each critical band can only be updated downward in an active signal. If signal energy in the current frame is below the previous estimate of background noise energy, the estimation will be updated. In the second stage, if an inactive frame is detected, EVS allows the noise estimation to be updated for the critical bands regardless of the current frame energy only if sensitivity to noise variation is low. Parameters such as spectral diversity, HF energy and tonal stability and complementary non-stationarity are some of the factors used to decide if the noise energy estimation can be increased.

Bandwidth decision

Regardless of the input sampling frequency, EVS examines the content of the input signal for existence of meaningful spectral content in different bands to choose one of the possible operational modes including: NB (maximum 4 kHz), WB (maximum 8 kHz), SWB (maximum 16 kHz) and FB (maximum 24 kHz). The decision is made by comparing the energy content in spectral regions with certain thresholds.

2.4.3 Classification of frame contents

Audio signals have very diverse content. To deal with such diversity, the general approach is to classify audio frames and select the proper compression method so that each frame is coded with maximum efficiency. The trade-off for increased coding performance is the added complexity of classification and the added complexity in the design of compression method for each class. Classification of content cannot be done efficiently without the context. As a result, EVS classifiers utilize long-term features from multiple frames of signal, while many other stages of coding only focus on processing the contents of a single frame. Transitioning between different compression methods without introducing artifacts is also a technical challenge. In this section, classification of contents by the EVS codec is discussed. The details on the corresponding coding approaches will be presented in later sections.

Signal activity detection

The purpose of signal activity detection (SAD) is to determine if speech or music or any meaningful signal content is present in a frame of speech signal. Information provided

by SAD is used by various subsequent modules to determine necessary trade-offs between quality and efficiency of coding. By distinguishing active speech, active music and inactive periods (or recording noise/background noise), EVS determines the appropriate operation mode as well as adaptive configuration of LP-based or MDCT-based coding cores.

The SAD module consists of three sub-SAD modules. Two of the modules are improved versions of a VAD from G.718. They operate on the spectral analysis of the 12.8 kHz sampled signal and provide preliminary activity decisions. The third one works on the QMF subband filter, which runs with the input sampling frequency, and is used to refine the decision and increase the reliability.

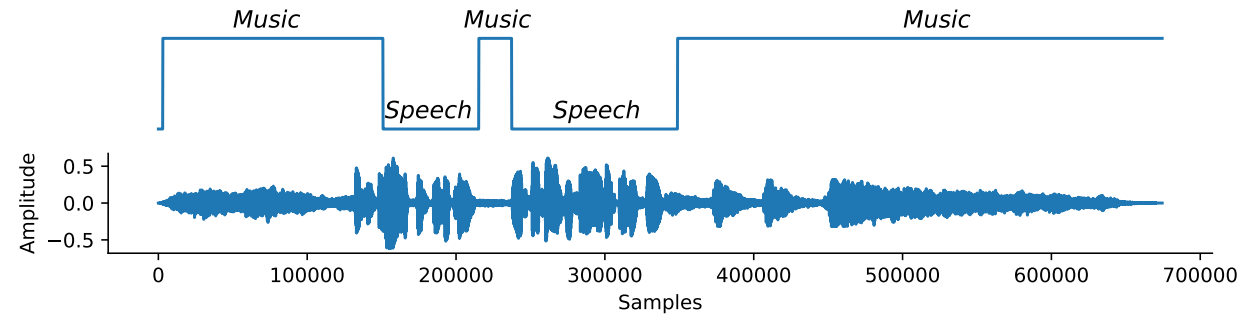
Coding mode determination

EVS uses dedicated LP-based coding modes for different speech classes. This principle is an extension of the mode classification technique used in the 3GPP2 VMR-WB standard [35] and ITU-T G.718 standard [36].

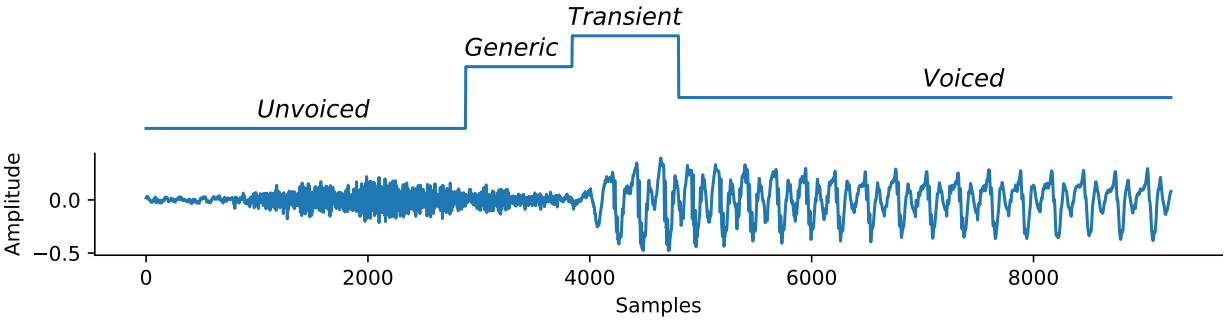
The LP-based core of EVS uses a signal classification algorithm with six coding modes which are customized for different classes of signal. This coding modes, arranged from highest priority to lowest, are: inactive coding (IC) mode, audio coding (AC) mode, the unvoiced coding (UC) mode, transition coding (TC) mode, voiced coding (VC) mode and generic coding (GC) mode. A classification algorithm determines the coding mode based on several parameters. Some of the parameters are optimized separately for NB and WB inputs. Based on classification results, different technologies are used for coding. Fig. 2.2 illustrates decisions taken by the classification algorithm in the EVS codec for two audio samples.

To carry out the mode decision, the SAD decision is first (Section 2.4.3), and if an inactive frame is detected, the IC mode is selected and the procedure is terminated. In the IC mode, two encoding technologies are used. For bitrates below 32 kbps, generic signal audio coder (GSC), which is a hybrid between an LP-based coder and a transform-domain coder, is used. Otherwise, only a transform domain method is used along with algebraic vector quantizer (AVQ) technology to quantize the frequency-domain coefficients.

Music signals are more complex than speech signals and can not be modeled efficiently with linear prediction. The AC mode is optimized to encode generic audio signals (particularly music) based on the GSC technology. A two stage speech/music classifier is used to detect suitable frames for the AC mode. First, a gaussian mixture model (GMM) is used to separate speech form music based on features such as the OL pitch, the spectral envelope (LSPs) and tonal stability. The second stage of speech/music classification only selects a



(a) Speech/Music classifier



(b) Coding modes for several speech frames

Figure 2.2 Decisions taken by the classifiers in the EVS codec. (a) Speech/Music classification for an audio signal with mixed contents (the sample includes both music and speech). (b) A detailed classification of a small speech segment selected from the long sample in (a).

subset of music frames (from the first stage) which are suitable for the GSC technology. In the second stage, up to the last 40 frames are analyzed based on their energy content to estimate stability measures and identify attacks in music signals. If a frame is not selected in the second stage, the decision is reverted back to speech and the CELP models are used.

Speech frames are further classified to voiced/ unvoiced frames based on the presence or absence of periodic components of the signal. To characterize periodicity, the following parameters are used as deciding factors: voicing, spectral tilt, sudden energy increase, total frame energy difference and energy decrease after a spike. If periodic components are missing, the signal is classified as unvoiced. For unvoiced frames, the UC mode is used to encode signals. The UC algorithm does not use adaptive codebooks for encoding the excitation of signals, instead a linear Gaussian codebook is used. If a voiced frame is detected, the VC mode is used to encode the quasi-periodic segment of signal which has a

smooth pitch evolution. ACELP is used to encode the frame and more bits are assigned to the algebraic codebook relative to a generic frame (GC mode).

In speech, transients occur at the beginning of speech segments and also when signal characteristics change, such as boundaries between voiced and unvoiced parts. Transients are characterized by rapid changes in signal energy and spectral distribution of the energy. If the TC mode is selected, due to the non-stationary nature of transient frames, the usage of past information is limited. The TC mode is limited to the most critical frames since limiting past information can have a negative impact on encoding clean speech when voiced frames follow voiced onsets.

If none of the above modes is selected, the most likely content of the frame is non-stationary speech, and the content is encoded with a generic ACELP model which allocates less bits to the algebraic codebook relative to viced frames.

Coder technology selection

The two generic principles for speech and audio coding are the LP-based (analysis-by-synthesis) method and the transform-domain (MDCT) method. The LP-based technology is based on CELP and is optimized for different bitrates, while the HQ MDCT technology is adopted as the transform domain method in EVS. There are also two hybrid methods, GSC and TCX technology, in which both LP-based and transform-domain-based methods are combined. The choice of technology for coding is decided based on the configurations of of the encoder, including the bitrate, the bandwidth as well as the selected coding mode. Even in a single mode, different technologies are usually used depending on the bitrate. The main reason for switching between technologies based on the bitrate is the tendency of CELP to saturate in coding efficiency after a certain bitrate. To achieve high speech quality at higher bitrates, frequency domain technologies are prefered.

2.4.4 LP-based coding modes

Speech-dominated contents at bitrates up to 64 kbps and generic audio at some lower bitrates are encoded using the analysis-by-synthesis linear prediction (LP) paradigm. The LP-based modes are variants of Algebraic Code-Excited Linear Prediction (ACELP) that are specialized for different speech classes. In these modes, the spectral envelope is modeled by LP coefficients. The LP excitation is encoded depending on the signal characteristics, such as whether it is voiced or unvoiced speech, general audio, inactive, etc. The coding efficiency of EVS depends on the bitrate but is rather independent of the input sampling rate. The input sampling frequency is either 12.8 kHz for lowbitrate or 16 kHz for high bitrates. Encoding the upper band in LP mode, which is not covered by LP-based coding, is achieved by means of bandwidth extension (BWE) technologies. Different BWE

strategies are used depending on the bitrate, including blind BWE, parametric BWE or full encoding of the upper band. Fig. 2.3 illustrates a speech sample encoded with the ACELP technology in both time and spectral domain. When the bitrate increases, the codec produces a waveform that is closer to the original waveform in the perceptual domain as well as in the original domain. In this section, details about waveform matching using the ACELP algorithm are presented.

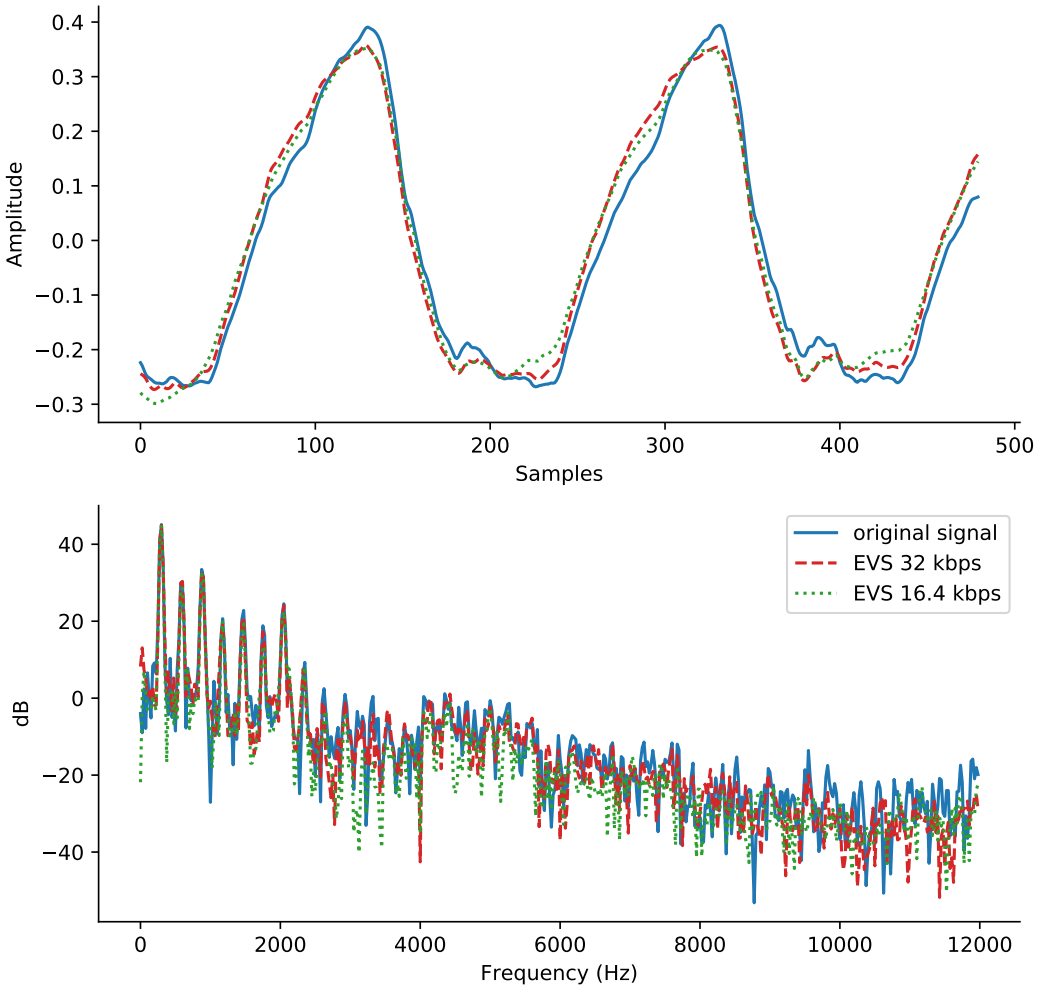


Figure 2.3 A speech sample encoded with the ACELP technology presented in time domain and spectral domain. When the bitrate increases, the codec produces a waveform that is closer to the original in a perceptually weighted domain.

LP-based coding starts with perceptual weighting, the same weighting approach used for OL pitch analysis (Section 2.4.2) and also calculated LP coefficients are quantized. Subsequently, the LP residual is calculated based on the following equation:

$$r(n) = s_{pre}(n) + \sum_{i=1}^{16} \widehat{a}_i s_{pre}(n-i), \quad n = 0, ..63 \tag{2.6}$$

where $\widehat{a}_i$ are the quantized LP filter coefficients and $s_{pre}(n)$ is the pre-emphasized input signal.

Excitation coding

Fig. 2.4 illustrates excitation coding in the GC and VC ACELP modes. In general, the excitation signal is coded using subframes of 64 samples. As a result, excitation is encoded four times per frame when the 12.8 kHz internal sampling rate is used and 5 times per frame in the case of the 16 kHz internal sampling rate. In some modes such as GSC, longer subrames are used to achieve lower bitrate. Adaptive and algebraic codebook search are further explained in following sections.

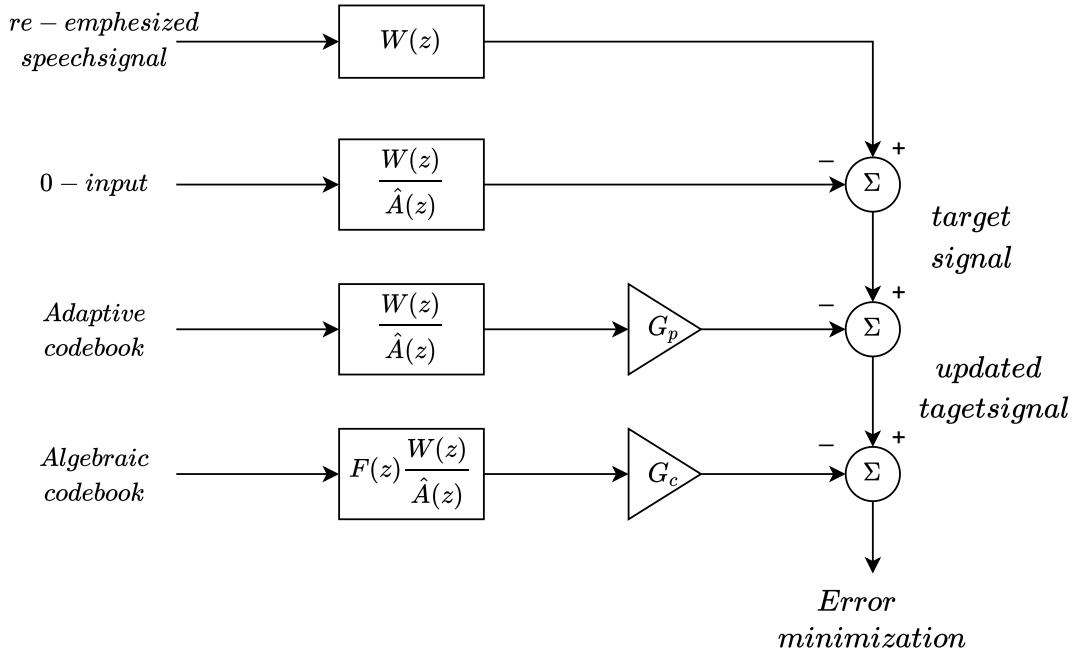


Figure 2.4 Schematic diagram of excitation coding in the GC and VC ACELP modes.

Adaptive codebook To perform the adaptive codebook search, a target signal computed according to Fig. 2.4 in which the zero-input response of the weighted synthesis filter $W(z)H(z) = A(z/\gamma_1)H_{de-emph}(z)/\hat{A}(z)$ is substracted from the weighted pre-emphasized

input signal computed using $W(z) = A(z/\gamma_1)H_{de-emph}(z)$ filter. Using the calculated target signal, the adaptive codebook encodes pitch parameters for every subframe including closed-loop pitch value and pitch gain (adaptive codebook gain). For the adaptive codebook search, a closed-loop pitch search is performed and the adaptive code vector is computed by interpolating the past signal using the selected fractional pitch values from the open-loop pitch estimates. For the closed-loop pitch search, a mean-squared weighted error between the target signal and the past filtered excitation is minimized which is equivalent to maximizing the correlation given by Equation 2.7.

$$C_{cl} = \frac{\sum_{n=0}^{63} x(n)y_k(n)}{\sqrt{\sum_{n=0}^{63} y_k(n)y_k(n)}} \tag{2.7}$$

where $x(n)$ is the target signal and $y_k(n)$ is the filtered excitation at delay k. The fractional pitch search is calculated by normalizing and interpolating Equation 2.7 and finding its maximum. Calculating fractional pitch values is concluded by applying a lowpass filter for further improvement. Subsequently, pitch gain is found by

$$G_p = \frac{\sum_{n=0}^{63} x(n)y_k(n)}{\sum_{n=0}^{63} y_k(n)y_k(n)} \tag{2.8}$$

Algebraic codebook Following the adaptive codebook, the algebraic codebook is used to model more pulses in the excitation signal and further reduce the error between the target signal and the input signal in the perceptual domain. The algebraic codebook structure and pulse indexing can be classified to 7-bit codebooks, 12-bit codebooks, 20-bit codebooks and above. For a 7-bit codebook, the algebraic vector only includes one non-zero pulse at one of 64 positions of a subframe. Six bits are used for the position of the pulse and one bit for the sign. For a 12-bit codebook, the algebraic vector includes two non-zero pulses. Positions in a subframe are divided into two tracks of odd and even positions. For each track, five bits are used for the position of the pulse and one bit for the sign. Similarly, up to four positions are encoded by dividing positions to more tracks and allocating bits to encode the position and the sign of pulses in the case of a 20-bit algebraic codebook. To extend the ability of codebooks to encode more pulses, a variety of joint indexing techniques are implemented to code multiple pulses in one track as well as multi-track joint coding.

Before codebook search in the algebraic domain, the algebraic code vectors are filtered using a pre-filter followed by a weighted synthesis filter. The pre-filter is given in Equation 2.9. It includes a periodicity enhancement part (denominator) and a tilt part (nominator):

$$F(z) = \frac{1 - \beta_1 z^{-1}}{1 - 0.85 z^{-T}} \tag{2.9}$$

where the parameter T is the integer part of the closed-loop pitch value in a given subframe and the parameter β_1 is related to the voicing of the previous subframe. Depending on the bitrate, the coding mode and the estimated level of background noise, a filter based on the spectral envelope is also included in the pre-filter to dampen frequencies between formant regions. Finally, the search criterion and the gain G_c for the algebraic codebook are derived from Equations 2.7 and 2.8 by replacing the target signal and the filtered excitation by the updated target signal and the filtered algebraic code, respectively.

The techniques covered in this section describe general notions of excitation coding and in practice the described approaches are tuned for different bitrates, bandwidths and operating modes in EVS. Most notably, in the IC mode and high bitrate GC mode, excitation coding also includes a frequency-domain coding stage in the form of a combined algebraic codebook. While the complexity of an algebraic codebook increases with codebook size, a combined algebraic codebook provides reasonable coding complexity.

Variable Bitrate (VBR) Coding

The amount of information in speech signals varies across time. Some contents, such as stationary voiced and unvoiced segments, can be encoded with lower bitrates with a minimal effect on subjective quality, while transients, due to lack of correlation with the past, carry more information and must be encoded with higher bitrates. This property of speech is used for VBR coding (sometimes called source controlled VBR Coding). Two low bitrate VBR coding modes at 2.8 kbps are included in EVS, specifically, the prototype pitch period (PPP) and the noise-excited linear prediction (NELP). These two methods are used for encoding stationary voiced and unvoiced frames, respectively.

The development of the PPP mode was inspired by the perceptual importance of periodicity in voiced speech. In this mode, a single representative PPP waveform is encoded in the frequency domain using discrete Fourier series (DFS). Due to the slow varying nature of waveforms in pitch-cycles of voiced segments, non-transmitted pitch-cycles are synthesized by means of interpolation. To encode the entire frame, in addition to a single pitch period, pitch lag values and phase offsets are encoded for time alignment of pitch periods within each frame. In the NELP coding mode, the objective is to encode excitation signals with a

minimum amount of data. As a result, the LPC analysis is the same for this scheme, while randomly generated sparse excitation signals are shaped in time and frequency domain to model prediction residual instead of adaptive and algebraic codebooks.

2.4.5 Frequency domain coding modes

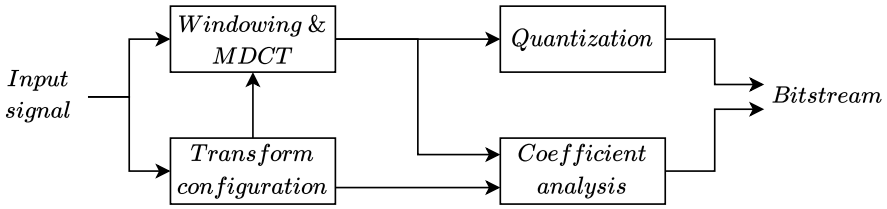


Figure 2.5 Frequency domain encoder based on MDCT.

Fig. 2.5 illustrates frequency domain coding in EVS, which is based on the MDCT transform. In general, the MDCT coding mode is used for signals other than speech, or when the bit rate for encoding is high enough for the transform coder to be used. Low delay communications constrains the overlap between consecutive frames in EVS compared to codecs such as AAC [2] that are designed for content distribution. Two variants of MDCT coding are implemented in the EVS codec to deal with the small frame overlap constraint: the Low-Rate/High-Rate High Quality-MDCT coding (LR/HR-HQ) [37], which is an advanced version of ITU G.719, and TCX [38], which is a low delay version of the homonymous core from the MPEG USAC standard [39]. To mitigate the effect of short frame overlap, several additionnal tools implemented in EVS, including an LTP post-filter, a harmonic model in the TCX algorithm, and a noise fill method [40]. For the MDCT cores, the coding of the upper band is considered as a part of the algorithm instead of being done by a bandwidth extension approach as in speech cores. Fig. 2.6 shows a speech sample encoded with MDCT in both time and spectral domain. Similar to ACELP, when the bitrate increases, the codec matches the waveform closer to its original shape in the perceptual domain and also converges to the original waveform with sufficient bitrate. In this section, details about waveform matching with MDCT are presented.

Starting with windowing, an asymmetrical low delay optimized (ALDO) window is used for long block transformation. The ALDO window is illustrated in figure 2.7 where L is 20 ms and L_z is 4,375 ms. The slopes of the window are made of sinusoidal functions raised to the power of a constant value which is tuned based on the sampling frequency.

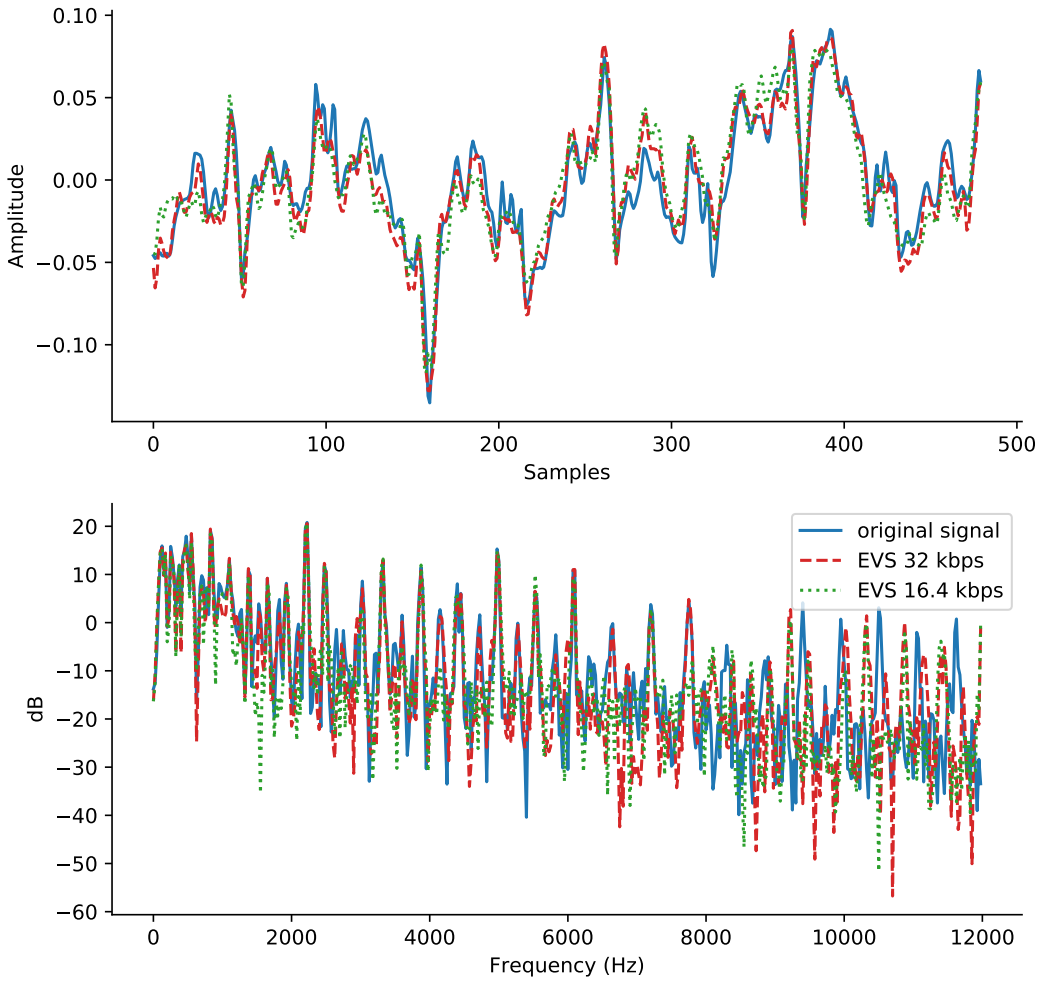


Figure 2.6 A speech sample encoded with MDCT is presented in time domain and spectral domain. When the bitrate increases, the codec produces a waveform that better matches the original in a perceptually weighted domain.

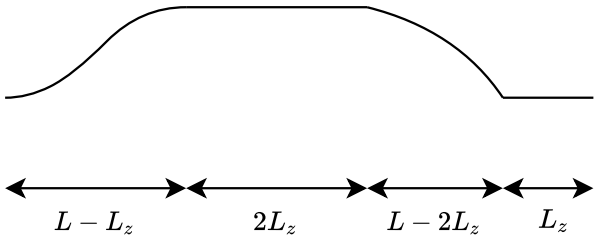


Figure 2.7 Asymmetrical low delay optimized (ALDO) window used for long block transformation in EVS.

After windowing the input signal, eDCT, which is a discrete cosine transform type IV variant with less storage and complexity requirement, is performed to transfer the signal to frequency domain. DCT type IV is described by Equation 2.10:

$$y(k) = \sum_{i=0}^{L-1} \widetilde{x}(i) cos\left[(n + \tfrac{1}{2}(k + \tfrac{1}{2})\tfrac{\pi}{L})\right] \tag{2.10}$$

where $y(k)$ is the k-th DCT component and $\widetilde{x}(i)$ is the windowed input signal.

MDCT-based TCX

To introduce noise shaping tools in spectral coding, an MDCT-based TCX mode is used in EVS. As a result, coding tools such as adaptive low frequency emphasis, temporal noise shaping, and noise filling can be used in this mode. The LPC parameters are calculated in the time domain and applied in the spectral domain. While the analysis methods are identical to that of ACELP, quantization of LSF can be different for some bitrates.

An example of how to apply LPC parameters in the spectral domain is by applying shaping gains to the MDCT spectrum. In this case, the MDCT coefficients corresponding to the CELP frequency range are grouped into sub-bands and multiplied by the corresponding LPC shaping gain.

High Quality MDCT coder (HQ)

The low-rate variant of HQ is limited to bitrates below 16.4 kbps and input signal is only encoded using the transient mode, the normal mode, and the harmonic mode. The harmonic mode is only available for SWB signals. The high-rate HQ is used for WB, SWB, and FB signals at bit-rates of 24.4, 32, and 64 kb/s. There are two more modes available in the high-rate HQ coder: the HVQ mode and the generic mode. A brief description of HQ MDCT modes is as follows: transient signals are handled by the transient mode using shorter transforms, harmonic signals are handled by harmonic mode, strongly harmonic signals are handled by the HVQ mode, and finally, for all other signals, the normal and generic modes are used.

The encoded parameters of HQ-MDCT are: the mode selection, energy envelope information, the quantized spectral coefficients, some LF parameters and the HF parameters. These parameters are extracted as follows. To encode the spectral envelope, the spectral coefficients are grouped into bands with unequal widths. The spectral envelope is calculated based on the quantized energies of bands, encoded using Huffman coding and finally the quantized energies are used for bit allocation. The calculated spectral coefficients are quantized using trellis vector quantization (TCQ) and uniform scalar quantization (USQ)

and then encoded based on the allocated bits for each frequency band. The level of the most significant spectral coefficients is also adjusted using an estimated gain then coded for transmission. Relatively few bits are also allocated for encoding the spectral bands which are not quantized in the HF regions.

2.4.6 Inactive signal coding/ CNG coding mode

The DTX/CNG strategy is to reduce the transmission rate by simulating background noise during inactive frames. Another benefit of this strategy is to make more efficient use of battery life in mobile communications. The regular DTX/CNG mode operates in bit rates up to 24.4 kbps; for higher bit rates, it is only used when the input signal power is low. The transmission rate is reduced by replacing background noise with CNG in the decoder. Comfort noise (CN) parameters are used to model the spectral and temporal content of the background noise at the encoder.

In the EVS Codec, two types of CNG algorithms are implemented: a linear prediction-domain based coding mode (LP-CNG) [41] and a frequency-domain based coding mode (FD-CNG) [42]. One of two modes is selected according to the input characteristics, and a different set of CN parameters is utilized for each coding mode. If the LP-CNG mode is used, four CN parameters are extracted and encoded: the low-band excitation energy, the low-band excitation envelope, the low-band signal spectrum, and the high-band energy. In the FD-CNG mode, the CN parameters include global gain and spectral energies which are grouped in critical bands. Finally, the extracted parameters are encoded using a vector quantizer.

2.4.7 Description of the decoder

The majority of decoder functions perform the inverse of the encoder's operations from quantized values. The decoder is also computationally less complex than the encoder since some functionalities such as classification of frame content, extraction of features for classification and codebook search are unique to the encoder. However, there are some operations that are unique to the decoder and that are performed in addition to the generation of signal components from transmitted data. The most notable such operations are bandwidth extension and frame loss concealment.

Bandwidth Extension

The EVS decoder performs estimates of signal regions where the transmitted signal representation is not complete. Noise fill and blind bandwidth extension are examples of techniques used for this purpose. Based on the characteristics of the input signal and on the coding mode (decided based on the low band part of signal), two technologies are used in EVS for bandwidth extension, time domain BWE (TD BWE or TBE) and multi-

mode frequency domain BWE (FD BWE). For speech signals, the high-band signal (or the super-high-band signal) is encoded by TD BWE, while multi-mode FD BWE is used for music and IC mode. In TBE, the output of the QMF filter bank is used to construct an estimation of the upper band signal from lower band using operations such as spectral flip and down mix. Decoding FD BWE is achieved based on spectral and time envelope indices which are extracted in the encoder. While for transient frames both time and spectral envelopes are constructed based on the received bitstream, in all other cases only spectral envelopes are used.

Frame loss concealment

Extrapolation algorithms for all of the coding modes are considered in the decoder in the case that a frame is lost in transmission. For both LP-based coding and frequency domain coding, information from the last received frame is used to extrapolate the signal and produce a smooth time evolution of signal from the last frame into the lost frame. Following a lost frame, received good frames are used to update the codec memory and to minimize the mismatch resulting from the lost frame. If the loss occurred for multiple consecutive frames and no reasonable extrapolation can be made, the signal is gradually faded to background noise or silence.

Another technique to improve the resiliency of codec against frame loss is the "channel aware" mode used in VoIP systems. In this mode, partial copies of the information from the last two frames are included in the current frame, and in the case of packet loss this information can be used to improve the recovery from the lost frame.

2.4.8 Other features

Backward Compatibility

Backward compatibility of a codec allows for interoperability with a previous generation of codecs in telecommunications. In the EVS codec, backward compatibility with the AMR-WB codec is considered using an interoperable (IO) mode. While this mode offers functions for decoding an AMR-WB bitstream, some improvements are made over AMR-WB by means of the post processing technologies that are implemented in EVS. Post processing of noisy content, mixed content and improved performance for low-level input signals are examples of improvements implemented in the IO mode relative to AMR-WB.

Jitter Buffer

Jitter in codecs is the deviation from true periodicity when receiving packets of encoded signal. Jitter buffers are buffers used to counter the introduced jitter. They operate by queuing received packets to ensure a continuous audio stream transmitted over the network. In the EVS decoder a jitter buffer management (JBM) solution is included

to compensate for variations in transmission delay. Depending on channel conditions, the JBM system uses time scaling methods and packet loss concealment to provide a continuous audio stream. JBM operation is balanced based on a trade-off between the amount of delay and the perceptual quality of speech.

2.5 Summary

In this chapter, the knowledge about speech and audio signals and codecs that is relevant to this thesis has been presented. An introduction to speech attributes makes it possible to understand speech and audio signals as well as the auditory system. This knowledge helps to justify certain strategies in codecs. Then, a summary of codec attributes makes it possible to define important criteria in designing codecs and to identify necessary trade-offs between these criteria.

An overview of the EVS codec, which is taken as an example of modern standardized codec is also provided with several goals in mind. First, EVS is a recently standardized codec and its attributes define a threshold of performance that any new research project should aim for. Second, techniques used in EVS further clarifies the nature of speech and audio signals and how this knowledge is exploited by technology currently in use. Finally, this description of EVS reveals some limitations of the current approach for codec design. Waveform matching strategies progressively increase the number of coding modes. This approach increases the chance of mistakes in classification of frame contents, and also increases the number of sub-blocks which should be designed and optimized. In other words, the proliferation of local frame-based optimizations in the codec tends to create global issues.

In the next chapter, we review the state-of-the-art machine learning tools and approaches that we will use to design a uniform codec that can process large chunks of speech signal, be optimized globally, and be used to build an alternative strategy to waveform matching. The alternative strategy will consist in extracting high-level and long-term attributes and preserving them in the synthesized speech.

CHAPTER 3

Machine learning in speech processing

3.1 Introduction

A sufficient knowledge about machine learning techniques is provided in this chapter to enable the reader to understand the experiments presented in Chapters 5 to 8. State-of-the-art machine learning-based speech coding techniques relevant to this thesis are also presented. An expert in the field of machine learning can skip this chapter.

In Sections 3.2 and 3.3, machine learning tasks and paradigms are introduced to explain how current machine learning knowledge is applied to optimization problems with a special focus on speech processing. In Sections 3.4, several artificial neural network (ANN) architectures relevant to this thesis are introduced, and in Section 3.5, training of ANNs is presented. Spiking neural networks (SNNs), an alternative to conventional artificial neural networks, are presented in section 3.6 and their training is also discussed. Finally, in section 3.7, generative models that are particularly interesting for speech compression and speech synthesis are presented.

3.2 Machine learning tasks

There are a wide range of tasks that can be performed with machine learning algorithms. Classification, regression and denoising are widely considered tasks in the field of speech processing. A more comprehensive description of machine learning tasks in general can be found in the Deep Learning [43] textbook.

Classification The objective of a classification algorithm is to recognize which one of the k categories the input sample belongs to. In other words, the model $y = f(x)$ produces a numeric value y for an input vector x. In probabilistic formulation, function f models the probability distribution over classes. Examples of classification tasks in speech coding are speech/music classification, speaker identification and emotion recognition. Classification algorithms are already an essential part of standardized speech codecs to distinguish signal frame contents and to choose the proper compression algorithm accordingly (see Section 2.4.3).

Regression The regression task is similar to classification task except that the output y has a continuous format. As for the classification task, a function f can be defined to model the probability distribution of outputs. Then an output can be generated by sampling the probability distribution. Linear regression, where f is a linear function, is a widely used regression method for predicting and forecasting because of its ease of use. A popular regression method in audio coding is the autoregressive function in which previously generated audio samples are fed to the model as an input to produce a sequence of signal samples. For example, in WaveNet [15], a generative model for raw audio is constructed to model the joint probability of a waveform $x = \{x_1, ..., x_N\}$ by factorizing it as a product of conditional probabilities as:

$$p(x) = \prod_{i=1}^{N} p(x_i | x_1, ..., x_{i-1}) \tag{3.1}$$

where each audio sample x_i is conditioned on the previous samples. After the probability function has been learned, it can be sampled to produce an audio waveform.

Denoising In denoising tasks, a clean input vector x is corrupted by an unknown noise source, and the machine learning algorithm is trained to predict the clean sample from the corrupted vector $\hat{x}$ by learning the conditional probability distribution $p(x|\hat{x})$. One of the common neural network structures used for denoising is the autoencoder (see Section 3.7.1 for autoencoders) and more specifically denoising autoencoders (DAEs) [44]. Their efficiency for speech enhancement has been clearly demonstrated [45, 46]. Adding noise to the input vector x also improves the robustness of the model in any machine learning task. If the noise is applied by forcing some values to be zero, denoising becomes very similar to dropout regularization, which is widely used for regularization of neural networks currently in use (Section 3.5.2).

Application of machine learning in the field of speech processing is not limited to the three tasks presented in this section. Other tasks such as automatic transcription and machine translation also deal with processing of speech signals. In some cases, other modalities or types of information, such as natural language, image and video, are combined with speech processing for achieving a machine learning task.

3.3 Learning paradigms

The ability of machine learning algorithms to learn from large amounts of data distinguishes them from traditional signal processing techniques. In machine learning, paradigms

are defined based on how data is utilized for learning. Supervised learning and unsupervised learning are widely used learning paradigms for speech processing.

Supervised learning In supervised learning, the outputs y as well as the inputs x are available for training, and the algorithm determines the relationship between them. In the probabilistic formulation, the algorithm estimates the probability distribution $p(y|x)$. Maximum likelihood estimation is one of the most common methods used for finding the best set of parameters for modeling the probability distribution. Logistic regression, support vector machine (SVM) and k-nearest neighbors are examples of supervised learning approaches. While supervised learning is the most commonly used method in machine learning, its use is limited by difficulties of collecting, either by humans or by automatic methods, reliable outputs y for training.

Unsupervised learning In general, the goal of unsupervised learning is to learn a density estimation similar to that learned by supervised learning, but without any information labeled by a supervisor. The goal is to find a data representation that preserves most of the useful information about the input, but in a simpler form than the raw input. The simplicity of representation can be defined in many different ways. Having lower dimension or sparsity are prefered attributes for a representation in many applications. Principal components analysis (PCA) and K-means clustering are examples of unsupervised learning approaches.

There are other categories of learning paradigms in machine learning depending on the nature of data. Aside from supervised and unsupervised learning, the most notable ones are semi-supervised and reinforcement learning. In semi-supervised learning, the output y is available to the learning algorithm only for a subset of samples. In reinforcement learning, the data is in the form of an environment instead of a fixed dataset, and feedback from the environment guides the learning algorithm to optimize the parameters.

3.4 Artificial neural network architectures

Initially, the development of artificial neural networks (ANNs) was inspired by biological neural networks. As a result of decades of research, a wide range of ANNs are now available. This section is dedicated to the subset of ANNs architectures that is used in the experiments presented in the Chapters 5-8 of this thesis, including convolutional neural networks (CNNs), recurrent neural networks (RNNs) and residual neural networks (ResNets). These architectures, because of their inherent ability for processing sequential data, are currently very common for processing speech signals. In this thesis, an appli-

cation of CNNs can be found in the experiments presented in Chapters 7 and 8. An application of RNNs also can be found in the experiments presented in Chapters 6 and 7. An example of utilization of ResNet blocks also can be found in the structure of the decoder in the cognitive speech codec presented in Chapter 8.

3.4.1 Feed-forward neural networks

Feed-forward neural networks (FFNNs) are a widely used architecture, and they are the basis for many other architectures such as CNNs and RNNs. A diagram of a FFNN is illustrated in Fig. 3.1 that includes l hidden layers and m unit in each layer. The network models a desired funcion $F(x)$ as a chain of nested functions $f_{l+1} \circ f_l \circ ...f_1(x)$ in which function f_i is implemented by i^{th} layer of the network. The functions of the different layers are recursively defined as:

$$\begin{aligned} f_i &= \phi(W_i^T f_{i-1} + b_i) \qquad 1 < i < l+1 \\ f_0 &= x \end{aligned} \tag{3.2}$$

in which $\phi(.)$ is the activation function (a nonlinear function such as sigmoid) and parameters W_i and b_i define an affine transformation. Back-propagation is commonly used to compute the gradients necessary for learning parameters of this model, which is presented in Section 3.5.2.

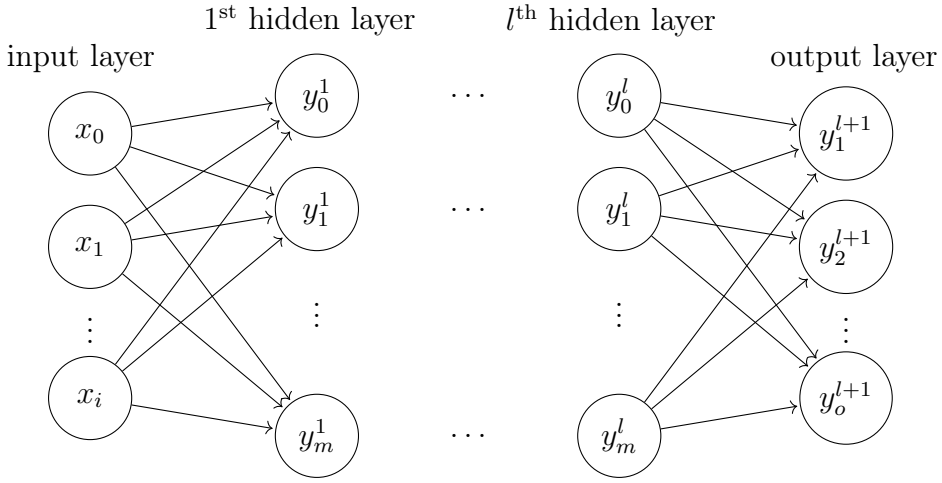


Figure 3.1 Architecture of a FFNN with l hidden layers, i input units, m hidden units in each layer and o output units.

3.4.2 Convolutional neural networks

While FFNNs use matrix multiplication in their layers, CNNs are based on the convolution operation described as:

$$s(t) = (x * w)(t) = \sum_{i=-\infty}^{+\infty} x(i)w(t-i) \tag{3.3}$$

In practice, many implementations of CNNs use a version of the cross-correlation function in which the kernel w is not flipped, but that function is still referred to as a convolution. Using convolution makes CNNs more suitable for processing data with strong local dependencies, such as image and audio signals, because of following reasons: First, length of the convolution can be limited to a small fraction of the input signal in which data exhibits meaningful dependencies. In other words, the connection in CNNs can be made sparse. Second, parameters defined by the convolution operation do not depend on a specific location of features in the input data, and a single set of parameters can learn the patterns that are present in the data regardless of certain variations, such as a uniform shift in the position of features [43].

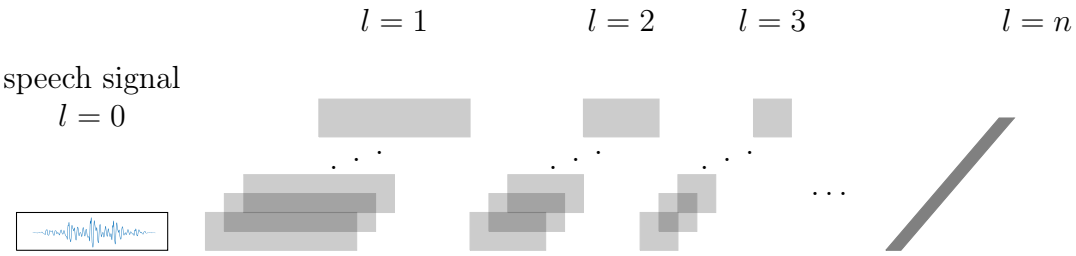


Figure 3.2 An example of convolutional neural networks for processing one-dimensional inputs such as audio signals. Each layer includes the convolution operation and a non-linearity. Vectors of features are shown as gray boxes. The second and third layer in this example include a pooling operation such as subsampling which reduces the length of feature vectors. The temporal resolution of the input is reduced in these layers until eventually it is reduced to a one-dimensional feature space at the n^{th} layer.

Fig. 3.2 illustrates the operation of CNNs. Similar to FFNNs, CNN layers also include a non-linear activation which is sometimes called a detector. Another operation that almost all CNNs use is the pooling function. Pooling replaces some adjacent outputs of a layer with a summary statistic. Common examples of pooling functions are subsampling and selection of maximal output from a rectangular region (also called max pooling) . As a result of the pooling function, learned representations become robust to small input translations. Other parameters of convolution operation such as padding, strides

and kernel size are also useful for customizing CNNs for different applications.Padding describes the number of features (which their value is usually zero) added to the extremities of the input of convolution to control the length of output of convolution. Stride describes the step size by which the convolution filter slides over an input. Finally, kernel size defines the length of the convolution filter.

3.4.3 Recurrent neural networks

Recurrent neural networks (RNNs) are a family of neural networks architecture specifically designed for processing sequential data, such as audio waveforms and text sentences. Similar to CNNs, RNNs are based on the idea of sharing parameters for learning patterns regardless of where in the sequence they occur. While each layer of a FFNN models a function f_i as an element in a chain to model a desired function $F(x)$, in RNNs, a single function f is used in a recursive manner to process the sequence x described by:

$$\begin{aligned} h^i &= f(h^{i-1}, x^i; \theta) \quad 1 < i < l \\ F(x) &= h^i \end{aligned} \tag{3.4}$$

where x^i is the i^{th} element in the sequence x and θ is the parameters of the function f. Fig. 3.3 illustrates the RNN architecture and its recursive process that can be unfolded over time. While originally the parameters θ in RNNs were arranged in the form of an affine transformation similar to a FFNN, more elaborate designs introduced gates to the model. The most common examples of RNNs with gates are long short-term memory (LSTM) and gated recurrent unit (GRU), which we discuss next.

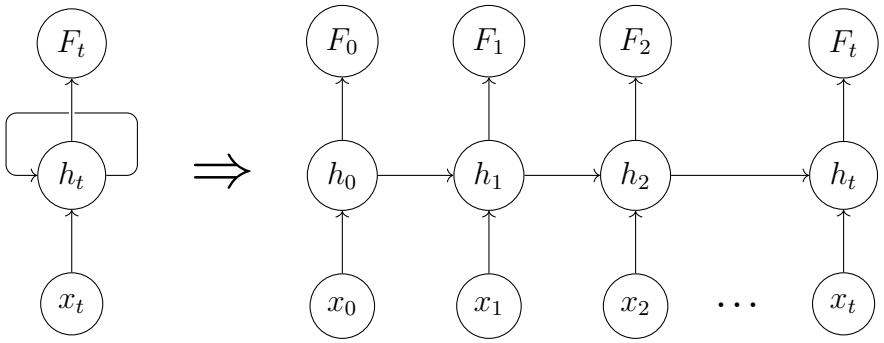


Figure 3.3 (A)Architecture of a RNN. x_t is the input, F_t is the output and h_t is the hidden state. (b) Architecture of a RNN unfolded over time.

LSTM and GRU

One of the main challenges in training RNNs is the vanishing gradients problem caused by long-term dependencies. In other words, when gradient-based optimization methods (see section 3.5.2) are used, the magnitude of gradients turns to a very small amount in long sequences, which reduces the ability of RNN architectures to capture long-term dependencies. Gates are introduced in RNNs architecture such as LSTM [47] and GRU [48] to mitigate this issue. A modern LSTM includes a cell and tree gates. The cell retains values over the time intervals and the gates regulate the flow of information into the cell and out of the cell. The following set of equations describe the process in LSTM:

$$
\begin{aligned}
i_t &= \sigma(W_i.[h_{t-1}, x_t] + b_i) \\
f_t &= \sigma(W_f.[h_{t-1}, x_t] + b_f) \\
o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \\
z_t &= tanh(W_z.[h_{t-1}, x_t] + b_z) \\
c_t &= f_t * c_{t-1} + i_t * z_t \\
h_t &= o_t * tanh(c_t)
\end{aligned}
\tag{3.5}
$$

where h_t is the hidden state vector (also known as the output vector of the LSTM). The input gate i_t and the output gate o_t regulate inward and outward information flow, and the forget gate f_t controls the amount of information which should be retained in the cell. A visualization of this process is provided in Fig. 3.4. GRU is an alternative to LSTM in which the output gate is removed to reduce the number of parameters. However, both have been shown to have a similar performance in many tasks.

3.4.4 Residual neural networks

Residual neural networks (ResNet) are a family of ANNs that utilize skip connections in their architecture. A skip connection is a path that avoids one or several layers and nonlinearities. Fig 3.5 illustrates the architecture of ResNet. In some architectures, such as HighwayNets [50], inspired by LSTM, a desired transfer function is also used to learn skip weights. Skip connections help to mitigate the vanishing gradients problem, and ResNet models with more than 100 layers can be trained successfully. In recent years, ResNet blocks have been used for speech processing especially in the design of speech synthesizers [51, 52, 53, 54].

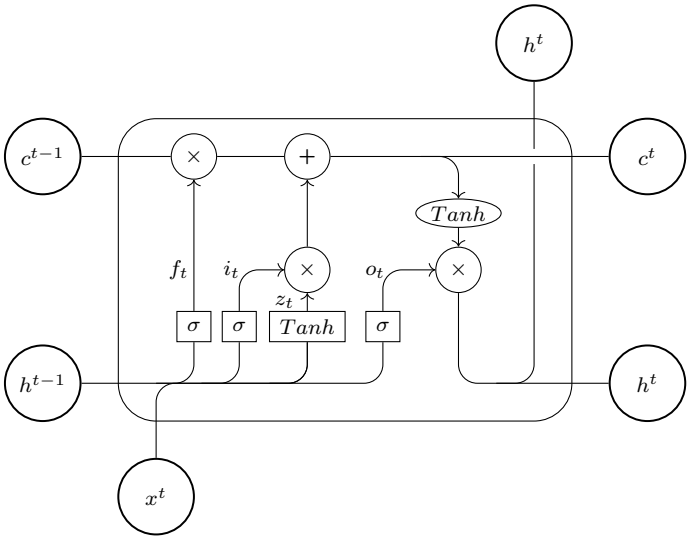


Figure 3.4 Architecture of an LSTM cell. The diagram provides a visual representation of Equations 3.5. Further explanation of architecture of an LSTM cell can be found in [49]

3.5 Training artificial neural networks

In previous section, different architectures of neural networks have been described as non-linear functions which are able to map an input vector x to an output vector y. Training neural networks is the process of learning the correct mapping from the input vector to the output vector. Training includes specification of a criterion and choice of a parameter optimization technique. There are numerous techniques for training neural networks. A general review of these techniques can be found in [55]. In this section, a subset of training techniques relevant to this thesis are presented.

3.5.1 Training criterion

To optimize the parameters θ of a neural network (or to train the network), a criterion (also called a loss, cost or objective function) should be defined. The criterion $\mathcal{L}(\theta, \mathcal{D})$ is a scalar value that defines the fitness of the set of parameters θ optimized on a dataset $\mathcal{D}$ to achieve a certain objective. The mean squared error (MSE) and the cross entropy are examples of training criterion widely used for regression and classification tasks, respectively. There are more elaborate training criterions for complex structures, especially when multiple networks are trained together to perform a single task. Examples of such criterions are given in Section 3.7.

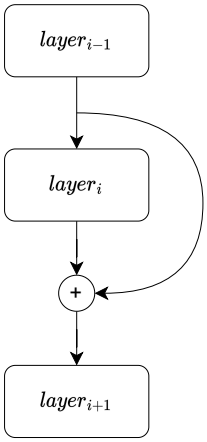


Figure 3.5 Illustration of the Residual neural networks. By adding a skip connection, an extra path that avoids $layer_i$ is added to the network.

Mean squared error

The mean squared error (MSE) is the expected value of the squared error over the dataset as:

$$\mathcal{L}(\theta, \mathcal{D}) = \mathop{\mathbb{E}}_{x \sim p_{data}(x)} \left[(y - \hat{y}_\theta(x))^2 \right] \tag{3.6}$$

where y is the true output and $\hat{y}_\theta$ is the predicted output vector. This criterion is derived from the square of the Euclidean distance; it is always positive and decreases as the error approaches zero.

Cross entropy

The cross entropy between the ground truth and the predicted probability vector is a widely used training criterion in classification tasks. It is defined as:

$$\mathcal{L}(\theta, \mathcal{D}) = - \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [H(y, \hat{y_\theta}(x)] = - \mathop{\mathbb{E}}_{x \sim p_{data}(x)} \left[\sum_j y^j log(\hat{y}^j_\theta(x) \right] \tag{3.7}$$

where j is the number of classes, y is the ground-truth vector and $\hat{y}_\theta$ is the predicted probability vector. In this formulation, minimizing the cross-entropy loss is same as maximizing the likelihood of generated targets with respect to the set of parameters θ.

3.5.2 Parameter optimization

Gradient descent (also known as steepest descent) is a widely used algorithm in training neural networks. Gradient descent iteratively updates the parameters θ in the direction of steepest gradient in order to minimize the criterion $\mathcal{L}(\theta, \mathcal{D})$ according to:

$$\theta^{\tau+1} = \theta^{\tau} - \eta \frac{\partial \mathcal{L}(\theta, \mathcal{D})}{\partial \theta}\bigg|_{\theta=\theta^{\tau}} \tag{3.8}$$

where η is the learning rate, which controls the size of the steps for updating the parameters. Since this simple gradient descent technique is not sufficient for training practical neural networks, several other techniques have been developed to extend the simple gradient for efficient training of multi-layer neural networks on large datasets. Optimization algorithms use other techniques to accelerate gradient descent algorithms. For example, Adam [56] replaces a simple gradient with an exponentially weighted average of the gradients. This technique is shown to converge towards the minimum with a faster pace. Some other notable optimization techniques are stochastic gradient descent, back-propagation, regularization and normalization.

Stochastic gradient descent

Computing an accurate gradient for training a neural network based on a complete dataset (also known as batch-mode gradient descent) needs extremely intense computation for a large dataset. Stochastic gradient descent (SGD) is an alternative approach in which an approximation of the accurate gradient is computed based on a single data point or on a small subsets of the dataset called mini-batches in each step of optimization [55]. While this approximation is less accurate than batch gradient descent, randomly shuffling the dataset and reducing bias in each minibatch can improve the accuracy of generated gradients. To reduce bias in mini-batches of speech datasets, a subset of samples can be selected to feature different speaker, phonetically diverse sentences, diverse recording conditions and environmental conditions. This practice helps train a generalized neural network for speech and audio processing.

Error backpropagation

Error backpropagation (BP) is an efficient algorithm to calculate partial derivatives with respect to the parameters of a neural network for the gradient descent algorithm. In Section 3.4, several ANN architectures have been discussed that, in multi-layer form, can be seen as series of nested functions. Direct computation of the gradient with respect to each weight in the network is not efficient because many operations should be repeated unnecessarily. In the backpropagation algorithm, starting from the output layer, gradients

are computed one layer at a time, and redundant calculation of intermediate terms is avoided. A detailed description of the backpropagation algorithm can be found in [55].

Regularization

Regularization is a technique to improve generalization and avoid over-fitting when training a neural network. In other words, regularization helps the model to perform on unseen data and not only the data that is used for training the model. Regularization is particularly necessary in neural networks which have a large capacity (large number of parameters) relative to the size of the dataset. Large capacity of a model can lead to memorizing the data instead of learning the underlying structure of the data. Parameter norm penalty, dropout and early stopping are examples of regularization methods in training neural networks.

Parameter norm penalty is a regularization term added to the loss function in the form of $\frac{1}{p}\|\theta\|_p^p$ where θ represents the parameters of the network and $\|\theta\|_p$ is the L^p norm. Using the L^p norm penalty forces the parameters to decay towards zero. In other words, using more parameters is penalized in training, and this forces the network to only use the subset of parameters that is sufficient to model the data. Using a low p value, for example using the L^1 norm, results in a sparser solution [55].

Regularization can also be achieved without modification of the training criterion using techniques such as early stopping or dropout. In early stopping, a random subset of the training set (also called validation set) is excluded from training. The validation set is a small percentage of data that should be representative of the whole dataset. During training, the training criterion is also computed for the validation set, and when the validation error starts to increase, this is an indicator of overfitting, therefore the training stops. Alternatively, using the dropout technique [57], some units and their connections are randomly dropped during training (this is also called thinning the network). As a result of dropout, an exponential number of thinned networks are trained during the training phase. Then, the prediction of thinned networks can be averaged by using the unthinned network during the testing phase.

Normalization techniques

Normalization is a set of techniques that aim to decrease the training time for a model. The most common method of normalization is batch normalization where outputs (features) of a layer are normalized based on their mean and variance. It has been shown that, by reparametrization of the optimization problem, batch normalization makes the optimization stable and smooth [58]. Alternatively, the normalization can be performed

separately for every feature in the output (instance normalization), or for features across multiple layers (group normalization).

3.6 Spiking neural networks

Previously discussed neural network architectures are based on highly simplified biological neurons. Biological neurons communicate via timing of a sequence of spikes they generate. These spikes are not modeled in conventional ANNs. Spiking neural networks (SNNs) are a family of neural networks that model the function of biological neurons to a higher degree by processing information using spikes between units rather than using a real value as in ANNs. The information transmitted in SNNs is usually encoded in the frequency of the spikes or in spike timing (also called pulse coding and rate coding, respectively). It has been shown that pulse coding is a more powerful method than rate coding [59] and that, in fact, rate coding is a special case of pulse coding.

Spiking neurons models vary from biologically accurate but computationally complex models such as Hodgkin–Huxley to simple single variable integrate-and-Fire units. Communication between neurons is carried out via synapses. Any neuron that receives spikes (the postsynaptic neuron) from various neurons that emit spikes (presynaptic neurons) experience a postsynaptic potential (PSP) which in turn can cause generation of a spike by the postsynaptic neuron. The strength of the PSP is affected by factors such as the strength of the synaptic connection and the relative arrival time of spikes [60].

3.6.1 Training SNNs

Strategies for training SNNs are more diverse than for ANNs and they can be categorized into bio-plausible unsupervised techniques, the most notable of which being spike-timing-dependent-plasticity, and gradient-based learning techniques that can be used regardless of learning paradigms.

Spike-timing-dependent-plasticity (STDP)

STDP [61] is a widely used technique for unsupervised learning in SNNs. In STDP, synaptic weights are manipulated instantaneously based on the presynaptic and postsynaptic spike timings. Learning with STDP accounts for the history of presynaptic and postsynaptic spikes locally, and it is a bio-plausible, fast and simple method.

Gradient-based learning

In the context of SNNs, one approach for gradient-based learning is to train different types of ANNs using the BP algorithm and then substitute the conventional artificial neurons by spiking neurons [62, 63, 64, 65]. While these approaches generally aim to use the more energy-efficient SNNs for inference, a loss of accuracy is expected as a result of the conversion from ANNs to SNNs. Alternatively, several techniques have been developed to directly train SNNs using the BP algorithm. In [66], a spike-based BP algorithm is introduced that treats the membrane potential as a differentiable activation of the neuron to update the weights for unsupervised layerwise training. In [67], another spike-based BP algorithm is introduced for end-to-end gradient descent optimization. Spike-based BP algorithms are in their early development phase and face some challenges. The current methods are computationally intensive. Moreover, complementary training techniques such as proper methods of initialization of synapse weights and regularization have yet to be developed [68].

3.6.2 Liquid state machine (LSM)

Training SNNs using techniques such as STDP alone is a difficult task and it is likely that state-of-the-art accuracy would not be achieved. In the liquid state machine (LSM) [69], besides a reservoir of SNNs, a simple supervised layer (read out) is also included in the model, which can improve classification results. Generally, neurons in the reservoir are randomly connected and they process the input with a variety of non-linear functions. Following the non-linear process, the read out layer can learn possible linear combinations to infer any desired output. In [70], using Stone-Weierstrass theorem, it's been shown that under a set of simple conditions, LSMs have universal computational power (universal approximation property in neural networks). LSM is the main processing block of the classification task presented in Chapter 5.

3.7 Generative models

Generative modeling is one of the guiding principles of machine learning. In a general definition, generative models capture the joint probability of input x and output y as $p(x, y)$. As a result, they are able to produce samples of data x, in contrast to discriminative models which capture $p(y|x)$ and only produce y when samples of data x are given. This property of generative models is particularly interesting for speech coding, which deals with speech synthesis in the decoder. Boltzman machines and deep belief networks are some of the most known examples of generative networks, but there are many other kinds of generative models. This section focuses on the approaches which have been shown to

be very successful in the field of speech processing and specially speech coding. These includes autoencoders [43], flow-based deep generative models [71] and generative adversarial networks [14].

3.7.1 Autoencoders

Autoencoders are a family of neural networks that are trained to generate a copy of the input in their output. The diagram of an autoencoder is illustrated in Fig. 3.6. A prominent property of autoencoders is that they contain an information bottleneck. This bottleneck captures a low-dimensional data representation that can be useful for data compression. Because of the information bottleneck, the fist part of the network is called encoder and the second part is called decoder. Autoencoders are trained in an unsupervised manner since no label information is needed to train the network, and the network simply learns $g(f(x)) = x$ in which f and g are the encoder and decoder functions, respectively.

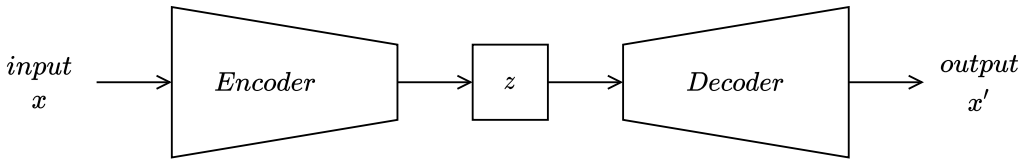


Figure 3.6 Illustration of the autoencoder architecture. The network is trained to generate a copy of the input in its output. There is an information bottleneck in the design of the architecture to capture a representation of signal in lower dimensions.

Variational autoencoders

In modern formulations of autoencoders, the function f and g are generalized to capture a stochastic mapping $p(z|x)$ and $p(x|z)$, respectively. Variational autoencoder (VAE) [72] is one of these formulations in which the input is mapped to a distribution instead of a fixed vector and the loss function is formulated as the variational lower bound:

$$\begin{aligned} &\mathcal{L}_{VAE}(\phi,\theta) = -log(p_\theta(x)) + D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x))) \\ &= \mathop{\mathbb{E}}_{z\sim q_\phi(x|z)} log(p_\theta(x|z)) + D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) \end{aligned} \tag{3.9}$$

where D_{KL} is the Kullback-Leibler divergence, and functions q_ϕ and p_θ are the encoder and decoder modeled by ϕ and θ parameters, respectively. One difficulty in this formulation is the fact that calculating the loss function requires sampling $z \sim q_\phi(x|z)$, which can be solved by a reparameterization trick [72].

While the Gaussian distribution is a popular choice as a continuous prior in VAEs, learning k-dimensional discrete representations with models such as vector-quantized VAE (VQ-VAE) [17] is more convenient for a data compression application because the network

already learns quantized representations. Representations extracted by VQ-VAE are used in the architecture of machine learning-based speech codecs [73, 74]. In these models, VQ-VAE is proven to be very efficient for low bit-rate speech compression.

3.7.2 Generative adversarial networks

Inspired by game theory, the generative adversarial network (GAN) is designed to include two neural networks, a generator and a discriminator that compete in a minimax game. Fig. 3.7 illustrates the diagram of GAN. The training criterion of GAN is described by:

$$\underset{G}{Min}\underset{D}{Max}\mathcal{L}(D,G) = \underset{x\sim p_{data}(x)}{\mathbb{E}}[log(D(x))] + \underset{z\sim p_z(z)}{\mathbb{E}}[log(1-D(G(z)))] \tag{3.10}$$

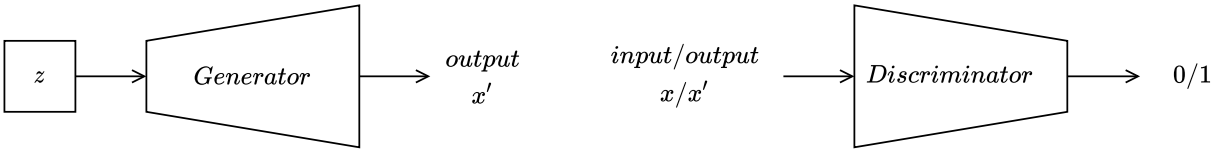


Figure 3.7 Illustration of the GAN architecture. The discriminator produces binary class labels corresponding to real/synthetic data. These labels are used for generating gradient for training the generator.

where the accuracy of discriminator D is maximized to recognize real data by the term $\mathbb{E}_{x\sim p_{data}(x)}[log(D(x))]$ and is also maximized for discriminating synthethic data from real data by the term $\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$. On the other hand, generator G is trained to compete with the discriminator by minimizing the term $\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$ simultaneously to produce realistic samples. In other words, this loss function quantifies the similarity between the generated data distribution and the real data distribution. The trained generator network has been shown to produce realistic samples for many real world applications with rich contents such as audio and image. GAN is one of the most studied generative networks, and over the years, many variant of GAN have been introduced such as WGAN [75], CycleGAN [76], StyleGAN [77], all showing improvements over the original GAN architecture.

GAN-base architectures, despite having a great potential to produce realistic data samples, face some challenges in training. For example, reaching equilibrium between generator and discriminator can be difficult: the generator can collapse to produce only a subset of outputs (mode collapse), or if the discriminator becomes perfect during training, the loss function becomes zero and does not produce any gradient for training. To mitigate these training problems, several solutions have been proposed. One strategy is to use feature matching by designing a discriminator to compare the statistics of the output of

the generator to those of real samples. Also, adding noise to the inputs of the discriminator can prevent the problem of vanishing gradients. Depending on the nature of the problem, there are a variety of alternative solutions to improve the training of GANs and solutions are not limited to these examples [78, 79].

GANs are exceptionally good for synthesizing realistic speech signals. They generally can produce higher speech quality than standard codecs currently in use. Compared to early deep learning-based approaches such as Wavenet [15], GANS are computationally less complex and much faster at synthesizing speech signals [52, 53, 54]. In Chapter 8, the architecture of the decoder in the proposed speech codec has a GAN structure.

3.7.3 Flow-based models

Learning the probability density function of data $p(x)$ with a generative model is a challenging task since exploring all of the possible values of the latent variable in $p(x) = \int p(x|z)p(z)dz$ can be difficult. However, unlike the two previous generative models discussed in this section, flow-based models use a normalizing flow to model the exact data distribution [71]. A normalizing flow is a series of invertible functions f_i which are used to map a latent variable z with a typically simple tractable density function such as the multivariate Gaussian distribution to the density function of the data as $x = f_k \circ f_{k-1} \circ ... f_1(z)$. Since functions f_i are invertible, the relationship between density functions can be expressed as $z = f_1^{-1} \circ f_2^{-1} \circ ... f_k^{-1}(x)$ to train the network. The training criterion of flow-based model is simply the negative log-likelihood over the training dataset $\mathcal{D}$:

$$\mathcal{L}(\mathcal{D}) = - \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [log\,(p_\theta(x))] \tag{3.11}$$

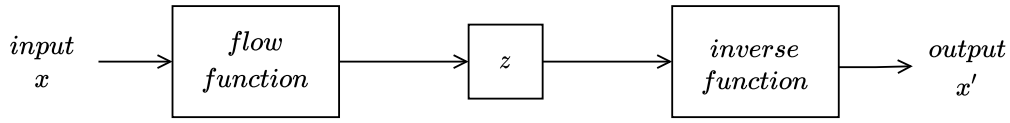


Figure 3.8 Illustration of flow-based models. These models are composed of a series of invertible transformations that maps a latent variable z to the density function of the data.

Different invertible transformations have been proposed for flow-based models to model the data probability density function [80, 81, 82]. In Glow [82], invertible 1x1 convolutions are introduced for generative flow, and in Waveglow [83], it is applied for speech synthesis by conditioning the distribution of audio samples on mel-spectrogram.

3.8 Summary

In this chapter, the knowledge about machine learning techniques that is relevant to understanding the models proposed later in this thesis has been presented. An introduction to machine learning tasks and learning paradigms helps understand the reasoning behind the tasks performed in this thesis. Since Chapters 5-8 are written in a scientific paper format, they do not contain all the necessary basic knowledge about the architecture and training of neural networks. In Sections 3.4-3.6, this knowledge is provided to facilitate understanding of the machine learning techniques used in the experiments. Generative models discussed in Section 3.7 introduce common techniques from the field of machine-learning that are currently used for developing similar approaches to this project. In all of the sections, contents are presented with a special focus on speech processing and the relevant references are given. In the following chapter, the content that has been presented in this chapter will be used to provide a coherent overview of the experiments performed in this thesis.

CHAPTER 4

Overview of the experiments

4.1 Introduction

In this chapter, a coherent view of the experiments performed in this thesis is provided. An overview of these experiments is provided in table 4.1. Each section in this chapter corresponds to one of the tasks performed in this thesis. A summary of the experimental results from published papers is presented with further insights and links between the development steps.

Table 4.1 Overview of experiments in this thesis.

Classification		
Steps	Training criterion	Components
• Learn unsupervised features x' • A supervised model of $p(C\|x')$ to predict C emotion classes	• STDP	• ERB filters • SNN reservoir • PCA • LDA
Regression		
Steps	Training criterion	Components
• A supervised model of $p(y\|x)$ to predict speech sample y	• MSE	• LSTM layers
Representation learning (encoder)		
Steps	Training criterion	Components
• An unsupervised model of $p(C_1, C_2\|x)$ to learn representations C_1 and C_2 • Linear classification of multiple attributes • Feature quantization	• Contrastive loss	• CNNs • GRUs
Speech synthesis (decoder)		
Steps	Training criterion	Components
• An unsupervised model of $p(x\|\hat{C}_1, \hat{C}_2)$ to synthesize speech signal from quantized representations $\hat{C}_1$ and $\hat{C}_2$	• Adversarial loss • Feature loss • Mel-spectrum loss • CC loss	• CNNs • ResBlocks

4.2 Classification

Fig. 4.1 illustrates a representation of the proposed classification model. The primary objective when designing a classifier in this project is to extract features of speech that preserve long-term speech attributes and that can eventually be used for coding. As a result, the bulk of the design focuses on extracting a set of unsupervised features x' from an input signal x. An intentionally simple classifier is used to test if long-term attributes C are preserved and if they can be identified easily. For simplicity, we focus on classifying emotions, one of the most complex long-term attributes of speech.

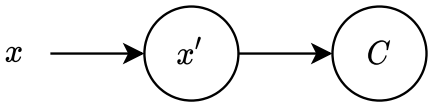


Figure 4.1 A graphical representation of the proposed classification model. A simple classifier recognizes the emotions from unsupervised biologically-inspired features x'.

To extract unsupervised features, we use the structure of the human auditory system as our inspiration. The biologically inspired design includes a model of the human auditory periphery, a processing stage based on SNNs, and the motor theory of speech perception [84]. We extract unsupervised features in two stages. The preprocessing stage, which should be able to preserve all of necessary information in a simpler form than the raw signal itself, and the processing stage, which should be able to handle the variety of information that is present in speech signals. The preprocessing stage is influenced by the human auditory periphery, where spectral information is extracted using the equivalent rectangular bandwidth (ERB) scale. The processing stage is based on SNNs and extracts features from speech signals with the STDP learning rule, a biologically plausible unsupervised learning method. The motor theory of speech perception influences both stages. Using linear prediction (LP), we decouple the source and vocal tract components of the speech signal, and we provide them separately to two SNN reservoirs for better access to vocal tract gestures in the model. Subsequently, the extracted features are classified with a simple classifier constructed with two stages, the first consisting in a principal component analysis (PCA) to reduce the dimensionality and the second being a linear discriminant analysis (LDA) to find a linear combination of features that characterize emotions. The results achieved by the proposed classifier were comparable with the state-of-the-art classifiers. The emotion classification task introduces a method for extracting biologically-inspired features, and demonstrates the possibility of using such features for classification of long-term speech attributes as a part of a speech codec.

While certain elements of this design were highly influential in the design of the following tasks, some other elements that were not compatible with the following tasks were abandoned. Unsupervised learning of representations with a biologically inspired design is the guiding principle of designing a fully machine learning-based encoder in the third task (Section 4.4). In that third task, we expand the notion of biologically-inspired design by including theories of cognition. Linear prediction and linear classification also play an important role in the design of the encoder. However, at the time of conception of the proposed classifier, SNNs had several important limitations. Modeling a large number of spiking neurons for construction of an entire codec was computationally expensive and did not seem to be feasible. Moreover, SNN training algorithms did not seem to be able to produce accurate speech signals. As a result, the following tasks are all performed using conventional ANNs. Besides, in all of the following tasks, preprocessing stages are abandoned, and building end-to-end designs with less manual engineering of features is prefered.

4.3 Regression

Unlike classification, prediction deals with a more complicated objective of producing a waveform. In its simplest form, the goal is to produce a speech signal with maximum prediction gain, and in its most complex form the goal is to synthesize speech signals with higher subjective quality.

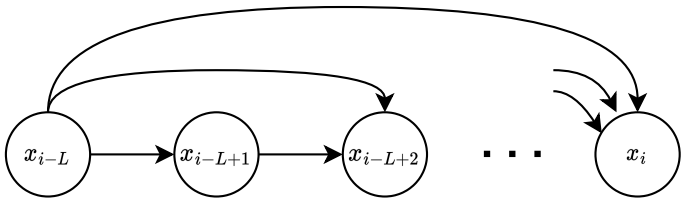


Figure 4.2 A graphical representation of the proposed predictor model. The network produces speech waveform in an autoregressive manner. The connections represent the relationship between samples that is modeled by the network.

Fig. 4.2 illustrates a representation of the proposed predictor model for the regression task. The objective is to train a small RNN for a few number of passes on the recent samples of speech (a 10-25 ms window depending on the configuration), in an online manner, to keep the network updated for predicting the following samples. Although the network has a very short receptive field of L samples (5 ms), to limit the size of the network and its complexity, online training extends the memory of the network to a longer window (hundreds of milliseconds). The extension of the network's memory varies depending on the network configuration. Fig. 4.3 demonstrates the extension of the memory of

the network as a result of our training approach. Longer memory helps to exploit long-term correlations in the signal and to make a better prediction of future samples. We apply the proposed predictor to the packet loss concealment (PLC) problem. If one or multiple consecutive speech frames are lost, the network produces a speech waveform in an autoregressive manner to replace the lost frames. We showed that the proposed method outperforms the standard ITU G.711 Appendix I PLC technique in terms of PESQ scores.

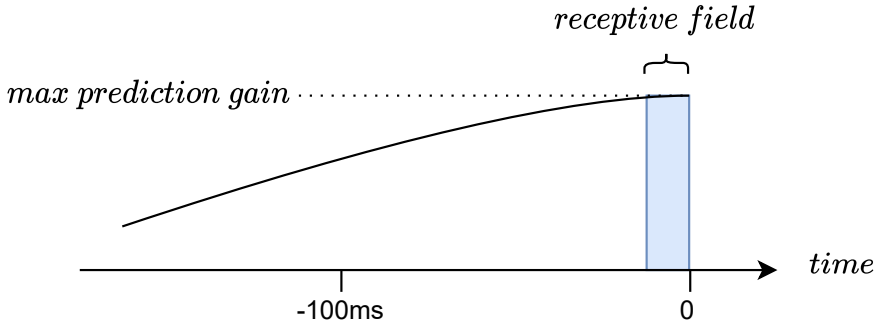


Figure 4.3 The proposed predictor has a memory of the input signal. This memory extends to hundreds of milliseconds with online training to capture long-term correlations.

While using a small RNN, trained in an online manner, can predict speech samples to produce high-quality speech up to several frames, the design needs modifications to produce features that can encode speech signals. In the next task, we use RNNs for prediction in a latent space, instead of prediction in sample space, as a method for representation learning.

4.4 Representations learning (encoder)

For the representation learning task, we design a network based on several theories of cognition, as an encoder. First, we extract hierarchical representations in two levels of abstraction. The lower and upper stages process information from short and long frames of speech, respectively. Secondly, a top-down pathway between the abstraction levels is introduced to improve the quality of the representations. Finally, we use predictive coding as the learning strategy.

Fig. 4.4 illustrates a representation of the encoder. The network is composed of multiple subnetworks. We extract latent variables z_s and z_l with two subnetworks based on CNNs. The successive subnetworks in the neural network represent increasing levels of abstraction. Irrelevant variations are reduced with CNNS, starting from smaller building blocks of speech signals to the bigger building blocks, to produce the latent variables. We use

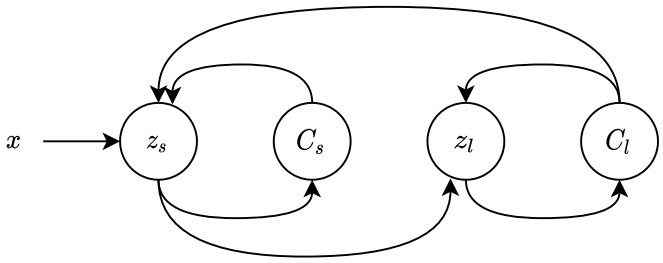


Figure 4.4 A graphical representation of the proposed encoder. Speech representations (C_s and C_l) are extracted by prediction in latent space (z_s and z_l).

autoregressive recurrent layers based on GRUs to derive contextual representations C_s and C_l. The extracted representations predict the future latent variables z_s and z_l with a linear function. The linear prediction of latent variables plays an important role in regularizing the model as well as producing representations that are interpretable with a linear classifier. A top-down pathway also decorrelates long-term attributes of speech from short-term representations. The combination of these elements shapes a network to carry out predictive coding in latent space.

We measure the performance of the proposed approach in terms of classification accuracy for several short-term and long-term speech attributes including speaker identity, emotions and phonemes. The results outperform the state-of-the-art approaches. The extracted representations are also extremely robust to quantization. With this task, we conclude the design of the encoder. In the following task, we propose a decoder to resynthesize high-quality speech from the quantized speech representations.

4.5 Speech synthesis (decoder)

Fig. 4.5 illustrates a representation of the proposed decoder. A two stage GAN-based structure is used to resynthesize natural-sounding speech signals x from the quantized representations $\hat{C}_s$ and $\hat{C}_l$ extracted by the encoder. Since training a GAN structure is a difficult task, multiple objectives are used to construct a proper criterion for training the network. First, an adversarial loss is used to minimize the difference between real samples and resynthesized samples in general. This measure is a good approximation of subjective measures, since the discriminator network minimizes any detectable difference between the resynthesized speech sample and the real speech sample. Secondly, two cognitive coding (CC) distances are used to minimize the difference between the signals, by minimizing the difference between long-term and short-term representations extracted from signals. Thirdly, the mel-spectrum distance is used to minimize the spectral difference between the

two signals. Finally, a feature-matching distance is used to further enhance the ability of the discriminator network in distinguishing real samples from resynthesized ones.

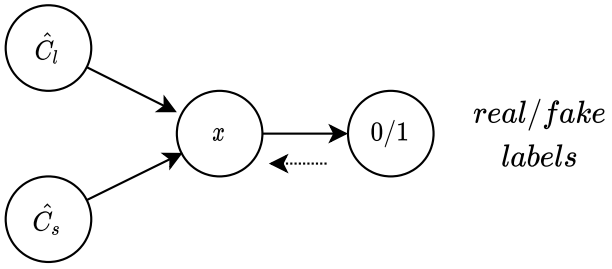


Figure 4.5 A graphical representation of the proposed decoder. Speech signal is resynthesized from the quantized representations with adversarial training.

We evaluate the subjective quality of the proposed approach to resynthesize speech signals with an AB test. The proposed method outperforms the standard AMR-WB codec in terms of subjective quality, delay and bitrate. With the proposed models in the third and fourth tasks, this conclude the design of a fully machine learning-based codec.

4.6 Summary

In this chapter, an overview of all of the tasks performed in this thesis, the links between the development steps, and further insights are provided. With the performed tasks, we provided new solutions for adressing classical challenges (classification, prediction) in conventional codecs as well as developing a fully machine learning-based speech codec. Our results show that machine learning tools are efficient for unfolding complicated patterns of speech signal, which current coding systems normally are struggling to exploit. In the following Chapters 5-8, published papers for the tasks performed in this thesis are reproduced.

CHAPTER 5

Biologically inspired speech emotion recognition

Preface

Authors and affiliation:

Reza Lotfidereshgi: PhD student, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Philippe Gournay: professor, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Status of paper: final version published

Acceptance date: 19 June 2017

Publication venue: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)

Contribution: A classifier designed to extract features while preserving emotions, one of the most complex long-term speech attributes, and then classifying them with a simple classifier as the output layer.

French title: Reconnaissance des émotions de la parole d'inspiration biologique

French abstract:

Les méthodes de classification conventionnelles basées sur des caractéristiques extraites du signal ne s'appliquent pas bien à la reconnaissance automatique des émotions de la parole, principalement parce que l'ensemble précis de caractéristiques spectrales et prosodiques nécessaires pour identifier l'état émotionnel d'un locuteur n'a pas encore été déterminé. Cet article présente une méthode qui opère directement sur le signal de parole, évitant ainsi l'étape problématique de l'extraction de caractéristiques. En outre, cette méthode combine les points forts du modèle classique source-filtre de la production de la parole humaine avec ceux de la machine à état liquide (LSM) récemment introduite, qui est un réseau de neurones à décharges d'inspiration biologique (SNN). Les composants de source et de conduit vocal du signal de parole sont d'abord séparés et convertis en représentations spectrales perceptuellement pertinentes. Ces représentations sont ensuite traitées

séparément par deux réservoirs de neurones. La sortie de chaque réservoir est réduite en dimensionnalité et envoyée à un classificateur final. Il a été démontré que cette méthode offre de très bonnes performances de classification sur la base de données de Berlin sur le discours émotionnel (Emo-DB). Cela semble un cadre très prometteur pour résoudre efficacement de nombreux autres problèmes de traitement de la parole.

Biologically inspired speech emotion recognition

Reza Lotfidereshgi, Philippe Gournay

5.1 Abstract

Conventional feature-based classification methods do not apply well to automatic recognition of speech emotions, mostly because the precise set of spectral and prosodic features that is required to identify the emotional state of a speaker has not been determined yet. This paper presents a method that operates directly on the speech signal, thus avoiding the problematic step of feature extraction. Furthermore, this method combines the strengths of the classical source-filter model of human speech production with those of the recently introduced liquid state machine (LSM), a biologically-inspired spiking neural network (SNN). The source and vocal tract components of the speech signal are first separated and converted into perceptually relevant spectral representations. These representations are then processed separately by two reservoirs of neurons. The output of each reservoir is reduced in dimensionality and fed to a final classifier. This method is shown to provide very good classification performance on the Berlin Database of Emotional Speech (Emo-DB). This seems a very promising framework for solving efficiently many other problems in speech processing.

5.2 Introduction

Speech is a fundamental means of communicating not only words, but also a vast range of human emotions. Consequently, speech processing applications, such as human-machine interfacing and speech recognition, could benefit from the introduction of a reliable method for automatic recognition of human emotions through speech.

Conventional speech emotion recognition methods consist of a feature extraction step followed by a classifier. Various spectral and prosodic features can be used [85]. Finding the "best" set of features, namely, one that is both complete and compact, is a critical step which has a considerable impact on the performance of the system. State of the art conventional methods mostly differ in their choice of features and of classifier type [86, 87].

In recent years, there has been an increasing trend toward developing speech processing methods that operate directly on the speech signal in order to avoid the problematic feature extraction step. For example, Convolutional Neural Networks (CNNs) [88] and Deep Neural Networks (DNNs) [89], have been successfully used for recognizing emotions directly from raw temporal or spectral data.

The Liquid State Machine (LSM) is another recently proposed method that operates directly on raw data. The LSM relies on a network of spiking neurons that are much closer to biological neurons than the rate-based model used in CNNs and DNNs. Despite its theoretical appeal, the LSM is slow in finding practical applications. The main problem when implementing an LSM is to create a specific reservoir design that is best adapted to the task at hand [90]. In this paper, this problem is solved by introducing prior knowledge about the human speech production system into the LSM. Without any loss in terms of information, the speech signal is divided into two components: the source and the vocal tract. Individually, each component is easier to process by a reservoir of spiking neurons. Furthermore, the inclusion of a production model in the recognition system is justified by the motor theory of speech perception, that states that people perceive speech by identifying the vocal tract gestures that produced it [84].

The outline of the paper is as follows. The source-filter model for human speech production and the principles underlying the liquid state machine are reviewed in section 5.3. The proposed biologically inspired method is presented in section 5.4. Some experimental results are given and discussed in section 5.5, and conclusions are drawn in section 5.6.

5.3 Relation to prior work

5.3.1 The source-filter speech production model

According to the source-filter model of human speech production, a speech signal is produced by passing a source of air pressure through an acoustic filter [91]. The source is a combination of a noise-like turbulent excitation produced by constrictions along the vocal tract (for unvoiced speech) and a quasi-periodic excitation produced by vibrating vocal folds (for voiced speech). The filter represents the variable response of the vocal tract. In practice, the most commonly used method to separate the contributions of the source and the filter is the Linear Predictive (LP) analysis. The LP analysis is a frame-based process which results in: (1) a set of LP coefficients which represent the filter for the frame; and (2) a residual error signal which represents the source. Equation 5.1 shows the calculation of a predicted speech sample $\widetilde{x}(n)$ from past speech samples $x(n-i)$ and the calculation of a residual sample $e(n)$. The Levinson-Durbin algorithm is usually used to find the a_i

coefficients that minimize the quadratic error E as shown in equation 5.2.

$$\widetilde{x}(n) = \sum_{i=1}^{M} a_i x(n-i), \quad e(n) = x(n) - \widetilde{x}(n) \tag{5.1}$$

$$E = \sum_{n} e(n)^2 \tag{5.2}$$

The LPC analysis has long proven to be a very efficient tool in speech processing and is now used for example in every speech coder.

5.3.2 The liquid state machine

A reservoir computing system consists of a Recurrent Neural Network (RNN) followed by an output layer of neurons that performs the final recognition/classification task [90]. In a reservoir computing system, the RNN is randomly created and does not need to be trained using supervised methods such as the gradient descent. The output layer, in contrast, is trained using a supervised method. Reservoir computing is successful for complex nonlinear classification tasks for two reasons. First, because training an RNN using a gradient-descent algorithm would be time consuming and prone to convergence issues. Secondly, because reservoir computing has been shown to outperform most other nonlinear identification, prediction and classification methods on various problems.

The Liquid State Machine (LSM) is a special type of reservoir computing method where the reservoir is a Spiking Neural Network (SNN) [69]. SNNs use temporal coding and therefore process information in very much the same way as a biological neural structure does. Fig.5.1 shows a typical LSM structure. First, the LSM uses a function L^M to map the input $u(t)$ to the "liquid state" $x(t)$, where $x(t)$ is an arbitrary nonlinear function of the input $u(t)$ and of the past input values. Secondly, a memoryless function f^M maps $x(t)$ to the output $y(t)$. This "readout function" is trained for the task to accomplish. The SNN performs a nonlinear mapping from the input space to the high dimensional "liquid state" space. As a result of this projection, the separation of different classes by the readout function is much easier. Several methods including Support Vector Machine (SVM), Multi Layer Perceptron (MLP) and ridge regression have been tried as reservoir readouts [90].

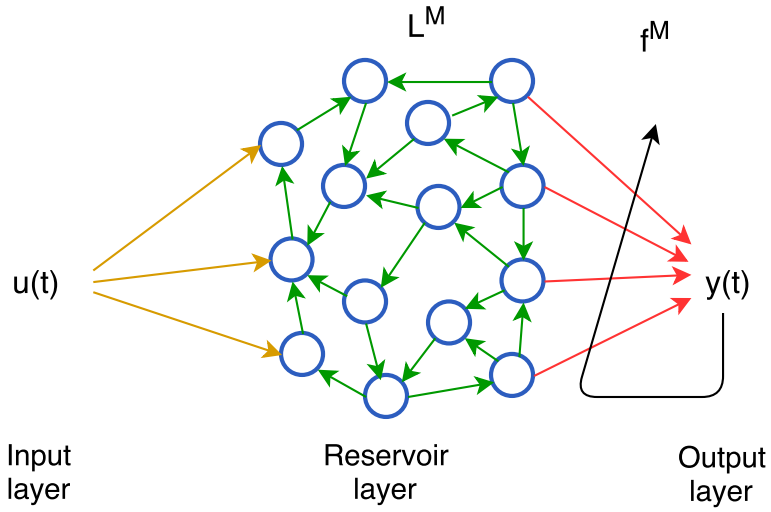


Figure 5.1 A typical LSM structure. Only the output layer is trained using a supervised methods.

5.4 Proposed method

Fig.5.2 shows the flowchart of the proposed speech emotion recognition method. The recognition process is divided into two steps: the preprocessing and the liquid state machine.

In the preprocessing step, the input speech signal is divided into two orthogonal and complementary components that are transformed and perceptually shaped according to the properties of the human cochlea. Specifically, an LP analysis is performed on a frame base. The prediction residual is calculated according to equation 5.1 and decomposed using a 77-channel gammatone filterbank with ERB scaling. This constitutes the input of the first reservoir. In parallel, the frequency response of each all-pole LP filter is computed to reveal the formant structure of the speech signal. This frequency response is also shaped using the exact same ERB scaling and constitutes the input of the second reservoir.

As in the lower auditory nuclei, even auditory cortex has the tonotopic structure [11]. Such structure suggests that closer frequency channels are processed by closer groups of neurons. In the design of the reservoirs, the neurons are therefore arranged in 3D structures, each reservoir containing 77 layers of 3*3 neurons. Each layer of neurons is excited by only one of the 77 input channels, in order of increasing frequency. Connections between closer neurons are favored, with a probability of connection between neuron $n1$ and $n2$ that depends on the distance $D(n1, n2)$ according to equation 5.3. Parameters C and λ are responsible for controlling the reach and density of the connections, and are set

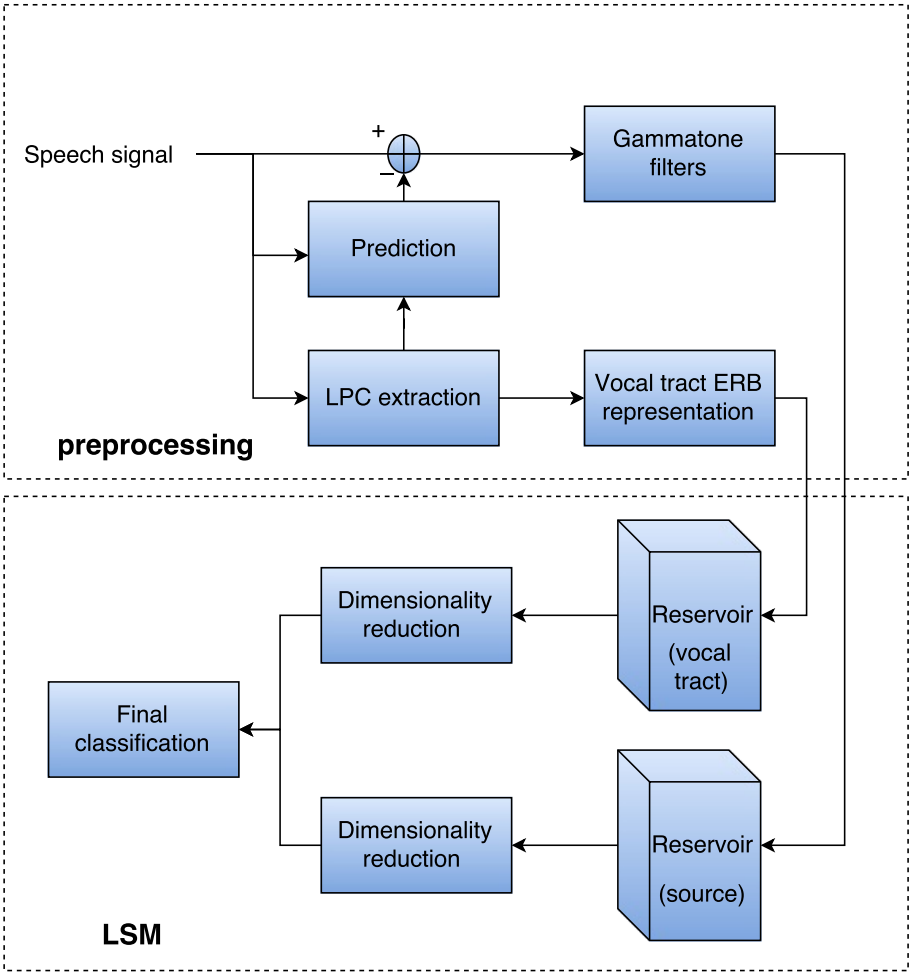


Figure 5.2 Flowchart of proposed emotion recognition method.

respectively to 1 and 3.4. These values were determined after some experiments and could probably be further optimized.

$$P(n1, n2) = Ce^{-\frac{D^2(n1,n2))}{\lambda^2}} \tag{5.3}$$

A standard implementation of the integrate-and-fire neuron by Troyer is used [92]. The Asymmetric Spike Time-Dependent Plasticity (STDP) is then used as the learning rule to adapt the conductance of the synapses throughout the speech sample. This learning rule is known to result in stable networks that are very effective at extracting the correlations present in the input [93]. The exact learning rule is given in equation 5.4.

$$f(\Delta) = \begin{cases} A_+ e^{\frac{\Delta}{\tau_+}}, \Delta < 0 \\ -A_- e^{-\frac{\Delta}{\tau_-}}, \Delta > 0 \end{cases} \tag{5.4}$$

More details about this learning rule can be found in [93]. Δ is the time difference between pre- and post-synaptic spikes. A_+ and A_- are maximum amount of synaptic modifications. Two key parameters to be set are the time constants τ_+ and τ_- because they condition the memory of the reservoir. Following [93], τ_+ was set to 20 ms and τ_- was tuned to maximize performance (see section 4). The simulation of neural activity is done using the Brian2 simulator [94] [1].

To reduce dimensionality, Principal Component Analysis (PCA) is applied to the average activity of the neurons from each reservoir. Compared to the widely used ridge regression, PCA presents the advantage of being able to shrink the output of the two reservoirs separately. The outputs of the two PCAs are simply combined. For final recognition, Linear Discriminant Analysis (LDA) is used.

5.5 Experiments

5.5.1 Berlin database of emotional speech

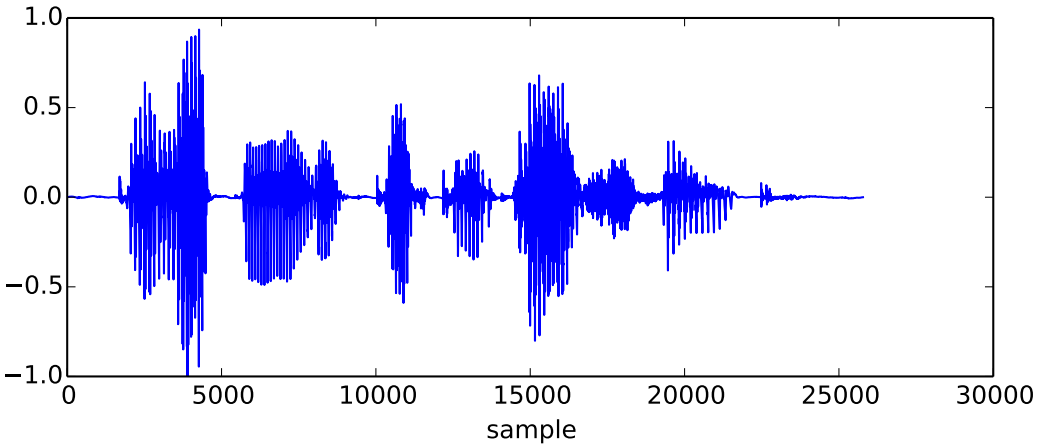
The proposed method was tested on the Berlin database of emotional speech (Emo-DB, [96]). This is a well recorded and now widely used emotional speech database. It is easily accessible and well documented. It contains 535 utterances produced by ten professional actors pronouncing ten different texts and covers seven different emotions.

5.5.2 Preprocessing

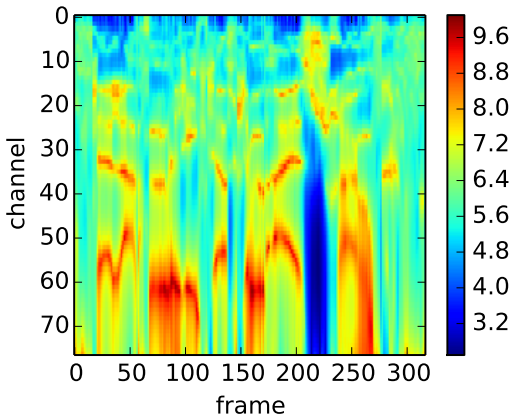
The preprocessing step first consists in an LP analysis of the input speech signal. The autocorrelation method is used to estimate LP filters of order 16. A 30 ms Hamming window is used so that the formant structure is adequately captured. The LP coefficients are updated every 5 ms in order to closely track the changes in the vocal tract. The source and vocal tract components of the speech signal are then separated. First, the LP residual is computed and fed to a 77-channel gammatone filterbank. For each channel of the filterbank, the energy of 5 ms segments is computed and a logarithm is applied to reduce the dynamic range of this representation of the source component. Secondly, the frequency response of each LP filters is computed and shaped using an ERB frequency scaling. A logarithm is also applied to reduce the dynamic range of this representation.

An example of emotional speech signal is represented in Fig.5.3(a). The corresponding source and vocal tract representations are presented in Fig.5.3(b) and Fig.5.3(c), respec-

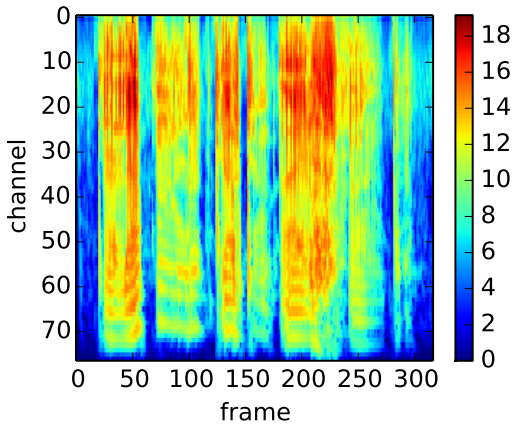
1. Regarding computational complexity, the proposed method can process speech signals faster than real time. The main source of computational complexity is simulating SNNs reservoirs. An estimation of complexity of simulating SNNs by Brian 2 can be found in [95].



(a) sample speech signal



(b) vocal tract contribution



(c) source contribution

Figure 5.3 Preprocessing of speech signal. The scales of the spectro-temporal representations are in dB.

tively. These two spectro-temporal representations of the speech signal are used as inputs for two reservoirs of spiking neurons.

5.5.3 LSM tuning

Fig.5.4 shows the recognition rate of the proposed method for different numbers of principal components for each of the two reservoirs. The reservoir for the vocal tract component was tuned for $\frac{\tau_-}{\tau_+} = 5$ and the reservoir for the source component was tuned for $\frac{\tau_-}{\tau_+} = 3$. The results are obtained using 50-fold cross validation where 90% of the database is used for training and 10% for testing. Results below 60% of recognition rate are not shown. The vertical and horizontal axes are the number of principal component selected from the vocal tract and source reservoirs, respectively. The borders of the figure shows the performance when only one reservoir is used (no component from the other reservoir is

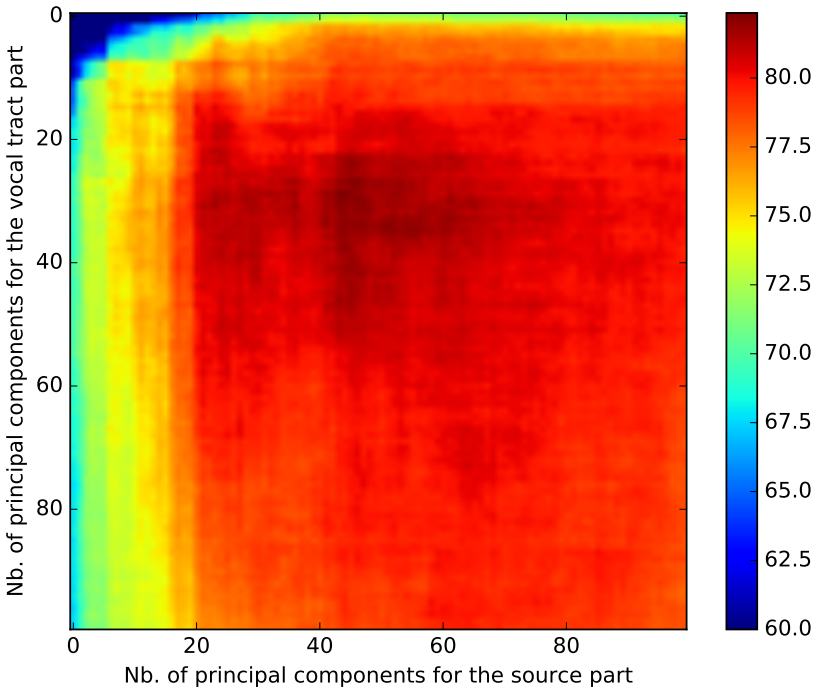


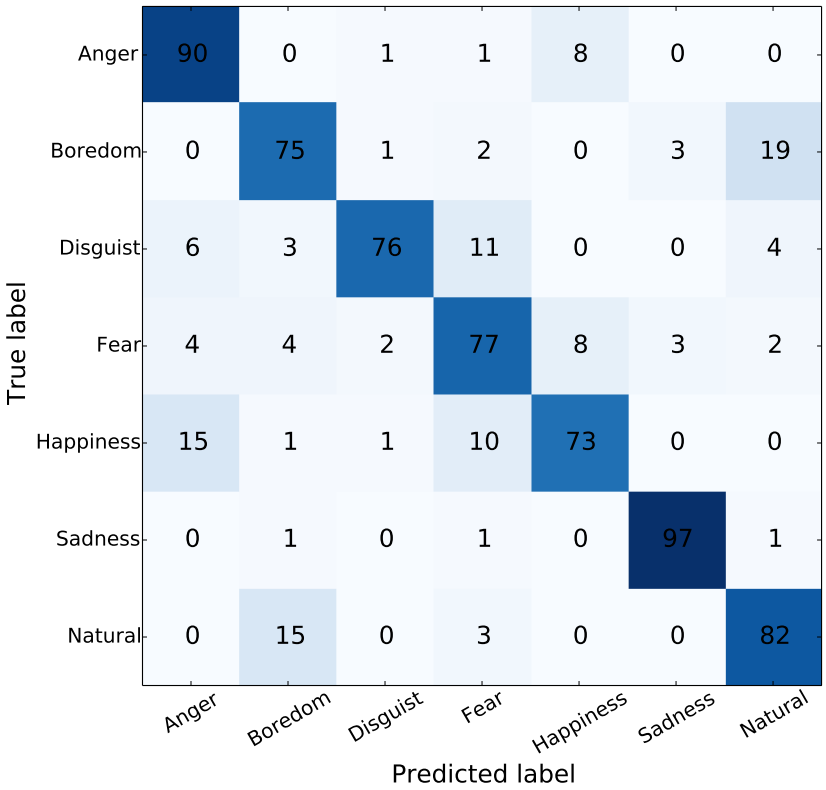
Figure 5.4 Performance (in percent) of the proposed method for different numbers of principal components for each reservoir.

selected). It is quite clear that both reservoirs contribute highly to the final recognition rate. The performance of the proposed method is not very sensitive to the choice of numbers of principal components, since the recognition rate stays above 80% for a wide range of numbers of components.

The highest recognition rate of 82.35% is achieved for 29 and 44 principal components for the vocal tract and the source reservoirs, respectively, and the 95% confidence interval is ±1.36%. Table 5.1 shows the corresponding confusion matrix.

Table 5.2 compares the recognition rate obtained with the proposed method to those obtained with other methods that have been tested on the same emotional speech database. Using feature selection and fusion, the method presented by Jin in [97] achieved 83.10% of correct recognition. It should be noted however that this method was tested on a subset of only 494 speech samples out of 535, which artificially increases the performance. Using an enhanced kernel isomap, the method presented by Zhang in [98] achieved a recognition rate of 80.85%. Finally, using rhyme and temporal features, the one presented by Bhargava in [99] achieved 80.60%. With a recognition rate of 82.35%, the method proposed in this paper compares favorably to these state of the art methods.

Table 5.1 Confusion matrix.



In another experiment, we used an LSM with one single reservoir to recognize emotion directly from the speech signal, without separating the source from the vocal tract. The same preprocessing as for the source component was used. The design of the rest of the system was not changed. After tuning the reservoir, a recognition rate of 75.73% was obtained. The 6.62% difference in recognition rate clearly indicates that including a source-filter model in the recognition system significantly improves performance.

5.6 Conclusions

This paper proposed a new method for automatic recognition of speech emotions based on the Liquide State Machine (LSM), an emerging and very promising tool. This method operates directly on the speech signal and thus requires no feature extraction. It is based on several biological elements. First, the LSM includes a reservoir of spiking neurons which are very close to biological cortical neurons. Then, its original LSM design with two separate reservoirs (one for the source signal and the other for the vocal tract) builds upon the motor theory of human speech perception. This design is more flexible and tunable, as for example the size and memory of the two reservoirs can be tuned separately. One could imagine decomposing the signal even further, using for example rapidly evolving and slowly

Table 5.2 Recognition rate compared with other methods.

Our method	Jin	Zhang	Bhargava
82.35%	*83.10%	80.85%	80.60%

* For a subset of Berlin Database

evolving waveform decomposition of the source signal [100] Finally, the source and vocal tract components of the speech signal are both analyzed on an Equivalent Rectangular Bandwidth (ERB) scale which is a good model for the human peripheral auditory system.

The experimental results showed that this method provides a very good classification performance for an emotion recognition task. It is, however, a very general framework that should also perform well for many other speech processing tasks.

CHAPTER 6

Speech prediction with application to PLC

Preface

Authors and affiliation:

Reza Lotfidereshgi: PhD student, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Philippe Gournay: professor, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Status of paper: final version published

Acceptance date: 13 September 2018

Publication venue: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)

Contribution: The proposed predictor is a single end-to-end network to capture all sorts of dependencies between samples including linear, non-linear and short-term and correlations.

Supplementary information: Appendix A

French title: Prédiction de la parole à l'aide d'un réseau neuronal récurrent adaptatif avec application à la dissimulation de perte de paquets

French abstract:

Cet article propose une nouvelle approche pour la prédiction du signal de parole basée sur un réseau neuronal récurrent (RNN). Contrairement aux prédicteurs à base de RNN existants, qui fonctionnent sur des caractéristiques paramétriques et sont entrainés hors ligne sur une large collection de ces caractéristiques, le prédicteur proposé fonctionne directement sur des échantillons de parole et est entraîné en ligne sur le passé récent du signal vocal. En option, le réseau peut être pré-entraîné hors ligne pour accélérer la convergence au démarrage. Le prédicteur proposé est un réseau unique de bout en bout qui capture toutes sortes de dépendances entre les échantillons et qui a donc le potentiel de surpasser les structures classiques de prédicteur de parole, qu'elles soient linéaire/non linéaire ou à court terme/long terme. Nous l'appliquons au problème de masquage de perte

de paquets (PLC) et montrons qu'il surpasse la technique standard ITU G.711 Appendice I PLC.

Speech prediction using an adaptive recurrent neural network with application to packet loss concealment

Reza Lotfidereshgi, Philippe Gournay

6.1 Abstract

This paper proposes a novel approach for speech signal prediction based on a recurrent neural network (RNN). Unlike existing RNN-based predictors, which operate on parametric features and are trained offline on a large collection of such features, the proposed predictor operates directly on speech samples and is trained online on the recent past of the speech signal. Optionally, the network can be pre-trained offline to speed-up convergence at start-up. The proposed predictor is a single end-to-end network that captures all sorts of dependencies between samples, and therefore has the potential to outperform classical linear/non-linear and short-term/long-term speech predictor structures. We apply it to the packet loss concealment (PLC) problem and show that it outperforms the standard ITU G.711 Appendix I PLC technique.

6.2 Introduction

Consecutive samples or blocks of natural signals are usually correlated with one another. In the case of speech signals [101], most of this correlation comes from the human speech production mechanism, which is by no means memoryless (due to the inertia of the vocal cords and articulators, and to resonances in the vocal tract). In addition to the particular anatomy of the speaker, the limited set of phonemes and words that compose the language he or she speaks (linguistics) and the specific message he or she wants to share (semantics) both introduce further correlation. Successful speech processing applications, including speech compression, recognition and synthesis, make extensive use of these correlations.

In classical speech signal processing, specific structures called predictors are normally used to capture and make use of these correlations. Linear Predictive Coding (LPC) for example relies on various types of linear predictors [102]. Short-term predictors, which

operate at the sample scale (one millisecond or less), handle correlations between nearby samples. Pitch predictors, which operate at a larger time scale (one pitch period, typically 2.5 ms to 20 ms), deal with longer-term correlations. To increase performance, nonlinear predictors (for example based on Volterra or Wiener series) are sometimes used to capture more subtle correlations [103]. These different predictors are generally combined in a cascade, each one taking care of its own type of correlation.

Recently, new machine learning and artificial intelligence tools have been developed to capture correlations in sequential data such as text and speech signals [104]. Hidden Markov Models (HMMs) and Deep Neural Networks (DNNs) [105], which are very efficient at unveiling statistical dependencies for the former and nonlinear dependencies for the latter, are two examples of such tools. Several speech processing applications, in particular speech and speaker recognition and speech synthesis, have made sudden and considerable progress since they were introduced [104].

These modern tools are often treated as black boxes in the sense that the painstaking manual tuning that characterised most classical tools has been replaced by an automatic, thus effortless (yet computationally intensive), training on the largest possible dataset. This approach makes it possible to design very complex systems that provide greater performance. Most of them, however, still rely on a combination of manually and cleverly designed features.

In this paper, we propose a new approach for speech signal prediction based on an adaptive recurrent neural network. Compared to existing approaches, the network operates directly on speech signal samples and is actively trained on the recent past of the speech signal. To demonstrate the merits of this predictor, we apply it to the Packet Loss Concealment (PLC) problem and compare it to the ITU G.711 Appendix I standard [106].

The outline of the paper is as follows. First, the Long Short-Term Memory (LSTM) architecture of Recurrent Neural Networks (RNN), along with classical and modern techniques for Packet Loss Concealment (PLC), are briefly reviewed in section 6.3. The proposed speech predictor structure and its application to PLC are then described in detail in section 6.4. The experimental setup and results obtained are presented in section 6.5. Finally, some conclusions are drawn and perspectives for future research are discussed in section 6.6.

6.3 Relation to prior work

This section presents a brief overview of the long short-term memory architecture used in the proposed approach for speech prediction, and some background information about packet loss concealment.

6.3.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a variation of Recurrent Neural Network (RNN) that is very efficient at solving problems related to sequential data. The concept was first proposed in 1997 [47] then refined over the years [107, 108, 109]. An LSTM network is composed of blocks, each block containing different gates that control the flow of information. The « input » gate controls the flow of information from the input of the block to its memory. The « forget » gate controls the duration for which the information is kept in memory. Finally, the « output » gate controls the contribution of the memorized information to the output activation of the block. Training of LSTM networks is normally done using backpropagation through time [109].

Over the past years, LSTM and other structures with gated units have proven to be very successful in solving various speech-related problems. This goes from processing of text information including automatic translation [110], to processing of acoustic signals such as speech recognition [111, 112]. Most applications to acoustic speech signals consist in classification or recognition tasks. For these applications, training the network first then operating (or testing) it makes much sense. Also, for these applications, the network usually operates on parametric features such as MFCCs or spectrograms rather than directly on input samples, because these features provides some degree of abstraction and the information lost does not really matter.

6.3.2 Packet loss concealment

The purpose of Packet Loss Concealment (PLC) in a Voice over Packet Network (VoPN) speech communication system is to provide a replacement for unavailable (either lost or overly delayed) speech packets [113]. Most, and possibly all, conventional (i.e. signal processing-based) PLC techniques rely exclusively on the most recent past and sometimes on the most immediate future of the speech signal. Conventional PLC techniques that operate directly in the signal domain simply extrapolate or interpolate from one or two pitch periods before and after the packet loss. Model-based or decoder-based conventional PLC techniques rely on parametric features or speech coding parameters that represent one or two frames of speech signal (one frame being typically 10 to 30 ms long). All these techniques have been engineered, or manually designed, based on the general properties of

the speech signal. In particular, the short time-horizon that is used stems from the short decorrelation time that characterizes speech signals.

In the last decade, some more modern (i.e. machine learning or artificial intelligence-based) PLC techniques have been proposed. In reference [114], a statistical Hidden Markov Model (HMM) is used to drive a sinusoidal analysis/synthesis model of the speech signal. In reference [115], a Deep Neural Network (DNN) is used to regenerate the log-power spectrum and phases of the missing frame. The DNN is first trained on a large set of spectral features. Then, in the reconstruction stage, it is fed with the spectral features of the previous frames. To our best knowledge, a modern PLC technique that that is not pre-trained and that operates directly in the signal domain has yet to be proposed.

6.4 Proposed method

This section presents the general structure of the proposed predictor then gives details about offline pretraining, online training and prediction.

6.4.1 General structure

The proposed method for speech prediction and packet loss concealment is illustrated in Fig.6.1. First, as indicated in the dotted-line box, the prediction network can be either randomly initialized or pretrained offline on a set of speech signals. This pretraining is optional but has the advantage of speeding-up convergence of the network at start-up. Then, depending on whether the input speech frame is available or not, the decision is made to simply copy it to the output and train (i.e. adapt or update) the prediction network, or to generate a replacement for the lost frame using the prediction network.

The prediction network operates directly on raw speech samples, without any feature extraction. This is the current trend in speech processing using neural networks [88], mostly because no set of parametric feature has been found so far that is complete enough to capture all dimensions of the speech signal (including speaker identity, emotional state and acoustic background ambiance).

During our first experiments, we considered two options to feed the network: either one single sample at every time step, or with a sliding window of consecutive samples at every time step (with a window shift of one sample). We used the latter approach for two reasons. First, because it allows the network to learn both an internal representation of the speech signal and its evolution through time. Then, because the former approach has proven to produce less stable results especially when reconstructing lost frames (the error in a predicted sample seems to propagate much easier in the single input network).

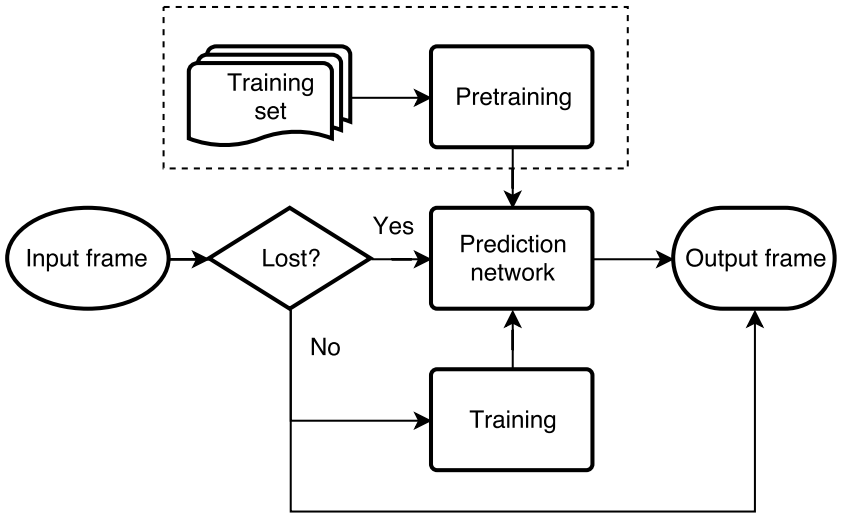


Figure 6.1 Flowchart of the proposed PLC algorithm.

The prediction network itself is composed of several layers of LSTM blocks. In the experiments described in section 6.5, one of the most commonly used LSTM architecture called Vanilla LTSM [116] is used. LSTM states are reset to zero between batches (stateless mode). Peephole connections were not implemented because, according to [116], they do not bring much benefit for prediction. The Mean Squared Error (MSE) between input samples and predicted samples is used as objective training criterion.

6.4.2 Offline pretraining

Optional offline pretraining on a set of speech signals can be used to initialize weights and biases in the network. In the experiments described in section 6.5, this is done using the Adam optimizer [56] and minibatches of 80 samples (10 ms at the 8 kHz sampling rate).

This pretraining alone is not enough to provide good prediction performance. Speech is both an exceptionally diverse and extremely dynamic signal. It is therefore difficult for a single network to follow continuously and accurately its slightest local variations. Training and executing a single giant network to do so does not seem practical either. Keeping track of the evolution of the signal while maintaining a reasonable complexity requires using a smaller network and training it on a more representative set of signals. This is why we use online training on the recent past of the speech signal.

6.4.3 Online training or prediction

There are two possibilities during the operation phase of the network. If the input speech frame is available, it is simply copied to the output of the system as illustrated in Fig.6.1. It is also considered as a small but extremely relevant training set for the network. Multiple

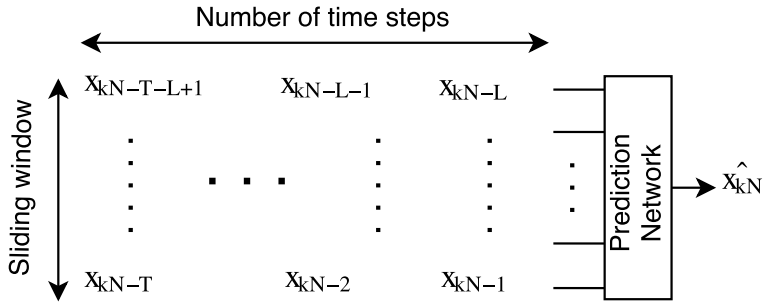


Figure 6.2 Signal samples used to train the prediction network for the first sample of frame number k. N is the frame duration in samples, L is the size of the sliding window, and T is the number of time steps

training passes are made over that small training set in a process that is often referred to as stochastic optimization [56]. The samples used to train the prediction network are illustrated in Fig.6.2.

If no input frame is available, then the prediction network performs regression to predict the first sample of the output frame. This sample is then used as input to predict the next sample of the lost frame. This process goes on until the entire lost frame has been reconstructed.

6.5 Experiments

The experiments presented in this section are intended to explore a variety of configurations for the proposed speech predictor. Performance on a PLC task is studied because this is a straightforward and typical application for speech prediction. Also, the ITU-T G.711 Appendix I standard is used as a reference, not only because of its good performance but also because it is a representative example of conventional signal processing-based PLC.

6.5.1 Speech material

The experiments are done on a subset of the TIMIT database [117]. The sampling frequency is 8000 Hz and the frame (or packet) duration is set to 10 ms. The number of neurons per layer is intentionally small to limit the complexity of online training. To reduce the extent of the experiments, the length of the sliding window is chosen to always be the same as the number of neurons per layer.

Two hundred speech files from the training subset of the TIMIT database are used for pretraining. They represent more than 50,000 frames of speech signal, for a total of 4 million of training speech samples. Since the number of neurons per layer is small, a single epoch of pretraining is performed.

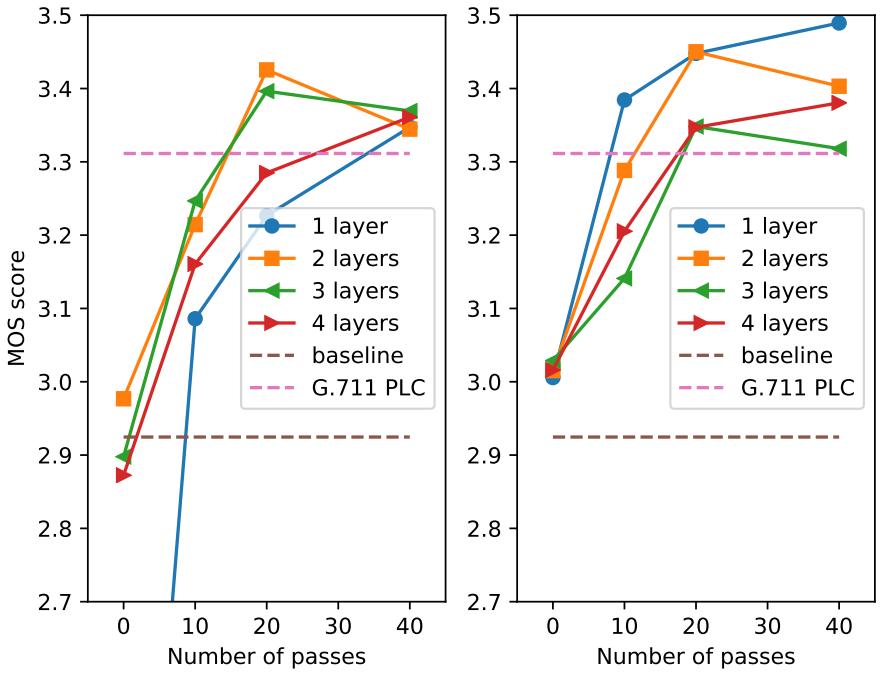


Figure 6.3 MOS score as a function of number of passes for different configurations of the network, for 80 time steps (left) and 160 time steps (right).

Ten speech files from the testing subset of the TIMIT database are used for testing. A 10% packet loss rate is simulated, with lost packets being evenly spaced. The test files therefore represent more than 400 lost packets, for a total of 32,000 lost, hence predicted, samples. The performance of the proposed method is evaluated in terms of Mean Opinion Score (MOS) and compared to that of the G.711 PLC algorithm. The MOS score is obtained using the Perceptual Evaluation of Speech Quality (PESQ) software tool [118] [1].

6.5.2 Results

Fig.6.3 present the MOS as a function of the number of training passes for different numbers of LTSM layers (from 1 to 4). The left panel corresponds to 80 time steps and the right one to 160 times steps. In all cases, both the number of neurons per layer and the size of the sliding window are set to 80. The lower and upper dotted lines correspond to the MOS of a baseline condition (lost packets simply set to zero) and to the MOS of the G.711 PLC algorithm, respectively. From these results, we conclude that the network cannot predict speech efficiently after pretraining only (zero training passes). Specifically, in the case of 80 time steps, the performance of the proposed PLC system is generally below the baseline condition. Another conclusion is that better performance is obtained with

1. PESQ is an objective method for end-to-end speech quality assessment and it was standardized as recommendation ITU-T P.862 [119]

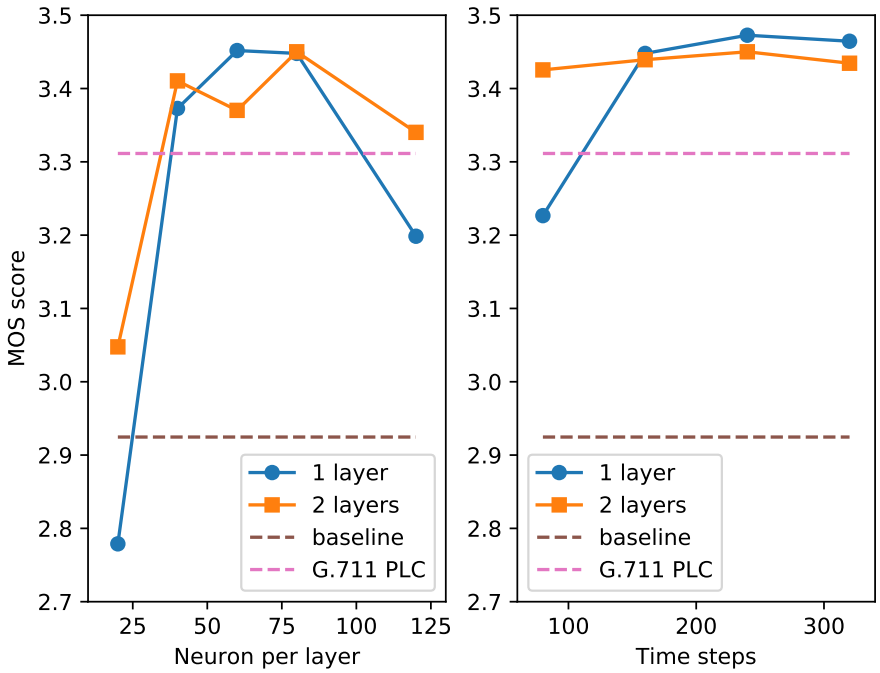


Figure 6.4 MOS score as a function of the number of neurons per layer (left) of the number of time steps (right).

160 time steps than with 80. Also, 20 training passes generally gives good performance. Lower performance for more that 20 passes probably results from overfitting because each pass is performed on a small number of data.

Fig.6.4 presents the results obtained when varying the number of neurons per layers (left panel) and the number of time steps (right panel). In both cases, the number of passes is equal to 20. The first conclusion is that even a single layer of only 40 LSTM neurons performs reasonably well when online training is used. The second conclusion is that increasing the number of time steps beyond 160 does not seem to increase performance significantly.

6.6 Conclusions

In this paper, a new approach for speech signal prediction based on a recurrent neural network was proposed. The main characteristic of this approach is that the network is actively trained on the recent past of the signal. Since this recent past represents the best possible training set almost at all times, the network is able to follow closely the evolutions of the signal. Furthermore, since the local variability of the signal is limited compared to the variability of speech in general, a small network structure is effective. This, in turn, allows the network to operate directly on speech samples. And since there is no more need

for a manually-design feature representation, no information is lost compared to systems that include a feature extraction step.

The proposed speech signal predictor was tested on a packet loss concealment task. With proper setting, it was shown to outperform the standard ITU-T G.711 Appendix I PLC algorithm. It is interesting to note that these good results were obtained using a completely speech-agnostic system, in the sense that no speech model nor prior information about subjective speech quality evaluation was introduced in its design. Good performance was achieved even when using a very small LSTM networks (one layer of forty neurons). This is interesting because complexity can rapidly be an issue when performing online training or output regression.

Only a limited number of LSTM configurations were tested. Different neural network configurations, different pretraining and training procedures, or even different types of neural networks may further improve performances.

Finally, the proposed predictor is a very general tool that could benefit to other applications in speech processing, and that could apply to other correlated yet highly dynamic types of data.

CHAPTER 7

Cognitive coding of speech

Preface

Authors and affiliation:

Reza Lotfidereshgi: PhD student, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Philippe Gournay: professor, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Status of paper: final version accepted

Acceptance date: 16 February 2022

Publication venue: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2022)

Contribution: We designed and trained a neural network to recognize the building blocks of speech signals as representations in its layers.

Supplementary information: Appendix B and C

French title: Codage cognitif de la parole

French abstract:

Nous proposons une approche de codage cognitif de la parole par extraction non supervisée de représentations contextuelles à deux niveaux hiérarchiques d'abstraction. Les attributs vocaux tels que l'identité du phonème qui durent cent millisecondes ou moins sont capturés dans le niveau d'abstraction inférieur, tandis que les attributs vocaux tels que l'identité du locuteur et l'émotion qui persistent jusqu'à une seconde sont capturés dans le niveau d'abstraction supérieur. Cette décomposition est réalisée par un réseau de neurones à deux étages, avec un étage inférieur et un étage supérieur fonctionnant à des échelles de temps différentes. Les deux étages sont entraînées pour prédire le contenu du signal dans leurs espaces latents respectifs. Une voie descendante entre les étages améliore encore la capacité prédictive du réseau. Avec une application dans la compression de la parole à l'esprit, nous étudions l'effet de la réduction de la dimensionnalité et de la quantification à faible débit sur les représentations extraites. Les performances mesurées sur les bases de

données LibriSpeech et EmoV-DB atteignent, et même dépassent pour certains attributs vocaux, celles des approches qui constituent l'état de l'art actuel.

Cognitive coding of speech

Reza Lotfidereshgi, Philippe Gournay

7.1 Abstract

We propose an approach for cognitive coding of speech by unsupervised extraction of contextual representations in two hierarchical levels of abstraction. Speech attributes such as phoneme identity that last one hundred milliseconds or less are captured in the lower level of abstraction, while speech attributes such as speaker identity and emotion that persist up to one second are captured in the higher level of abstraction. This decomposition is achieved by a two-stage neural network, with a lower and an upper stage operating at different time scales. Both stages are trained to predict the content of the signal in their respective latent spaces. A top-down pathway between stages further improves the predictive capability of the network. With an application in speech compression in mind, we investigate the effect of dimensionality reduction and low bitrate quantization on the extracted representations. The performance measured on the LibriSpeech and EmoV-DB datasets reaches, and for some speech attributes even exceeds, that of state-of-the-art approaches.

7.2 Introduction

The human cognitive system is known to have a hierarchical organization, the most cognitively complex operations being performed at the top of the hierarchy. While information mostly flows from the bottom to the top of the hierarchy, this bottom-up flow is often influenced by what is already known at the top of the hierarchy. Furthermore, there is substantial evidence for the predictive nature of this top-down influence [120, 121]. A parallel can be drawn between these defining elements of the cognitive system and the models used in machine learning. One of the first successful applications of deep learning was precisely in the field of automatic learning of hierarchical representations [122, 44]. It was also found that introducing top-down processes in hierarchical models improves the

quality of learned representations, thereby increasing the accuracy of recognition systems based on these representations [123, 124]. Predictive coding has also been shown to be a successful strategy in machine learning when processing various data modalities [124, 125].

Unsupervised learning not only reduces the need for labeled datasets, it also makes it possible to build comprehensive hierarchical representations that provide a deep insight into the nature of the input data. This is particularly important in speech compression, where efficiency depends on the completeness and compactness of the representation, which should capture all sorts of speech attributes. Yet despite the great potential of unsupervised learning, domain-specific representation learning, which can only capture a subset of the attributes from labeled data, is still prevalent in the literature. Currently, one of the very few approaches to extract comprehensive speech representations is the Vector Quantized Variational Autoencoder (VQ-VAE) [17]. Its use in recent deep learning-based speech coders and synthesizers [73, 74, 126] substantiates the need for compact and complete speech representations.

In this paper, we propose and evaluate a new approach for unsupervised learning and extraction of speech representations that heavily relies on the principles of cognition. First, a two-stage neural network model is used to extract representations in two levels of abstraction, with a lower stage and an upper stage processing information from short and long frames of data, respectively. Secondly, a top-down pathway between stages is introduced, which has the effect of improving the quality of the representations. Finally, predictive coding is used as the learning strategy. The performance of the proposed approach is measured in terms of classification accuracy for speaker identity, emotions and phonemes. To position the results of the proposed approach with respect to the current state of the art, Contrastive Predictive Coding (CPC) [125] is used as a baseline. We observe that the second stage of the proposed model delivers a compact and remarkably high-quality long-term representation of the speech signal. The quality of the short-term representation extracted by the first stage is improved compared to that of the CPC baseline, especially when the dimension of the representation is reduced. Finally, we demonstrate that the extracted representations are extremely robust to quantization.

7.3 Relation to prior work

The proposed Cognitive Coding model utilizes predictive coding in two stages and includes a top-down process between stages. These two stages produce two representations that evolve at a different pace and thus correspond to different levels of abstraction. The representations are extracted by maximizing the mutual information between the latent

variables and the speech signal. Finally, the mutual information is maximized by minimizing a contrastive loss.

Mutual information is a fundamental quantity measuring the relationship between random variables. In previous work, it has been used in the formulations of Generative Adversarial Networks (GANs) [14] and Variational Autoencoders (VAEs) to make them learn interpretable representation of data [127, 128, 129]. Noise Contrastive Estimation (NCE) is a method for parameter estimation of probabilistic models by discriminating data from noise [130, 131]. In the model called Contrastive Predictive Coding (CPC) [125], NCE is also formulated as a probabilistic contrastive loss that maximizes the mutual information between the encoded representations and the input data.

In the CPC model, an encoder maps the input data to a sequence of latent variables, and an autoregressive model produces another sequence of latent variables. The InfoNCE loss introduced in [125] optimizes the discrimination of a positive sample from multiple negative samples. In this paper, we optimize a similar objective with consideration of two levels of abstraction and the presence of a top-down process. We also implemented the CPC algorithm as a baseline against which to compare our results.

7.4 Cognitive coding of speech

The architecture and learning algorithm of the Cognitive Coding model are illustrated in Fig. 7.1 [1]. The architecture can be described as follows. First, an encoder maps short frames of speech signal $x_s(t)$ to a sequence of latent variables $z_s(t)$ while decreasing the temporal resolution. Then, another encoder maps the first sequence of latent variables $z_s(t)$ to another set of latent variables $z_l(t)$ while further decreasing the temporal resolution and increasing the receptive field to match long frames of speech signal. In this study, we use layers of Convolutional Neural Networks (CNNs) as encoders. Finally, two autoregressive models map $z_s(t)$ and $z_l(t)$ to two sequences of contextual representations $c_s(t)$ and $c_l(t)$. In this study we use Gated Recurrent Units (GRUs) for the autoregressive models.

We begin by describing the learning algorithm for the lower stage of the model. In this lower stage, the mutual information between both contextual representations and short frames of speech signal can be expressed as:

$$I(x_s; c_s, c_l) = \sum_{x_s, c_s, c_l} p(x_s, c_s, c_l) log \frac{p(x_s|c_s, c_l)}{p(x_s)} \tag{7.1}$$

1. In Fig. 7.1 solid lines represent forward path in the network and dotted lines represent prediction paths used for formulation of the loss function.

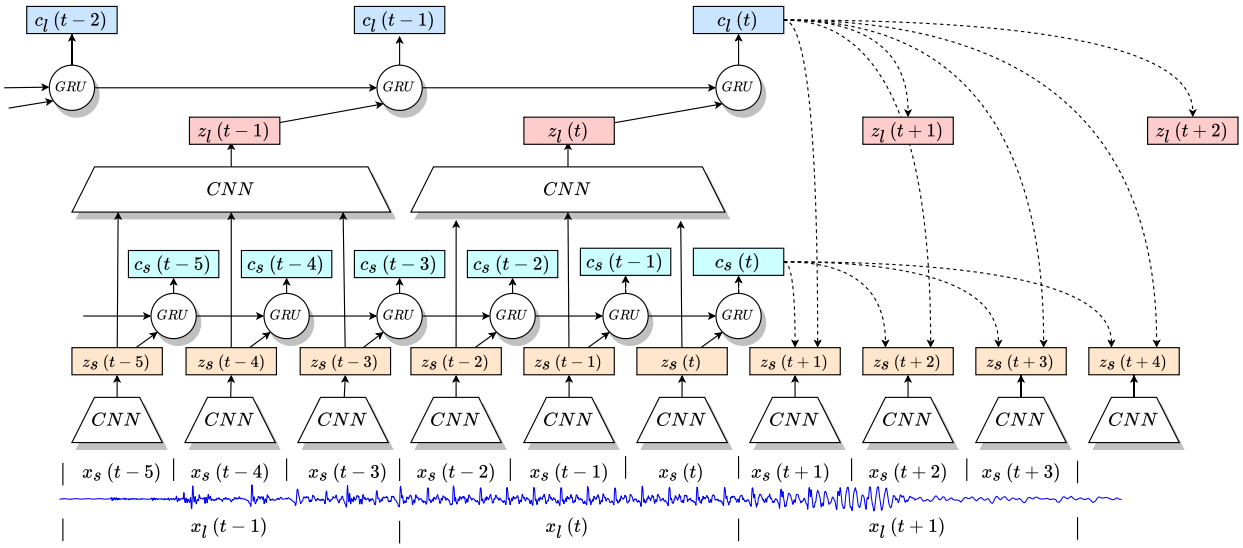


Figure 7.1 The architecture and learning algorithm of the cognitive coding model. The ratio between long and short frames in the diagram is chosen to be three for illustration purposes. In this study the actual frame ratio is eight.

The following unnormalized density ratio captures the mutual information between a future short frame of speech signal at step $t+k$ and both contextual representations:

$$f_k(x_s(t+k), c_s(t), c_l(t)) \propto \frac{p(x_s(t+k)|c_s(t), c_l(t))}{p(x_s(t+k))} \tag{7.2}$$

As in the CPC model, we do not use a generative model to produce future frames of speech signal. Rather, we use the following quantity to approximate f_k:

$$\exp(z_s^T(t+k)W_s(k)g(c_s(t), c_l(t))) \tag{7.3}$$

In equation (7.3), $W_s(k)$ is a linear transformation used for the prediction of $z_s(t+k)$ (k steps in the future) and $g(c_s(t), c_l(t))$ is a function of both contextual representations that constitutes the input of the linear transformation. While a neural network could be used for g to perform a nonlinear transformation, we simply repeat the long-term representation to match the temporal resolution of the short-term representation and concatenate it with the short-term representation to be used as input for the linear prediction of $z_s(t+k)$ by $W_s(k)$. This is perfectly justified because the upper stage of our model produces a long-term representation that is easily interpretable by linear classifiers (see section 7.5.1).

Finally, the loss function is derived according to noise contrastive estimation which is the categorical cross entropy of classifying one positive sample of short frames of speech signal

from $N-1$ negative ones [2]:

$$L_N = \mathop{\mathbb{E}}_{X_s}\left[log\frac{f_k(x_s(t+k), c_s(t), c_l(t))}{\sum_{x_s(j)\in X_s} f_k(x_s(j), c_s(t), c_l(t))}\right] \tag{7.4}$$

For the upper stage of the model, an equivalent of equations (7.1-7.4) can be derived based on long frames of speech signal $x_l(t)$. c_s is omitted from equations (7.1-7.2). Furthermore, since there is no top-down pathway in the upper stage, the prediction of $z_l(t+k)$ is based only on the long-term contextual representation $c_l(t)$ and the approximation for the density ratio becomes:

$$\exp(z_l^T(t+k)W_l(k), c_l(t)) \tag{7.5}$$

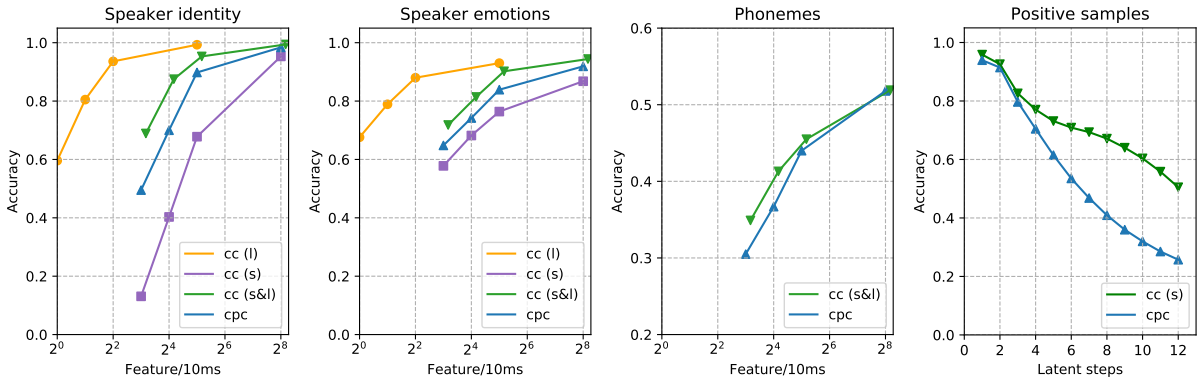
The loss function is derived by substituting equation (7.5) in equation (7.4), and samples are drawn from long frames of speech signal.

7.5 Experiments

This section presents experimental results regarding various speech attributes and investigates the effects of dimensionality reduction and quantization on the quality of the representations. Two different datasets were used. First, a 100-hour subset of the LibriSpeech dataset [132] was used to evaluate the performance of the proposed approach on phonemes (a short-term attribute) and on speaker identity (a long-term attribute). We used forced-aligned phoneme labels as well as the test and train split from [125] so that we could obtain comparable results. Secondly, we used the Emov-DB dataset [133] to evaluate the performance of the proposed approach on speaker emotions which is another long-term attribute.

The encoder used in the lower stage consists of five layers of CNNs with filter sizes [10, 8, 4, 4, 4] and with strides [5, 4, 2, 2, 2]. The encoder in the upper stage consists of three layers of CNNs with filter sizes [4, 4, 4] and with strides [2, 2, 2]. Each layer has 512 hidden dimensions with ReLu activations. As a result, the lower and upper encoders downsample their input by a factor of 160 and 8, respectively. We trained on 20480-sample windows of speech signal sampled at 16kHz. As a result, the lower and upper encoders produce z_c and z_l vectors of features once every 10 ms and 80 ms, respectively. We decided that the dimension of the hidden state of GRUs would be either 8, 16, 32 or 256 so that the network

2. Positive samples for short and long frames are from distributions $p(x_s(t+k)|c_s(t), c_l(t))$ and $p(x_l(t+k)|c_s(t), c_l(t))$ respectively. Negative samples for short and long frames are from distributions $p(x_s(t+k))$ and $p(x_l(t+k))$ respectively.



(a) (b) (c) (d)

Figure 7.2 Linear classification of attributes and prediction accuracy of positive samples in the loss function. (s: short-term, l: long-term, CC: Cognitive Coding. CPC: Contrastive Predictive Coding.)

can produce representations of various dimensions. Prediction is done twelve steps in the future, which extends the window of prediction up to 120 ms in the future for the lower stage and 960 ms for the upper stage. We trained with a learning rate of 2e-4, using mini batches of 8 samples, and performed approximately 300k updates.

7.5.1 Linear classification

The performance of our model is measured by training linear classifiers for various speech attributes to show to what extent the extracted features are linearly interpretable. Fig. 7.2 (a-c) presents the performance of linear classification for speaker identity, emotion and phonemes. Fig. 7.2 (d) shows the ability of the lower stage of the proposed model to predict positive samples in the loss function up to twelve steps in the future. The results are reported for classifying contextual representations extracted from long frames of signal (l), short frames of signal (s), combined contextual representations (s&l) as well as contextual representations of the CPC model. The following observations can be made based on the results.

Regarding the baseline, the results reported in [125] for the 256-dimension representation which produces 256 features every 10 ms are 97.4% and 64.6% of accuracy for speaker identity classification and phoneme classification, respectively. With our implementation of CPC, we were able to achieve a higher accuracy of 98.4% for speaker identity but a lower accuracy of 51.9% for phonemes.

Since the upper stage of our model produces a set of features for each 80 ms of speech signal, the number of features per 10 ms is 8 times less relative to the lower stage of our

model and to the CPC model. For long-term attributes (speaker identity and emotion) the proposed network outperforms CPC in terms of linear classification for combined 256-dimension representations by achieving an accuracy of 99.3% and 94.4% for speaker identity and emotion, respectively. The corresponding accuracy achieved by the CPC model was 98.4% and 91.9%. By reducing the dimensionality of the representations, we observe that a high degree of linear separation between speaker identities and emotions is maintained when considering the features extracted by the upper stage of our model. Features extracted by the lower stage provide lower performance for long-term attributes. Overall this is a desirable effect that we attribute to the top-down pathway which helps decorrelate long-term attributes from short-term representations.

Regarding linear classification of phonemes based on contextual representations, we achieved 52% accuray, a lower performance than the state of the art with forced aligned features provided by [125]. Strangely enough, this is true even with our implementation of CPC baseline model. However, phoneme information is encoded in latent variable z_s which has a smaller receptive field compared to both contextual representations. Besides, not all information is linearly interpretable. In an experiment we used a classifier with one hidden layer on contextual representations and latent variables z_s and z_l and accuracy increased to 64.1%. Features of z_s are also a candidate for dimensionality reduction to encode information in a smaller time scale.

We also investigated the effect of the top-down pathway on the prediction of positive samples in the lower stage and compared the performance of our model with that of the CPC baseline in the same setup. Fig. 7.2 (d) shows that the proposed approach is able to predict positive samples of short frames more efficiently beyond 3 latent steps.

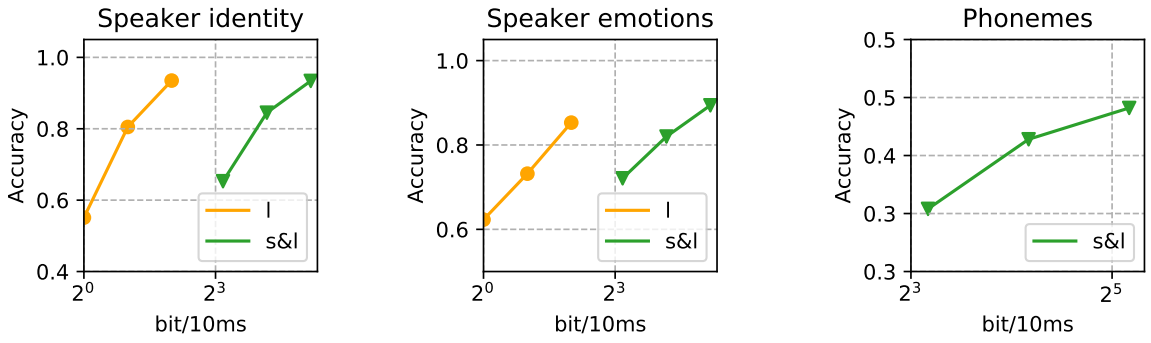


Figure 7.3 Linear classification of quantized features.

7.5.2 Quantization

In this study, we also investigated the compressibility of the features. Since each stage predicts twelve time steps in the future, the contextual representations have a slow-evolving nature and we observe that the features exhibit a high degree of temporal dependency. For this reason, we decided we would quantize the features using 1-bit Δ-modulation. The initial values of the features are encoded on 5 bits. Fig. 7.3 shows the results obtained when the features are quantized for the most interesting configurations from Fig. 7.2. We only consider representations with 32 dimensions and less because they are the most likely to be used in speech compression applications. For the majority of the cases, the performance of the linear classification is within 5% of the corresponding performance from Fig. 7.2. Most notably, we observe that our model can encode long-term speech attributes such as speaker identity and emotion with more that 50% accuracy at bitrates as low as 100 bit/s.

7.6 Conclusion

In this paper, we presented a new model for cognitive coding of speech that combines several principles of cognition. Specifically: (1) it produces a hierarchy of representations that correspond to different levels of abstraction; (2) it uses the predictive coding principle; and (3) it includes a top-down pathway between levels of abstractions. The hierarchy of representations captures a wide variety of speech attributes over a broad range of time scales. Experiments show that this hierarchy is also easily interpretable, well suited for compression, and remarkably robust to quantization. This cognitive coding model should therefore find applications in high-quality speech synthesis, voice transformation and speech compression.

CHAPTER 8

Practical cognitive speech compression

Preface

Authors and affiliation:

Reza Lotfidereshgi: PhD student, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Philippe Gournay: professor, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Status of paper: final version accepted

Acceptance date: 18 February 2022

Publication venue: IEEE Data Science and Learning Workshop (DSLW)

Contribution: A fully machine learning-based speech codec developed based on the cognitive coding approach presented in Chapter 7.

Supplementary information: Appendix B and C

French title: Compression cognitive pratique de la parole

French abstract:

Cet article présente une nouvelle méthode de compression neuronale de la parole qui est pratique dans le sens où elle fonctionne à faible débit, introduit une faible latence, est compatible en termes de complexité de calcul avec les appareils mobiles actuels et fournit une qualité subjective comparable à celle des codecs de téléphonie mobile standard. D'autres vocodeurs neuronaux récemment proposés ont également la capacité de fonctionner à faible débit. Cependant, ils ne produisent pas le même niveau de qualité subjective que les codecs standards. D'autre part, les codecs standard s'appuient sur des métriques objectives et à court terme telles que le rapport signal/bruit segmental qui ne sont que faiblement corrélés à la perception. De plus, les codecs standard sont moins efficaces que les réseaux de neurones non supervisés pour capturer les attributs de la parole, en particulier ceux à long terme. La méthode proposée combine un codeur à codage cognitif qui extrait une représentation hiérarchique non supervisée interprétable avec un décodeur multi-étages qui a une architecture basée sur GAN. Nous observons que cette méthode est

très robuste à la quantification des caractéristiques de représentation. Un test AB a été effectué sur un sous-ensemble des phrases de Harvard qui sont couramment utilisées pour évaluer les codecs de téléphonie mobile standard. Les résultats montrent que la méthode proposée surpasse le codec standard AMR-WB en termes de délai, de débit et de qualité subjective.

Practical cognitive speech compression

Reza Lotfidereshgi, Philippe Gournay

8.1 Abstract

This paper presents a new neural speech compression method that is practical in the sense that it operates at low bitrate, introduces a low latency, is compatible in computational complexity with current mobile devices, and provides a subjective quality that is comparable to that of standard mobile-telephony codecs. Other recently proposed neural vocoders also have the ability to operate at low bitrate. However, they do not produce the same level of subjective quality as standard codecs. On the other hand, standard codecs rely on objective and short-term metrics such as the segmental signal-to-noise ratio that correlate only weakly with perception. Furthermore, standard codecs are less efficient than unsupervised neural networks at capturing speech attributes, especially long-term ones. The proposed method combines a cognitive-coding encoder that extracts an interpretable unsupervised hierarchical representation with a multi stage decoder that has a GAN-based architecture. We observe that this method is very robust to the quantization of representation features. An AB test was conducted on a subset of the Harvard sentences that are commonly used to evaluate standard mobile-telephony codecs. The results show that the proposed method outperforms the standard AMR-WB codec in terms of delay, bitrate and subjective quality.

8.2 Introduction

Voice quality, bitrate and latency are important attributes in real-time speech compression. Voice quality and latency both play a major role in customer experience, while bitrate is a requirement dictated by network operators. High voice quality requires, first, encoding the most relevant features with a high compression ratio, second, quantization with a

minimum amount of information loss, and finally, reconstruction of a natural-sounding speech signal while avoiding perceivable imperfections and artefacts. Designing a low latency codec that delivers high voice quality is a challenging task since latency limits both encoder and decoder algorithms on their access to information. Consequently, it also limits to some extent the quality of synthesized speech in conversational codecs.

Currently deployed conversational speech codecs use classical signal processing methods [134, 135, 5]. In the range of medium bitrates such as the one used in mobile telephony (around 13 kbits/s), the main strategy is to synthesize a speech signal that is as close as possible to the original one (waveform matching) using objective metrics such as the signal-to-noise ratio (SNR) measured on short segments of speech signal. The second strategy used in this range of bitrates is to weight the SNR so that the coding noise is shaped according to the properties of human perception.

In recent years, machine-learning based compression methods have successfully improved some attributes of speech codecs. Deep learning-based speech synthesizers have been used as decoders to produce speech from classical speech-coding parameters and it has been shown that they can produce higher quality speech than standard decoders currently in use. However, early approaches such as Wavenet [15] were much more computationally complex and slow to synthesize speech signals. More recent approaches synthesize speech with comparable quality, with much higher synthesis speed [83, 136] and less complexity [52, 53, 54]. In some other models, learned speech features such as the ones extracted by Vector-Quantized Variational Auto-Encoders (VQ-VAE) [17] replace classical features. This leads to fully machine-learning based speech codecs. These models have been proven to achieve higher compression ratio compared to other approaches [73, 74].

It is a known fact that the objective measures used in current standard speech codecs do not correlate well with perceived speech quality. The short-term waveform-matching strategy does not correspond to what is known about the biological mechanism of hearing, and speech is processed in a much different way in the human auditory system compared to what happens in current coding algorithms. Recent speech codecs based entirely on machine-learning (neural vocoders) also produce features that preserve a subset of speech attributes, but while they produce intelligible speech at very low bitrates, they don't achieve the same subjective quality as the codecs used in mobile telephony.

In this paper, we propose a fully-learned speech codec based on hierarchical and interpretable features. Using a model of Cognitive Coding (CC) of speech [137], we extract unsupervised hierarchical representations that preserve short-term and long-term contex-

tual speech attributes in two levels of abstraction. With the extracted features, attributes such as phonemes and speaker identity and emotions are well preserved and highly separable with linear classifiers. We also observe that these features have a large dynamic range and that they are very robust to quantization. We introduce a two stage decoder to reconstruct a speech signal from the representations in both levels of abstraction with a GAN-based approach. We also introduce CC loss for training decoder as a measure to improve the accuracy of both short-term and long-term attributes in the synthesized speech signal.

To prove the concept, we trained a cognitive speech codec that operates at 8 kbps and we show the speech quality surpasses that of AMR-WB at both 8.85 kbps and 12.65 kbps. The algorithmic delay of the proposed method is 20ms compared to a minimum of 25ms for AMR-WB. In summary, we achieved higher speech quality relative to AMR-WB with less latency while keeping the bitrate at the lower end of standard conversational speech codecs.

8.3 Relation to prior work

Cognitive compression aims to extract fundamental and interpretable representations of speech signals in the encoder and synthesize natural-sounding speech in the decoder. By maximizing mutual information in the encoder, using cognitive coding of speech [137], two compact representations that evolve at different paces and correspond to different levels of abstraction are extracted while redundant information is discarded. An adversarial loss combined with multiple feature-based losses is used to infer the redundancy that was eliminated at the encoder in a two stage decoder with a minimum amount of sound compression artefacts.

In previous works, unsupervised features extracted by models such as VQ-VAE have been used as a basis for low-bitrate compression [73, 74] and they have been shown to produce interpretable representations of phonemes [17, 129]. Contrastive Predictive Coding (CPC) [125], a method based on Noise Contrastive Estimation (NCE) [130], has been used to extract unsupervised representations and it has been demonstrated that some classes such as speaker identity and phonemes are separable with a linear classifier under these representations. Using theories of cognition, representations in multiple levels of abstraction are also extracted by the CC model. Some aspects of the CPC model are improved in the CC model which also serves as a basis for the encoder in the proposed method.

To synthesize speech, the majority of neural vocoders model the probability distribution of speech samples. Autoregressive models form a major class of synthesizers. They factorize

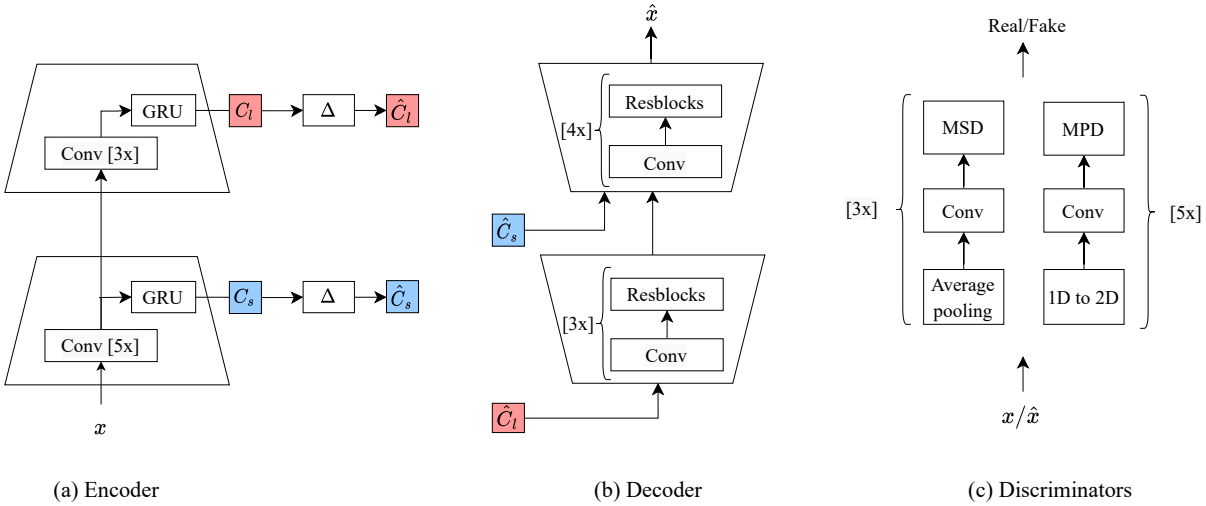


Figure 8.1 Components of the cognitive speech compression model including the encoder, the decoder and the discriminators that are used for training the decoder.

the probability distribution as a product of conditional distributions. The ability of autoregressive models to produce high quality speech has been demonstrated in models such as WaveNet [15], SampleRNN [138] and WaveRNN [16]. Although autoregressive models are able to produce high-quality speech, they are generally computationally complex and their serial-processing nature makes them less suitable for parallel processing and real-time applications.

Non-autoregressive models applied to speech synthesis can be divided into flow-based generative models and GAN-based models. Parallel WaveNet [136] and WaveGlow [83] are examples of flow-based models which can take full advantage of parallel processing and are also able to produce speech signals with high perceptual quality. GAN-based models such as MelGAN [52], HiFiGAN [54] and StyleGAN [53] are commonly used for synthesis of a speech signal from a Mel-spectrogram. Until recently, despite better computational efficiency and memory usage, GAN-based approaches lagged behind autoregressive models and flow-based models in terms of quality of generated speech. However, recent advances such as Feature Matching Loss [139], Multi-Scale Discriminator [52] and Multi-Period Discriminator [54] have greatly improved the quality delivered by GAN-based methods.

8.4 Proposed approach

The architecture of the proposed cognitive speech compression model is illustrated in Fig.8.1. The goal of cognitive compression is to encode the input speech signal x into two

sets of representation (C_s, C_l) and quantize these as ($\hat{C}_s$, $\hat{C}_l$). An output signal $\hat{x}$ is then reconstructed from the quantized representations. The different components and training objectives of the proposed model are explained in the rest of this section.

8.4.1 Encoder

The design of the encoder network is illustrated in Fig.8.1(a). We use a variation of the CC model. As described in [137], the encoder architecture is composed of two levels of abstraction. At each level, an encoder is composed of convolutional layers followed by Gated Recurrent Units (GRUs). The output of the GRUs, C_s and C_l (contextual representations), are trained to predict an intermediate latent representation within each level of abstraction using NCE loss. Further details about the CC model can be found in [137]. In this paper, first, convolution layers are made causal to avoid any look ahead window and additional latency. Second, GRUs have linear activation instead of tanh to facilitate learning and produce features with more dynamic range that are more suitable for the following quantization stages. GRUs in each layer are followed by a Δ modulation block that quantizes the difference between the current and the previous value of each feature.

8.4.2 Decoder

The design of the decoder network is illustrated in Fig.8.1(b). We utilize a two-stage decoder to generate a synthesized speech signal $\hat{x}$ from two sets of representations. First, the quantized long-term representations C_l are upsampled through transposed convolutions [1] until the length of the output sequence matches the short-term representation C_s. Then the output sequence is combined with C_s and upsampled until the length of the final sequence matches the resolution of the speech signal waveform. In both stages of the decoder, transposed convolutions are used without padding to avoid algorithmic delay by filters. Transposed convolutions are followed by stacked blocks of multiple residual convolutions, the output of which being summed. This layer of residual blocks was introduced in [54] as Multi-Receptive Field Fusion.

8.4.3 Discriminators

Coupling a generator (the decoder in our model) with an ensemble of multiple discriminators for training said generator is a common practice in recent GAN-based speech synthesizers [52, 53, 54, 141]. The design of the network is illustrated in Fig.8.1(c) and

1. Transposed convolution is also known as a deconvolution or a fractionally-strided convolution. More information about deconvolution can be found in [140] and an implementation of transposed convolution can be found in: https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose1d.html

is composed of MSD [1] and MPD [2] blocks. MSD, which was introduced in the MelGAN [52] model and reused in [54], is a mixture of three discriminator blocks operating on a raw signal and downsampled signals with factors of 2 and 4 using strided average pooling. Each block in MSD is a stack of 4 strided convolutions. MPD, which was introduced in [54], is also a mixture of discriminator blocks. Each block consists of a stack of 5 convolutional layers which operate on a 2D array of speech samples. The 2D array is composed of selected samples of raw speech with length T. Samples are selected periodically with periods of $p \in [2, 3, 5, 7, 11]$ for each block and rearranged into a 2D array with dimensions [p, T/P]. Further description of these two types of discriminators can be found in [52, 54].

8.4.4 Decoder's training objective

Loss terms and intuitions for training the decoder as generator G using discriminator D (which represents both MSD and MPD discriminators) are described as follows.

GAN objective

In the proposed model, we use the LSGAN formulation of adversarial losses [142], which has shown better performances compared to some other formulations in speech synthesizers. Expressions (8.1-8.2) demonstrate adversarial loss for training discriminator D and generator G (decoder), respectively.

$$\mathbb{E}_{x,(\hat{C}_s,\hat{C}_l)}\left[(D(x)-1)^2 + D(G(\hat{C}_s,\hat{C}_l))^2\right] \tag{8.1}$$

$$\mathbb{E}_{(\hat{C}_s,\hat{C}_l)}\left[D(G(\hat{C}_s,\hat{C}_l)-1)^2\right] \tag{8.2}$$

CC representation distances

Let's consider the encoder in Fig.8.1(a) without quantization stages as two separate functions: $C_s = \xi_s(x)$ and $C_l = \xi_l(x)$ which extract short-term and long-term representations from the raw waveform, respectively. We introduce two objectives in addition to the adversarial objectives:

$$\mathbb{E}_{x,(\hat{C}_s,\hat{C}_l)}\left[||\xi_i(x) - \xi_i(G(\hat{C}_s,\hat{C}_l))||_1\right] \tag{8.3}$$

in which $i \in \{s, l\}$ for short-term and long-term representations. CC representations capture all sorts of short-term and long-term speech attributes. By minimizing the distance in the latent space of representations, we first enforce the fidelity of the synthesized signal

1. Multi-scale discriminator
2. Multi-period discriminator

to the original in terms of phoneme accuracy, speaker identity, emotion, etc. Second, since unquantized representations are used in the expressions above, the decoder recovers some of the information that has been lost in the quantization stage.

Mel-spectrum distance

Based on the Mel-spectrum distance, the decoder is able to infer more precise spectral details about the original signal from CC features extracted by the encoder and thus increase the fidelity of the synthesized speech to the original. The Mel-spectrum objective is described in the following expression:

$$\mathbb{E}_{x,(C_s,C_l)}\left[||\phi(x) - \phi(G(C_s, C_l))||_1\right] \tag{8.4}$$

where $\phi(x)$ is the mel-spectrogram of the waveform x.

Feature matching distance

Introduced in [139], feature matching is a similarity metric between the discriminator's intermediate features while the discriminator operates on real speech and synthesized speech. This loss term can be described as the following expression:

$$\mathbb{E}_{x,(\hat{C}_s,\hat{C}_l)}\left[\sum_{i=1}^{L} ||D_i(x) - D_i(G(\hat{C}_s, \hat{C}_l))||_1\right], \tag{8.5}$$

where N_i denotes the number of features D_i in the ith layer of discriminator blocks.

Total objective

While the loss value for training the discriminator can be described as expression 8.1, the loss term for training the generator is a weighted sum of expressions 8.2-8.5 by the values $\lambda = [1, 10, 10, 50, 2]$ respectively.

8.5 Experiments

This section presents experimental conditions and subjective test results for the proposed speech compression method.

8.5.1 Configurations and training

A 100-hour subset of the LibriSpeech dataset [132] was used for training the networks. The hyperparameters of the encoder and decoder networks are shown in the Table.8.1. The parameters for the discriminator networks can be found in [52, 54]. Encoder layers have 512 hidden dimensions with ReLu activation. Both encoder stages produce a 64 dimensional representation, the lower stage once every 10ms and the upper stage once every 80ms. Each stage of the encoder is followed by a single-bit quantizer based on Δ modulation.

Table 8.1 Hyper-parameters of encoder and decoder networks. Lower and top level refer to blocks as they appear in Fig.8.1(a,b)

Encoder		
lower level	*filter size*	[10, 8, 4, 4, 4]
	downsample	[5, 4, 2, 2, 2]
	GRU hidden size	64
top level	*filter size*	[4, 4, 4]
	downsample	[2, 2, 2]
	GRU hidden size	64
Decoder		
top level	*filter size*	[4, 4, 4]
	upsample	[2, 2, 2]
	residual filter size	[3, 7, 11] $\times$ 3
lower level	*filter size*	[10, 8, 8, 4]
	upsample	[5, 4, 4, 2]
	residual filter size	[3, 7, 11] $\times$ 3

The quantization step size is different for the two stages of the encoder but it is the same for all 64 features of a representation. The total number of bits after quantization of features adds up to 8 kbps. Fig.8.2 illustrates an example of raw and quantized features extracted by the encoder from 800ms of a speech signal sample. Transposed convolutions in the decoder start with 128 hidden dimensions for the lower stage and 256 for the upper stage and the number of dimensions drops by a factor of 2 progressively. Only short-term representations from 20ms of signal are buffered for speech synthesis which results in an overall algorithmic delay of exactly 20ms.

The total number of parameters is 9.8M for the encoder and 6.3M for the decoder. Without any further optimization and using a GTX 1080 GPU, the speed of speech synthesis is more than 100x faster than real-time which places the proposed method well within the range of computational ability of current mobile devices.

8.5.2 Subjective tests

AMR-WB is a widely-used standard codec for mobile communications which can operate at different bitrates. We compared our proposed method with AMR-WB operating at 8.85 kbps which is the closest bitrate to the 8 kbps of the proposed method. We also compared it to AMR-WB operating at 12.65 kbps, which offers a higher level of subjective quality and it is the most common configuration currently in use. It should be noted that both

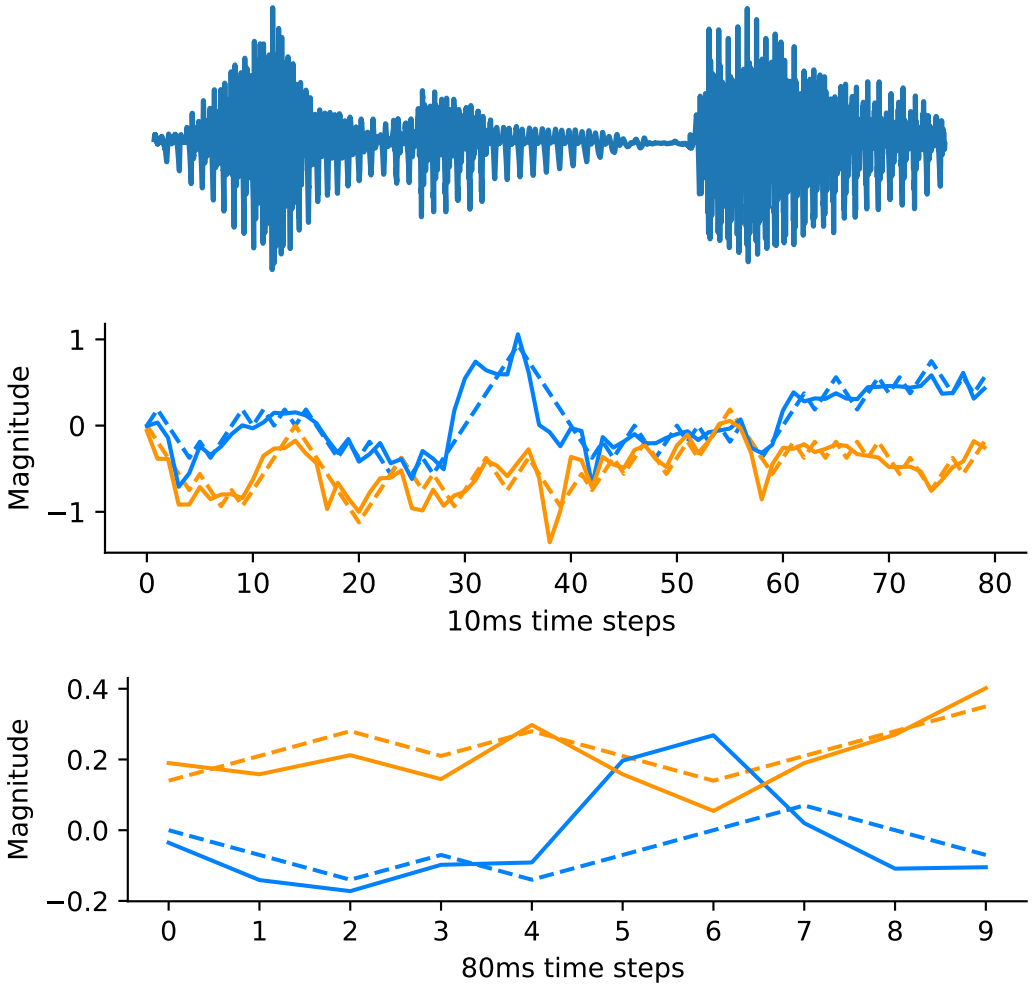
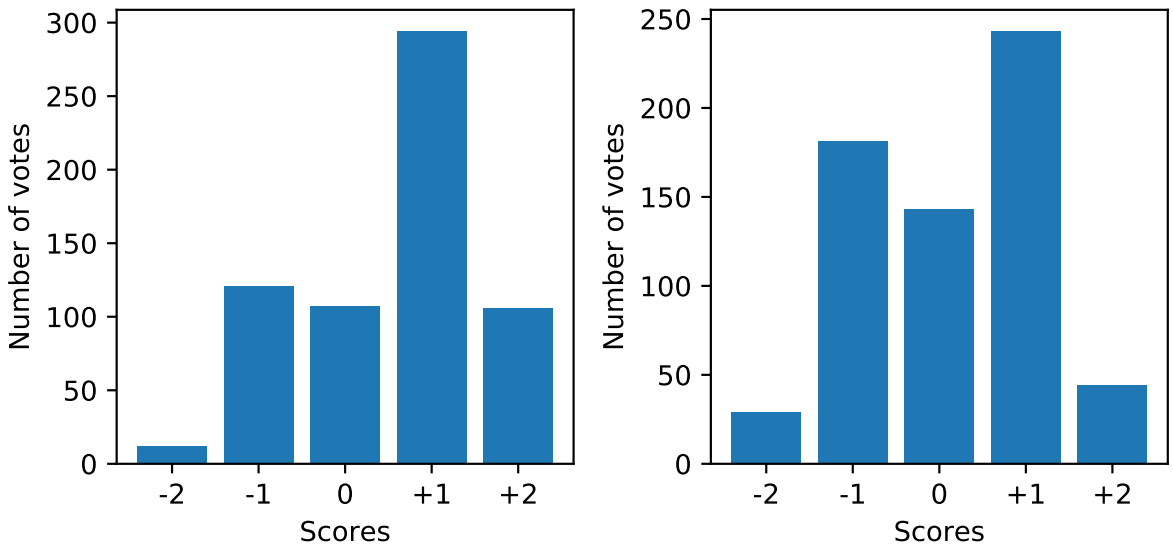


Figure 8.2 Raw and quantized (dashed lines) features extracted with encoder form 800ms of a speech signal. Only two features from each level of representation are illustrated.

codecs operate at 16kHz and that the proposed method does not extend the bandwidth to increase the perceptual quality of speech.

To validate our results, we performed two AB tests without reference with a five-point scale from -2 to +2 [1]. The test equipment consists in a Focusrite Scarlett 2i2 USB audio interface + Beyerdynamic DT 770 headphones. The test material was composed of 32 pairs of Harvard sentences that were recorded by Dynastat and are regularly used to evaluate speech coding standards. Test sentences and speakers were not part of the training set. There were 20 test subjects, all considered as naive listeners.

1. In the performed AB test, the subject's preference when comparing two variations A and B of the same audio sample is recorded. Score 0 indicates no preference, 1 small preference and 2 strong preference. A positive score indicates preference towards B and a negative towards A.



(a) AMR-WB (8.85 kbps) (b) AMR-WB (12.65 kbps)

Figure 8.3 AB test results. Comparison of the proposed method operating at 8 kbps with AMR-WB operating at 8.85 kbps and 12.65 kbps. Positive scores denote preference for the proposed method.

The results of the experiment are shown in Fig.8.3. The average scores are summarized in Table 8.2 for male speakers, female speakers and all speakers combined. Results show that, on average, the proposed codec at 8 kbps clearly outperforms AMR-WB at 8.85 kbps and appears to be even slightly better than AMR-WB at 12.65 kbps.

An audio demonstration of the proposed approach is available [1].

8.5.3 Discussion

While the proposed method is shown to have superior quality relative to both AMR-WB references, some minor artifacts such as occasional reverberation effects and some temporal discontinuities can be perceived and still need to be eliminated. One source of potential improvement is simply more training. The decoder used in the subjective test was trained for only 900k iterations. While this amount of training is perfectly adequate for some types of neural networks, millions of iterations are more common when training GAN-based speech synthesizers. Experiments done after the listening test show that the majority of the observed artifacts disappear beyond 2M iterations and the speech quality can be improved further.

1. https://rezalo.github.io/CognitiveCoding

Table 8.2 Average difference of scores based on the performed AB tests between proposed method and reference codec. Positive scores denotes preference for the proposed method.

$Reference$	$Total$	$Male$	$Female$
$AMR-WB(8.85kbps)$	+0.56	+0.49	+0.64
$AMR-WB(12.65kbps)$	+0.14	+0.11	+0.18

We strongly believe that the proposed approach can be further improved in terms of compression ratio, complexity and latency. Previous studies have shown that CC features preserve essential speech attributes even when representations are extracted with much lower dimensions [137]. Other studies have shown that GAN-based speech synthesis can be performed with much less complexity [54]. Testing different network configurations and coding attributes is extremely time consuming but we will explore such possibilities for further improvements.

8.6 Conclusion

In this paper, we presented a new method for real-time compression of speech that is based on a cognitive coding model. Low latency, low bitrate, acceptable complexity and superior quality are the key attributes that make the proposed method a practical real-time speech compression tool. Because it relies on easily interpretable representations that are also robust to quantization, the proposed approach has potential for a wide range of applications in speech communication, and there is a clear path for further developments to meet various needs from service providers.

CHAPTER 9

English conclusion

9.1 Summary

In this thesis, to deal with the complexity of designing a codec, a gradual approach with multiple tasks is chosen to design a fully machine learning-based speech codec. While in each task, a tool for speech compression is introduced with a potential to be used independently, each development step also provides insights and the basis for the next step towards a self-contained codec.

In the first two tasks, classification and regression techniques are applied to the solution of classical problems in speech processing but could also be used as building blocks in existing speech codecs. The results of these tasks are published in two papers: *Biologically Inspired Speech Emotion Recognition* [143], reproduced in Chapter 5, and *Speech Prediction Using an Adaptive Recurrent Neural Network with Application to Packet Loss Concealment* [144], reproduced in Chapter 6. In the first paper, we extracted features from raw speech signals with unsupervised learning using a biologically inspired process, and showed that a simple classifier can learn emotions, a sophisticated long-term attribute, from these features. In the second paper, a speech sample predictor, which is an essential part of many speech codecs, is developed and it is shown that it is capable of producing high quality speech when applied to PLC.

Next, elements used for partial development of codecs are combined and extended towards a fully machine learning-based speech codec. Two tasks are carried out to design first an encoder, then a decoder. The results of these tasks are published in two papers: *Cognitive Coding of Speech* [137], reproduced in Chapter 7, and *Practical Cognitive Speech Compression* [145], reproduced in Chapter 8. In the first paper, we extracted speech representations with unsupervised learning based on theories of cognition to function as a speech encoder. The principles of cognition that influenced the design are: hierarchically-organized levels of abstraction, predictive coding, and a top-down pathway between the representations. The extracted representations are compact, easily interpretable and robust to quantization. In the second paper, a decoder with a multi stage GAN-based architecture is trained

to produce speech signals from the cognitive coding (CC) representations. Both the design and its training focus on achieving high perceptual quality. The codec resulting from these two tasks operates at a low bitrate and low latency, and it delivers high quality speech.

9.2 Comparison with existing approaches

Techniques developed for each task in this thesis can be compared with alternative approaches from the fields of machine learning or signal processing. In this section, we compare each technique with its alternatives.

9.2.1 Classification

Classification is the most common task in machine learning, and there are a multitude of approaches for classification of all sorts of speech attributes. However, the proposed approach differs from alternatives due to a combination of several factors: modeling biological components of the human auditory system, using SNNs to extract features, and also the unsupervised learning method employed for feature extraction. Compared to the majority of alternative classifiers, the proposed method focused on extracting general-purpose features and using a simple classifier to recognize emotions; nevertheless, it achieves the performance of state-of-the-art classifiers that use manually engineered features [97, 99] on the same emotion classification task.

9.2.2 Regression

The regression task performed in this thesis can be best compared with signal processing-based methods. A predictor is a specific structure that is used to capture and make use of correlations in speech for sample prediction. The most widely used method of prediction is linear prediction, which typically relies on two types of linear predictors [102]: one for short-term correlations operating at the sample scale ($\sim$ one millisecond), and the other for a larger time scale ($\sim$ 2.5-20ms) called pitch predictor. A few alternative approaches have also been proposed to capture limited and subtle non-linear correlations, such as the ones based on Volterra series [146, 103]. Compared with such approaches, the LSTM used in the proposed approach has a more general ability in capturing non-linearities. Secondly, using online training, the network is able to capture correlations on a larger time scale (hundreds of milliseconds, instead of tens of milliseconds when nonlinear series are used). However, in terms of computational complexity, the proposed approach is more complex than the conventional signal processing tools.

9.2.3 Representation learning

Despite the great potential of unsupervised representation learning for capturing speech representations, currently, only few approaches are used to extract comprehensive speech

representations. Among these approaches, the CPC and VQ-VAE models are the most relevant to the representation learning task presented in this thesis. The proposed approach in Chapter 7 is comprehensively compared with the CPC model. As a summary, compared to CPC, CC provides hierarchical representations from two time scales. Specially, when a compact long-term representation is extracted, CC is much more predictive of long-term attributes of speech (such as speaker ID and emotions) than CPC. VQ-VAE is another method to extract compact representation, and is commonly used in recent deep learning-based speech coders and synthesizers [73, 74, 126]. Contrary to VQ-VAE, CC extracts hierarchical representations that are also highly interpretable with linear classifiers. This property can help extend the functionality of a codec beyond compression, for example to perform voice transformation.

9.2.4 Speech compression

The proposed fully-learned speech codec that operates based on hierarchical and interpretable features (Chapter 8) can be compared with two classes of speech codecs: standardized conversational speech codecs that are based on classical signal processing methods and machine-learning based codecs.

Currently standardized conversational speech codecs [134, 135, 5] mainly attempt to synthesize a speech signal that is as close as possible to the original waveform. Metrics such as the SNR is a common objective for such waveform matching strategies. Sometimes, the SNR is weighted to take into account how the coding noise is perceived by humans. Unlike waveform matching, instead of preserving the waveform, the proposed method preserves short-term and long-term speech attributes based on the metrics derived from mutual information in the encoder and an adversarial loss in the decoder. Multiple other metrics are also included in the decoder's trainig to approximate subjective quality of speech in a more comprehensive way than objective measures used in signal processing-based codec designs. However, computational complexity of the proposed model is higher than signal processing-based codecs, although real-time implementation on cellphones with the current technology is feasible. Moreover, the development of the proposed codec is in its early stage. The proposed model is trained on clean speech and its resilience to environmental factors such as noise has not been considered yet. Also, measures to increase its reliability in the presence of network errors such as frame loss and jitter have not been considered.

Alternative machine learning-based methods, which can be used in the structure of an encoder, have already been discussed in Section 9.2.3. In recent years, some machine-learning based methods have also been developed that can be used for speech synthesis and as a decoder. Initially, it has been shown that the quality of signal processing meth-

ods can be surpassed with machine learning-based methods such as WaveNet [15], even if classical speech parameters are used for speech synthesis. However, these methods are computationally complex and the speed of synthesis is slow. The more recent approaches synthesize speech with comparable quality and much higher speed [83, 136]. The computational complexity is also further reduced with GAN-based structures [52, 53, 54]. While the computational complexity of the proposed model is similar to such GAN-based structures, it introduces two novel techniques in the design. First, the two stage GAN-based structure to combine the CC representations, and secondly, a metric based on CC representation (CC representation distance) to match the quality of representation between the resynthesized samples and the original ones.

9.3 Future work

In this thesis, we established a framework for gradual development of a fully machine learning-based speech compression method. We performed multiple tasks, many points are discussed, and some are worth further investigation. In this section, we discuss a number of suggestions for future research to further develop machine learning-based speech compression.

Classification At the time of conception of the proposed classifier, SNNs had several important limitations, including computational complexity of simulating a large number of spiking neurons, and inefficiency of learning methods for reconstruction of accurate speech signals. As a result, all the tasks that followed classification are performed with ANNs. However, the learning algorithms and the computational efficiency of SNNs have improved significantly in recent years for both supervised and unsupervised learning [62, 63, 64, 65, 66, 67]. SNN-based processors are also very efficient in terms of power consumption. As a result, the development of a SNN-based speech codec could be reconsidered. Especially, with the advent of techniques to translate ANN-based models to run on SNN-based processors, implementing a SNN-based variation of a fully learned codec such as the one developed in this thesis seems feasible.

Regression After having developed a predictor for the regression task, we extended the prediction task from sample space to latent space in order to achieve representation learning. However, there are alternative paths for the development of codecs based on predictors. For example, a predictor such as the one proposed in this thesis can be optimized in terms of prediction gain, instead of focusing on subjective quality with PESQ as we did. A predictor with a superior prediction gain can find alternative applications in

speech compression. For example, such a predictor can improve the compression ratio in lossless coding.

Speech coding based on representation learning The proposed approach is developed with only a subset of potential applications and codec attributes in mind. The proposed method can be further improved to extend its application scope, and it can be improved to meet the necessary criteria considering all of the codec attributes. The followings are potential tasks that can be performed to further develop the proposed speech codec:

- To increase the robustness of the codec, a jitter management and a packet loss concealment can be implemented for the proposed method. Both the network's design and its training can be modified to accommodate for errors in the communication network.
- Environmental effects such as acoustic background can affect the quality of speech coding. The proposed approach can be trained with noisy speech to improve the proposed method for practical applications.
- We only trained the network on speech datasets, and the network was trained to only learn speech attributes. Learning music attributes, or general audio attributes, are interesting tasks to be considered, to extend the application scope of the proposed method.
- Encoding multichannel audio is another interesting task, and the proposed approach seems to be able to capture representation of multichannel audio very efficiently without too much overhead compared to single channel coding. There clearly is a potential to extend the current approach for multichannel coding.

CHAPTER 10

Conclusion française

10.1 Résumé

Dans cette thèse, pour faire face à la complexité de la conception d'un codec, une approche progressive avec plusieurs tâches est choisie pour concevoir un codec vocal entièrement basé sur l'apprentissage automatique. Alors que dans chaque tâche, un outil de compression de la parole est introduit avec un potentiel d'utilisation indépendante, chaque étape de développement fournit également des informations et la base de l'étape suivante vers un codec complet.

Dans les deux premières tâches, les techniques de classification et de régression sont appliquées à la résolution de problèmes classiques de traitement de la parole, mais pourraient également être utilisées comme blocs de construction dans les codecs vocaux existants. Les résultats de ces tâches sont publiés dans deux articles : *Biologically Inspired Speech Emotion Recognition* [143], reproduit au chapitre 5, et *Speech Prediction Using an Adaptive Recurrent Neural Network with Application to Packet Loss Concealment* [144], reproduit au chapitre 6. Dans le premier article, nous avons extrait des caractéristiques de signaux de parole bruts avec un apprentissage non supervisé à l'aide d'un processus d'inspiration biologique, et avons montré qu'un simple classificateur peut apprendre à reconnaitre l'émotion, un attribut à long terme particulièrement complexe, à partir de ces caractéristiques. Dans le deuxième article, un prédicteur d'échantillon de parole, qui est une partie essentielle de nombreux codecs vocaux, est développé et il est démontré qu'il est capable de produire une parole de haute qualité lorsqu'il est appliqué à la dissimulation des paquets perdus.

Ensuite, les éléments utilisés pour le développement partiel des codecs sont combinés et étendus vers un codec vocal entièrement basé sur l'apprentissage automatique. Deux tâches sont réalisées pour concevoir d'abord un codeur, puis un décodeur. Les résultats de ces tâches sont publiés dans deux articles : *Cognitive Cognitive of Speech* [137], reproduit au chapitre 7, et *Practical Cognitive Speech Compression* [145], reproduit au chapitre 8. Dans le premier article, nous avons extrait des représentations de la parole avec un

apprentissage non supervisé basé sur des théories de la cognition pour fonctionner comme un encodeur de la parole. Les principes de cognition qui ont influencé la conception sont: des niveaux d'abstraction organisés de manière hiérarchique, un codage prédictif, et une voie descendante entre les représentations. Les représentations extraites sont compactes, facilement interprétables et robustes à la quantification. Dans le deuxième article, un décodeur avec une architecture basée sur GAN à plusieurs étages est formé pour produire des signaux vocaux à partir des représentations de codage cognitif (CC). La conception du réseau et son entrainement se concentrent tous deux sur l'obtention d'une qualité de perception élevée. Le codec résultant de ces deux tâches fonctionne à un faible débit binaire et une faible latence, et il délivre une voix de haute qualité.

10.2 Comparaison avec les approches existantes

Les techniques développées pour chaque tâche dans cette thèse peuvent être comparées à des approches alternatives issues des domaines de l'apprentissage automatique ou du traitement du signal. Dans cette section, nous comparons chaque technique avec ses alternatives.

10.2.1 Classification

La classification est la tâche la plus courante en apprentissage automatique, et il existe une multitude d'approches pour la classification de toutes sortes d'attributs vocaux. Cependant, l'approche proposée diffère des alternatives en raison d'une combinaison de plusieurs facteurs : la modélisation des composants biologiques du système auditif humain, l'utilisation de SNN pour extraire les caractéristiques, ainsi que la méthode d'apprentissage non supervisée utilisée pour l'extraction des caractéristiques. Par rapport à la majorité des classificateurs alternatifs, la méthode proposée s'est concentrée sur l'extraction de caractéristiques à usage général et sur l'utilisation d'un classificateur simple pour reconnaître les émotions ; néanmoins, il atteint les performances des classificateurs de pointe qui utilisent des caractéristiques conçues manuellement [97, 99] sur la même tâche de classification des émotions.

10.2.2 Régression

La tâche de régression effectuée dans cette thèse peut être mieux comparée aux méthodes basées sur le traitement du signal. Un prédicteur est une structure spécifique qui est utilisée pour capturer et utiliser des corrélations dans la parole pour la prédiction d'un échantillon. La méthode de prédiction la plus largement utilisée est la prédiction linéaire, qui repose généralement sur deux types de prédicteurs linéaires [102]: l'un pour les corrélations à court terme opérant à l'échelle de l'échantillon ($\sim$ une milliseconde), et l'autre pour

une échelle de temps plus grande ($\sim$ 2,5-20 ms) appelée prédicteur de pitch. Quelques approches alternatives ont également été proposées pour capturer des corrélations non linéaires limitées et subtiles, telles que celles basées sur les séries de Volterra [146, 103]. Par rapport à de telles approches, le LSTM utilisé dans l'approche proposée a une capacité plus générale à capturer les non-linéarités. Deuxièmement, grâce à l'entrainement en ligne, le réseau est capable de capturer des corrélations sur une plus grande échelle de temps (des centaines de millisecondes, au lieu de dizaines de millisecondes lorsque des séries non linéaires sont utilisées). Cependant, en termes de complexité de calcul, l'approche proposée est plus complexe que les outils conventionnels de traitement du signal.

10.2.3 Apprentissage des représentations

Malgré le grand potentiel de l'apprentissage non supervisé des représentations pour capturer des représentations vocales, actuellement, seules quelques approches sont utilisées pour extraire des représentations vocales complètes. Parmi ces approches, les modèles CPC et VQ-VAE sont les plus pertinents pour la tâche d'apprentissage de représentation présentée dans cette thèse. L'approche proposée au chapitre 7 est globalement comparée au modèle CPC. En résumé, par rapport à CPC, CC fournit des représentations hiérarchiques à partir de deux échelles de temps. En particulier, lorsqu'une représentation compacte à long terme est extraite, CC est beaucoup plus prédictif des attributs à long terme de la parole (tels que l'identification du locuteur et les émotions) que CPC. VQ-VAE est une autre méthode pour extraire une représentation compacte et est couramment utilisée dans les récents codeurs et synthétiseurs vocaux basés sur l'apprentissage profond [73, 74, 126]. Contrairement à VQ-VAE, CC extrait des représentations hiérarchiques qui sont également hautement interprétables avec des classificateurs linéaires. Cette propriété peut aider à étendre la fonctionnalité d'un codec au-delà de la compression, par exemple pour effectuer une transformation vocale.

10.2.4 Compression de la parole

Le codec vocal entièrement appris proposé qui fonctionne sur la base de caractéristiques hiérarchiques et interprétables (chapitre 8) peut être comparé à deux classes de codecs vocaux : les codecs vocaux conversationnels standardisés basés sur des méthodes classiques de traitement du signal, et les codecs basés sur l'apprentissage automatique.

Les codecs vocaux conversationnels actuellement standardisés [134, 135, 5] tentent principalement de synthétiser un signal vocal aussi proche que possible de la forme d'onde d'origine. Des métriques telles que le SNR sont un objectif commun pour de telles stratégies de suivi de forme d'onde. Parfois, le SNR est pondéré pour tenir compte de la façon dont le bruit de codage est perçu par les humains. Contrairement au suivi de forme d'onde,

au lieu de préserver la forme d'onde, la méthode proposée préserve les attributs vocaux à court et à long terme basés sur les métriques dérivées des informations mutuelles dans l'encodeur et d'une perte contradictoire dans le décodeur. De multiples autres métriques sont également incluses dans l'entrainement du décodeur pour se rapprocher de la qualité subjective de la parole d'une manière plus complète que les mesures objectives utilisées dans les conceptions de codecs basés sur le traitement du signal. Cependant, la complexité de calcul du modèle proposé est supérieure à celle des codecs basés sur le traitement du signal, bien que leur mise en œuvre en temps réel sur les téléphones portables avec la technologie actuelle soit réalisable. De plus, le développement du codec proposé en est à ses débuts. Le modèle proposé est entraîné sur de la parole propre et sa robustesse aux facteurs environnementaux tels que le bruit n'a pas encore été prise en compte. De plus, les mesures visant à augmenter sa fiabilité en présence d'erreurs de réseau telles que la perte de trames et la gigue n'ont pas été envisagées.

Des méthodes alternatives basées sur l'apprentissage automatique, qui peuvent être utilisées dans la structure d'un encodeur, ont déjà été discutées dans la section 10.2.3. Ces dernières années, certaines méthodes basées sur l'apprentissage automatique ont également été développées et peuvent être utilisées pour la synthèse vocale et comme décodeur. Initialement, il a été démontré que la qualité des méthodes de traitement du signal peut être surpassée avec des méthodes basées sur l'apprentissage automatique telles que WaveNet [15], même si des paramètres vocaux classiques sont utilisés pour la synthèse vocale. Cependant, ces méthodes sont complexes en termes de calcul et la vitesse de synthèse est lente. Les approches les plus récentes synthétisent la parole avec une qualité comparable et une vitesse beaucoup plus élevée [83, 136]. La complexité de calcul est également encore réduite avec les structures basées sur GAN [52, 53, 54]. Bien que la complexité de calcul du modèle proposé soit similaire à de telles structures basées sur GAN, il introduit deux nouvelles techniques dans la conception. Premièrement, la structure basée sur GAN en deux étapes pour combiner les représentations CC, et deuxièmement, une métrique basée sur la représentation CC (distance de représentation CC) pour faire correspondre la qualité de représentation entre les échantillons resynthétisés et ceux d'origine.

10.3 Travaux futurs

Dans cette thèse, nous avons établi un cadre pour le développement progressif d'une méthode de compression de la parole entièrement basée sur l'apprentissage automatique. Nous avons effectué plusieurs tâches, de nombreux points sont discutés, et certains méritent d'être approfondis. Dans cette section, nous discutons d'un certain nombre de suggestions

de recherches futures pour développer davantage la compression de la parole basée sur l'apprentissage automatique.

Classification Au moment de la conception du classificateur proposé, les SNN présentaient plusieurs limitations importantes, notamment la complexité de calcul de la simulation d'un grand nombre de neurones à décharges et l'inefficacité des méthodes d'apprentissage pour la reconstruction de signaux vocaux précis. En conséquence, toutes les tâches qui ont suivi la classification sont effectuées avec des ANN. Cependant, les algorithmes d'apprentissage et l'efficacité de calcul des SNN se sont considérablement améliorés ces dernières années pour l'apprentissage supervisé et non supervisé. Les processeurs basés sur SNN sont également très efficaces en termes de consommation d'énergie. En conséquence, le développement d'un codec vocal basé sur SNN pourrait être reconsidéré. En particulier, avec l'avènement de techniques permettant de traduire des modèles basés sur ANN pour les exécuter sur des processeurs basés sur SNN, la mise en œuvre d'une variante basée sur SNN d'un codec entièrement appris tel que celui développé dans cette thèse semble faisable.

Régression Après avoir développé un prédicteur pour la tâche de régression, nous avons étendu la tâche de prédiction de l'espace échantillon à l'espace latent afin de réaliser l'apprentissage de la représentation. Cependant, il existe des voies alternatives pour le développement de codecs basés sur des prédicteurs. Par exemple, un prédicteur tel que celui proposé dans cette thèse peut être optimisé en termes de gain de prédiction, au lieu de se concentrer sur la qualité subjective avec PESQ comme nous l'avons fait. Un prédicteur avec un gain de prédiction supérieur peut trouver des applications alternatives dans la compression de la parole. Par exemple, un tel prédicteur peut améliorer le taux de compression dans le codage sans perte.

Codage de la parole basé sur l'apprentissage des représentations Le codec basé sur l'apprentissage automatique proposé peut être développé davantage en termes d'applications et d'attributs. L'approche proposée est développée avec seulement un sous-ensemble d'applications potentielles et d'attributs de codec à l'esprit. Le procédé proposé peut être encore amélioré pour étendre sa portée d'application, et il peut être amélioré pour répondre aux critères nécessaires compte tenu de tous les attributs de codec. Les tâches suivantes peuvent être effectuées pour développer davantage le codec vocal proposé:

– Pour augmenter la robustesse du codec, une gestion de la gigue et une dissimulation de perte de paquets peuvent être implémentées pour la méthode proposée. La con-

ception du réseau et son entraînement peuvent être modifiés pour tenir compte des erreurs dans le réseau de communication.

- Les effets environnementaux tels que le bruit de fond acoustique peuvent affecter la qualité du codage de la parole. L'approche proposée peut être entraînée avec de la parole bruitée pour améliorer la méthode proposée pour des applications pratiques.
- Nous n'avons entraîné le réseau que sur des ensembles de données de parole, et le réseau a été entraîné pour n'apprendre que les attributs de la parole. L'apprentissage des attributs musicaux, ou des attributs audio généraux, sont des tâches intéressantes à considérer, pour étendre le champ d'application de la méthode proposée.
- L'encodage de l'audio multicanal est une autre tâche intéressante, et l'approche proposée semble être capable de capturer très efficacement la représentation de l'audio multicanal sans trop de surcharge par rapport au codage monocanal. Il existe clairement un potentiel d'extension de l'approche actuelle pour le codage multicanal.

APPENDIX A

Details for implementation of experiments in Chapter 6

A.1 Details for implementation

To help the reproducibility of results presented in Chapter 6, we describe the proposed method in further detail in this section. First of all, the implementation of 'Vanilla' LSTM mentioned in Charter 6 is the same structure as the LSTM presented in Section 3.4.3 where σ is the sigmoid function. Second, to further clarify the terminology, the *number of neurons* in Charter 6 is synonymous to *hidden size* or *number of units* in an LSTM cell. Furthermore, the term *block* or *layer* is used to describe an LSTM cell when it is a part of a larger structure in a network. Finally, when LSTM layers are stacked, the output h_t of the previous layer becomes the input x_t of the following layer.

A.2 Hyper parameters

The ability of the network presented in Chapter 6 to capture correlations between speech samples depends on the configuration of the network. A list of the hyperparameters used for experiments performed in Section 6.5 is presented in the following tables.

Table A.1 A list of hyperparameters of experiments performed in Chapter 6

Hyperparameters in Fig. 6.3 (left)	
window size	80
neurons	80
layers	$\in [1, 2, 3, 4]$
time steps	80
training passes	$\in [0, 10, 20, 30, 40]$

Hyperparameters in Fig. 6.3 (right)	
window size	80
neurons	80
layers	$\in [1, 2, 3, 4]$
time steps	160
training passes	$\in [0, 10, 20, 30, 40]$

Hyperparameters in Fig. 6.4 (left)	
window size	80
neurons	$\in [20, 40, 60, 80, 120]$
layers	$\in [1, 2]$
time steps	160
training passes	20

Hyperparameters in Fig. 6.4 (right)	
window size	80
neurons	80
layers	$\in [1, 2]$
time steps	$\in [80, 160, 240, 320]$
training passes	20

APPENDIX B

Details for implementation of Cognitive Codec (Chapters 7 and 8)

B.1 Hardware and software requirements

Hardware and software used for training the proposed model is presented in Table B.1. These requirements are not strict and it is expected that the model can be trained successfully in a similar hardware or with more recent softwares.

Table B.1 Hardware and software used for training the proposed model.

Hardware requirements	
hardware	*Specification*
GPU	$(GTX\ 1080\ Ti) \times 2$
RAM	$32GB$
Software requirements	
software	*version*
PyTorch	1.9.0
CUDA	10.2
NumPy	1.17.0

B.2 Datasets and the training input

Two different datasets were used for experimental results presented in Chapters 7 and 8. First, a 100-hour subset of the LibriSpeech dataset [132] was used to train and to evaluate the performance of the proposed approach on phoneme classification and on speaker identity classification. The 100-hour subset can be accessed in Open Speech and Language Resources [1]. The forced-aligned phoneme labels for phoneme classification are the same as labels used in CPC model [125] for both train and test sets [2]. For emotional speech, the Emov-DB dataset [133] was used [3].

For minibatch optimization, in every training epoch, minibatches of random segments of speech files were selected. We used 20480-sample windows for training the encoder and 30720-sample windows for training the decoder.

1. https://www.openslr.org/resources/12/train-clean-100.tar.gz
2. https://drive.google.com/drive/folders/1BhJ2umKH3whguxMwifaKtSra0TgAbtfb
3. https://openslr.org/115/

B.3 Configuration of layers

Configuaration of the proposed network in Chapters 7 is summarized in Table B.2. A range of hidden sizes are used in the experiments of Fig. 7.2 and Fig. 7.3. The configuration of the encoder in Chapter 8 is the same as Table B.2 where the hidden size of GRUs are 64. The configuration of the decoder is presented in Table 8.1.

Table B.2 Hyperparameters of encoder network. Lower and top level refer to blocks as they appear in Fig.7.1.

Encoder layers		
lower level	*filter size*	[10, 8, 4, 4, 4]
	downsample	[5, 4, 2, 2, 2]
	GRU hidden size	$\in [8, 16, 32, 64, 128, 256]$
top level	*filter size*	[4, 4, 4]
	downsample	[2, 2, 2]
	GRU hidden size	$\in [8, 16, 32, 64, 128, 256]$

B.4 Initialization of weights

Usually, small random values are used to define the starting point for the optimization for training a neural network model. For initialization of weights in the encoder structure, we used the method described in [147], using a normal distribution with *fan out* mode [4]. The weights in the decoder architecture are initialized with a normal distribution where the mean value and the standard deviation are 0 and 0.01 respectively.

Table B.3 Training hyperparameters

learning rate	$2.0e-4$
prediction steps	12
negative samples	10
batch size	8
number of iterations(encoder)	300
number of iterations(decoder)	900

B.5 Training

B.5.1 Training hyperparameters

The hyperparameters for training the proposed networks in Chapters 7 and 8 are summarized in Table B.3. For training the encoder, *prediction steps* indicates that we predict contents of frames in speech signal for 1-12 steps for both short-term and long-term attributes. The parameter *negative samples* shows that for every positive samples of short and long frames (which are from distributions $p(x_s(t+k)|c_s(t), c_l(t))$ and $p(x_l(t+k)|c_s(t), c_l(t))$ respectively), 10 negative samples for short and long frames (which are from distributions $p(x_s(t+k))$ and $p(x_l(t+k))$ respectively) are used. Finally, when training the decoder,

4. https://pytorch.org/docs/stable/nn.init.html

although 900k value for *number of epochs* is used for the experiment in Section 8.5.2, extended experiments show that the quality of speech can be further improved until 2M training epochs.

B.5.2 Training algorithms

Algorithms for training the encoder and the decoder are described in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1: Encoder training algorithm.

Require: training hyperparameters;

Require: w_0; initial weights;

foreach *number of iterations* **do**

Sample a minibatch of m samples $\{x^1, ..., x^m\}$ from speech dataset $\mathcal{D}$;

Compute the outputs of the encoder: $\{C_s^1, ..., C_s^m\}$, $\{C_l^1, ..., C_l^m\}$;

Select positive and negative samples from $\{C_s^1, ..., C_s^m\}$;

Compute $L_{Ns} = \mathbb{E}_{X_s}\left[\sum_k \left[log \frac{f_{sk}(x_s(t+k), c_s(t), c_l(t))}{\sum_{x_s(j) \in X_s} f_{sk}(x_s(j), c_s(t), c_l(t))}\right]\right]$;

Select positive and negative samples from $\{C_l^1, ..., C_l^m\}$;

Compute $L_{Nl} = \mathbb{E}_{X_l}\left[\sum_k \left[log \frac{f_{lk}(x_l(t+k), c_s(t), c_l(t))}{\sum_{x_l(j) \in X_l} f_{lk}(x_l(j), c_s(t), c_l(t))}\right]\right]$;

Update the encoder by descending its stochastic gradient:

$\nabla_w (L_{Ns} + L_{Nl})$;

end foreach

The various functions used in this algorithms are explained in Chapter 7.

Expectations are computed from positive and negative samples taken from each minibatch.

Algorithm 2: Decoder training algorithm.

Require: training hyperparameters;

Require: w_0, θ_0; initial weights of decoder (generator) and discriminators;

foreach *number of iterations* **do**

Sample a minibatch of m samples $\{x^1, ..., x^m\}$ from speech dataset $\mathcal{D}$;

Compute $\{C_s^1, ..., C_s^m\}$, $\{C_l^1, ..., C_l^m\}$; outputs of the encoder;

Quantize the outputs to compute $\{\hat{C}_s^1, ..., \hat{C}_s^m\}$, $\{\hat{C}_l^1, ..., \hat{C}_l^m\}$;

Compute loss functions:

$L_1 = \frac{1}{m}\sum_{i=1}^{m}\left[(D(x^i)-1)^2 + D(G(\hat{C}_s^i, \hat{C}_l^i))^2\right]$

$L_2 = \frac{1}{m}\sum_{i=1}^{m}\left[||\xi_s(x^i) - \xi_s(G(\hat{C}_s^i, \hat{C}_l^i))||_1\right]$

$L_3 = \frac{1}{m}\sum_{i=1}^{m}\left[||\xi_l(x^i) - \xi_l(G(\hat{C}_s^i, \hat{C}_l^i))||_1\right]$

$L_4 = \frac{1}{m}\sum_{i=1}^{m}\left[||\phi(x^i) - \phi(G(C_s^i, C_l^i))||_1\right]$

$L_5 = \frac{1}{m}\sum_{i=1}^{m}\left[\sum_{j=1}^{L}||D_j(x^i) - D_j(G(\hat{C}_s^i, \hat{C}_l^i))||_1\right]$;

Update the discriminators by ascending its stochastic gradient:

$\nabla_\theta(\sum_{i=1}^{5}[\lambda_i * L_i])$;

Update the decoder by ascending its stochastic gradient:

$\nabla_w \frac{1}{m}\sum_{i=1}^{m}\left[D(G(\hat{C}_s^m, \hat{C}_l^m) - 1)^2\right]$;

end foreach

The various functions used in this algorithms are explained in Chapter 8.

APPENDIX C

Computational Complexity of Cognitive Codec (Chapters 7 and 8)

C.1 Computational complexity of standardized codecs

The computational complexity of standardized codecs is normally expressed in weighted millions of operations per second (WMOPS). The different operations (addition, multiplication, inverse, logical test, etc.) are weighted according to their relative complexities [1]. The computational complexity of conventional speech and audio codecs typically ranges from a couple of WMOPS for older, simpler codecs up to several tens of WMOPS for more recent and elaborate codecs. For example, for the EVS codec (see Section 2.4), the worst case complexity is approximately 88 WMOPS (56 for the encoder + 32 for the decoder) [2]. For recent multi-mode codecs, computational complexity also depends on the coding mode used, thus on the audio content. Memory occupation (RAM, ROM) is also often taken into consideration when discussing complexity. However, this aspect is currently given less importance because the memory requirements are not considerable compared to the amount of available memory.

C.2 Computational complexity of Cognitive Codec

The computational complexity of neural networks is normally expressed in floating point operations (FLOPS). The methods to estimate computational complexity of the networks used in Cognitive Codec (which includes CNNs and GRUs) can be found in [148, 149]. For Cognitive Codec, estimated computational complexity is approximately 8.1 GFLOPS and 2.4 GFLOPS for the encoder and the decoder, respectively.

The speed of a neural network also depends on other factors such as the extent that operations can be performed in parallel, the memory speed, etc. To take such limits into account, it is common to measure the speed of a neural network on specific devices. Our experiments show that the Cognitive Codec can operate more than 100x faster than real-time using a GTX 1080 Ti GPU. Also, at the time of conception of this thesis, the encoder of Cognitive codec had been tested on an Apple A13 processor. At its peak performance, A13 processor is able to encode one second of speech signal in 7 milliseconds. Similar to standardized codecs, the memory occupation of Cognitive Coded also is not considerable compared to the amount of available memory. The total number of parameters is 9.8M for the encoder and 6.3M for the decoder.

1. ITU-T Recommendation G.191: https://www.itu.int/rec/T-REC-G.191-201901-S
2. https://www.etsi.org/deliver/etsi_tr/126900_126999/126952/12.04.00_60/tr_126952v120400p.pdf

C.3 Discussion

While there is a large difference between the number of computational operations required for the Cognitive Codec and the recent standardized codecs, it should be noted that the Cognitive Codec is designed to run on a different hardware. Conventional codecs have an essentially sequential architecture, where the operations must be executed one after the other. They are typically executed on a CPU. Neural networks, in contrast, are highly parallelizable and can be executed on GPUs. Similar to standardized codecs, Cognitive Codec can also run on current portable devices while only using a fraction of resources available.

LIST OF REFERENCES

[1] T. Bäckström, *Speech coding: with code-excited linear prediction*. Springer, 2017.

[2] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, and M. Dietz, "ISO/IEC MPEG-2 advanced audio coding," *Journal of the Audio engineering society*, vol. 45, no. 10, pp. 789–814, 1997.

[3] J. Herre and M. Dietz, "MPEG-4 high-efficiency aac coding [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 137–142, 2008.

[4] P. Robitaille, S. Trempe, P. Gournay, and R. Lefebvre, "On the influence of quantization on the identifiability of emotions from voice coding parameters," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5950–5954, IEEE, 2016.

[5] S. Bruhn, H. Pobloth, M. Schnell, B. Grill, J. Gibbs, L. Miao, K. Järvinen, L. Laaksonen, N. Harada, N. Naka, *et al.*, "Standardization of the new 3GPP EVS codec," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5703–5707, IEEE, 2015.

[6] M. Dietz, M. Multrus, V. Eksler, V. Malenovsky, E. Norvell, H. Pobloth, L. Miao, Z. Wang, L. Laaksonen, A. Vasilache, *et al.*, "Overview of the evs codec architecture," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5698–5702, IEEE, 2015.

[7] N. Lavan, A. M. Burton, S. K. Scott, and C. McGettigan, "Flexible voices: Identity perception from variable vocal signals," *Psychonomic bulletin & review*, vol. 26, no. 1, pp. 90–102, 2019.

[8] S. L. Heald and H. C. Nusbaum, "Talker variability in audio-visual speech perception," *Frontiers in psychology*, vol. 5, p. 698, 2014.

[9] G. Fant, "The source filter concept in voice production," *STL-QPSR*, vol. 1, no. 1981, pp. 21–37, 1981.

[10] T. Koc and T. Ciloglu, "Nonlinear interactive source-filter models for speech," *Computer Speech & Language*, vol. 36, pp. 365–394, 2016.

[11] R. Munkong and B.-H. Juang, "Auditory perception and cognition," *IEEE signal processing magazine*, vol. 25, no. 3, 2008.

[12] N. Mesgarani, C. Cheung, K. Johnson, and E. F. Chang, "Phonetic feature encoding in human superior temporal gyrus," *Science*, vol. 343, no. 6174, pp. 1006–1010, 2014.

[13] N. Mesgarani and E. F. Chang, "Selective cortical representation of attended speaker in multi-talker speech perception," *Nature*, vol. 485, no. 7397, pp. 233–236, 2012.

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[15] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[16] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *International Conference on Machine Learning*, pp. 2410–2419, PMLR, 2018.

[17] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.

[18] J. Boilard, P. Gournay, and R. Lefebvre, "A literature review of WaveNet: Theory, application, and optimization," in *Audio Engineering Society Convention 146*, Audio Engineering Society, 2019.

[19] D. Ellis, *Modeling the auditory component of speech.* Mahwah, NJ: Lawrence Erlbaum Assoc, 2006.

[20] A. M. Liberman and I. G. Mattingly, "The motor theory of speech perception revised," *Cognition*, vol. 21, no. 1, pp. 1–36, 1985.

[21] P. D. Eimas, E. R. Siqueland, P. Juscyk, and J. Vigorito, "Speech perception in infants," *Science*, vol. 171, no. 3968, pp. 303–306, 1971.

[22] B. C. Moore, *An introduction to the psychology of hearing.* Brill, 2012.

[23] C. C. Wier, W. Jesteadt, and D. M. Green, "Frequency discrimination as a function of frequency and sensation level," *The Journal of the Acoustical Society of America*, vol. 61, no. 1, pp. 178–184, 1977.

[24] J. Herre and J. D. Johnston, "Enhancing the performance of perceptual audio coders by using temporal noise shaping (TNS)," in *Audio Engineering Society Convention 101*, Audio Engineering Society, 1996.

[25] J. Herre and J. D. Johnston, "Exploiting both time and frequency structure in a system that uses an analysis/synthesis filterbank with high frequency resolution," in *Audio Engineering Society Convention 103*, Audio Engineering Society, 1997.

[26] R. V. Cox, S. F. D. C. Neto, C. Lamblin, and M. H. Sherif, "Itu-t coders for wideband, superwideband, and fullband speech communication [series editorial]," *IEEE Communications Magazine*, vol. 47, no. 10, pp. 106–109, 2009.

[27] L.-W. Chen, C.-T. Chien, Y.-H. Hsiao, W.-C. Lee, Z.-W. Li, C.-M. Liu, M.-T. Su, and C.-H. Yang, "Efficient bit reservoir design for MP3 and AAC," in *Audio Engineering Society Convention 117*, Audio Engineering Society, 2004.

[28] I. T. Union, "Methods for subjective determination of transmission quality, ITU-T recommendation."

[29] I. Recommendation, "Method for the subjective assessment of intermediate sound quality (MUSHRA)," *ITU, BS*, pp. 1543–1, 2001.

[30] I. R. Assembly, "Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," 1994.

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

[32] K. D. Anderson and P. Gournay, "Pitch resynchronization while recovering from a late frame in a predictive speech decoder," in *Ninth International Conference on Spoken Language Processing*, 2006.

[33] K. Sayood, *Introduction to data compression*. Newnes, 2012.

[34] T. ETSI, "126 445 v12. 0.0," *Universal Mobile Telecommunications System (UMTS)*, pp. 1–627.

[35] M. Jelinek and R. Salami, "Wideband speech coding advances in VMR-WB standard," *IEEE transactions on audio, speech, and language processing*, vol. 15, no. 4, pp. 1167–1179, 2007.

[36] M. Jelinek, T. Vaillancourt, and J. Gibbs, "G. 718: A new embedded speech and audio coding standard with high resilience to error-prone transmission channels," *IEEE Communications Magazine*, vol. 47, no. 10, pp. 117–123, 2009.

[37] S. Nagisetty, Z. Liu, T. Kawashima, H. Ehara, X. Zhou, B. Wang, Z. Liu, L. Miao, J. Gibbs, L. Laaksonen, *et al.*, "Low bit rate high-quality MDCT audio coding of the 3GPP EVS standard," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5883–5887, IEEE, 2015.

[38] G. Fuchs, C. R. Helmrich, G. Marković, M. Neusinger, E. Ravelli, and T. Moriya, "Low delay LPC and MDCT-based audio coding in the evs codec," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5723–5727, IEEE, 2015.

[39] M. Neuendorf, M. Multrus, N. Rettelbach, G. Fuchs, J. Robilliard, J. Lecomte, S. Wilde, S. Bayer, S. Disch, C. Helmrich, *et al.*, "The ISO/MPEG unified speech and audio coding standard—consistent high quality for all content types and at all bit rates," *Journal of the Audio Engineering Society*, vol. 61, no. 12, pp. 956–977, 2013.

[40] J. Svedberg, V. Grancharov, S. Sverrisson, E. Norvell, T. Toftgård, H. Pobloth, and S. Bruhn, "MDCT audio coding with pulse vector quantizers," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5937–5941, IEEE, 2015.

[41] Z. Wang, L. Miao, J. Gibbs, T. Toftgård, M. Sehlstedt, S. Bruhn, V. Atti, V. Rajendran, and D. Dewasurendra, "Linear prediction based comfort noise generation in the EVS codec," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5903–5907, IEEE, 2015.

[42] A. Lombard, S. Wilde, E. Ravelli, S. Döhla, G. Fuchs, and M. Dietz, "Frequency-domain comfort noise generation for discontinuous transmission in EVS," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5893–5897, IEEE, 2015.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[44] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[45] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in *Interspeech*, vol. 2013, pp. 436–440, 2013.

[46] B. Xia and C. Bao, "Speech enhancement with weighted denoising auto-encoder," in *Interspeech*, pp. 3444–3448, 2013.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[49] C. Olah, "Understanding LSTM networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Last accessed 3 AUG 2022, 2015.

[50] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[51] Q. Tian, Y. Chen, Z. Zhang, H. Lu, L. Chen, L. Xie, and S. Liu, "TFGAN: Time and frequency domain based generative adversarial network for high-fidelity speech synthesis," *arXiv preprint arXiv:2011.12206*, 2020.

[52] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," *arXiv preprint arXiv:1910.06711*, 2019.

[53] A. Mustafa, N. Pia, and G. Fuchs, "StyleMelGAN: An efficient high-fidelity adversarial vocoder with temporal adaptive normalization," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6034–6038, IEEE, 2021.

[54] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *arXiv preprint arXiv:2010.05646*, 2020.

[55] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[56] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[58] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," *Advances in neural information processing systems*, vol. 31, 2018.

[59] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," *Advances in neural information processing systems*, vol. 9, 1996.

[60] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[61] T. V. Bliss and G. L. Collingridge, "A synaptic model of memory: long-term potentiation in the hippocampus," *Nature*, vol. 361, no. 6407, pp. 31–39, 1993.

[62] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[63] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, IEEE, 2016.

[64] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[65] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[66] P. Panda and K. Roy, "Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 299–306, IEEE, 2016.

[67] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[68] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning," *Frontiers in neuroscience*, vol. 12, p. 435, 2018.

[69] W. Maass, T. Natschläger, and H. Markram, "A model for real-time computation in generic neural microcircuits," in *Advances in neural information processing systems*, pp. 213–220, 2002.

[70] W. Maass, T. Natschläger, and H. Markram, "Computational models for generic cortical microcircuits," *Computational neuroscience: A comprehensive approach*, vol. 18, p. 575, 2004.

[71] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*, pp. 1530–1538, PMLR, 2015.

[72] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[73] C. Gârbacea, A. van den Oord, Y. Li, F. S. Lim, A. Luebs, O. Vinyals, and T. C. Walters, "Low bit-rate speech coding with VQ-VAE and a WaveNet decoder," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 735–739, IEEE, 2019.

[74] J. Casebeer, V. Vale, U. Isik, J.-M. Valin, R. Giri, and A. Krishnaswamy, "Enhancing into the codec: Noise robust speech coding with vector-quantized autoencoders," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 711–715, IEEE, 2021.

[75] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[76] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[77] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[78] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in neural information processing systems*, vol. 29, 2016.

[79] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[80] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[81] "Density estimation using real NVP,"

[82] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.

[83] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621, IEEE, 2019.

[84] A. M. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studdert-Kennedy, "Perception of the speech code," *Psychological review*, vol. 74, no. 6, p. 431, 1967.

[85] K. S. Rao and S. G. Koolagudi, *Robust emotion recognition using spectral and prosodic features*. Springer Science & Business Media, 2013.

[86] M. El Ayadi, M. S. Kamel, and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition*, vol. 44, no. 3, pp. 572–587, 2011.

[87] C.-N. Anagnostopoulos, T. Iliou, and I. Giannoukos, "Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011," *Artificial Intelligence Review*, vol. 43, no. 2, pp. 155–177, 2015.

[88] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, S. Zafeiriou, *et al.*, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5200–5204, IEEE, 2016.

[89] H. Fayek, M. Lech, and L. Cavedon, "Towards real-time speech emotion recognition using deep neural networks," in *Signal Processing and Communication Systems (ICSPCS), 2015 9th International Conference on*, pp. 1–5, IEEE, 2015.

[90] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[91] G. Fant, *Acoustic theory of speech production: with calculations based on X-ray studies of Russian articulations*, vol. 2. Walter de Gruyter, 1971.

[92] T. W. Troyer and K. D. Miller, "Integrate-and-fire neurons matched to physiological fi curves yield high input sensitivity and wide dynamic range," in *Computational Neuroscience*, pp. 197–201, Springer, 1997.

[93] S. Song and L. F. Abbott, "Cortical development and remapping through spike timing-dependent plasticity," *Neuron*, vol. 32, no. 2, pp. 339–350, 2001.

[94] M. Stimberg, D. F. Goodman, V. Benichoux, and R. Brette, "Equation-oriented specification of neural models for simulations," *Frontiers in Neuroinformatics*, vol. 8, p. 6, 2014.

[95] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *Elife*, vol. 8, p. e47314, 2019.

[96] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, "A database of german emotional speech.," in *Interspeech*, vol. 5, pp. 1517–1520, 2005.

[97] Y. Jin, P. Song, W. Zheng, and L. Zhao, "A feature selection and feature fusion combination method for speaker-independent speech emotion recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4808–4812, IEEE, 2014.

[98] S. Zhang, X. Zhao, and B. Lei, "Speech emotion recognition using an enhanced kernel isomap for human-robot interaction," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[99] M. Bhargava and T. Polzehl, "Improving automatic emotion recognition from speech using rhythm and temporal feature," *arXiv preprint arXiv:1303.1761*, 2013.

[100] E. L. Choy, *Waveform interpolation speech coder at 4 kb/s*. PhD thesis, Citeseer, 1998.

[101] J. Clark, C. Yallop, and J. Fletcher, *An Introduction to Phonetics and Phonology*. Blackwell textbooks in linguistics, Blackwell Publishing, 2008.

[102] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, 1975.

[103] J. Thyssen, H. Nielsen, and S. D. Hansen, "Non-linear short-term prediction in speech coding," in *1994 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1994)*, vol. 1, pp. 185–188, IEEE, 1994.

[104] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[105] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[106] I.-T. R. G. A. I, "A high quality low complexity algorithm for packet loss concealment with G.711," 1999.

[107] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.

[108] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the 2000 IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*, vol. 3, pp. 189–194, IEEE, 2000.

[109] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[110] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[111] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, pp. 6645–6649, IEEE, 2013.

[112] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

[113] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE network*, vol. 12, no. 5, pp. 40–48, 1998.

[114] C. A. Rodbro, M. N. Murthi, S. V. Andersen, and S. H. Jensen, "Hidden markov model-based packet loss concealment for voice over IP," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1609–1623, 2006.

[115] B.-K. Lee and J.-H. Chang, "Packet loss concealment based on deep neural networks for digital speech transmission," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 2, pp. 378–387, 2016.

[116] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems*, pp. 2222–2232, 2017.

[117] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[118] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Transactions on audio, speech, and language processing*, vol. 16, no. 1, pp. 229–238, 2008.

[119] I.-T. Recommendation, "Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," *Rec. ITU-T P. 862*, 2001.

[120] J. S. Bruner, "On perceptual readiness," *Psychological review*, vol. 64, no. 2, p. 123, 1957.

[121] M. Heilbron and M. Chait, "Great expectations: is there evidence for predictive coding in auditory cortex?," *Neuroscience*, vol. 389, pp. 54–73, 2018.

[122] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[123] W. Wang, J. Shen, M.-M. Cheng, and L. Shao, "An iterative and cooperative top-down and bottom-up inference network for salient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5968–5977, 2019.

[124] H. Wen, K. Han, J. Shi, Y. Zhang, E. Culurciello, and Z. Liu, "Deep predictive coding network for object recognition," in *International Conference on Machine Learning*, pp. 5266–5275, PMLR, 2018.

[125] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[126] X. Wang, S. Takaki, J. Yamagishi, S. King, and K. Tokuda, "A vector quantized variational autoencoder (VQ-VAE) autoregressive neural F_0 model for statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 157–170, 2019.

[127] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2180–2188, 2016.

[128] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[129] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 12, pp. 2041–2053, 2019.

[130] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304, JMLR Workshop and Conference Proceedings, 2010.

[131] A. Hyvarinen and H. Morioka, "Unsupervised feature extraction by time-contrastive learning and nonlinear ica," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3765–3773, 2016.

[132] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.

[133] A. Adigwe, N. Tits, K. E. Haddad, S. Ostadabbas, and T. Dutoit, "The emotional voices database: Towards controlling the emotion dimension in voice generation systems," *arXiv preprint arXiv:1806.09514*, 2018.

[134] B. Bessette, R. Salami, R. Lefebvre, M. Jelinek, J. Rotola-Pukkila, J. Vainio, H. Mikkola, and K. Jarvinen, "The adaptive multirate wideband speech codec (AMR-WB)," *IEEE transactions on speech and audio processing*, vol. 10, no. 8, pp. 620–636, 2002.

[135] J.-M. Valin, K. Vos, and T. Terriberry, "Definition of the opus audio codec," *IETF, September*, 2012.

[136] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *International conference on machine learning*, pp. 3918–3926, PMLR, 2018.

[137] R. Lotfidereshgi and P. Gournay, "Cognitive coding of speech," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7772–7776, IEEE, 2022.

[138] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[139] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International conference on machine learning*, pp. 1558–1566, PMLR, 2016.

[140] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pp. 2528–2535, IEEE, 2010.

[141] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," *arXiv preprint arXiv:1909.11646*, 2019.

[142] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

[143] R. Lotfidereshgi and P. Gournay, "Biologically inspired speech emotion recognition," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5135–5139, IEEE, 2017.

[144] R. Lotfidereshgi and P. Gournay, "Speech prediction using an adaptive recurrent neural network with application to packet loss concealment," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5394–5398, IEEE, 2018.

[145] R. Lotfidereshgi and P. Gournay, "Practical cognitive speech compression," in *2022 IEEE Data Science and Learning Workshop (DSLW)*, pp. 1–6, 2022.

[146] V. Despotovic, N. Goertz, and Z. Peric, "Nonlinear long-term prediction of speech based on truncated volterra series," *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 3, pp. 1069–1073, 2011.

[147] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[148] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597–1600, IEEE, 2017.

[149] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.