UNIVERSITAT POLITÈCNICA DE CATALUNYA

Universitat Politècnica de Catalunya
Teoria del Senyal i Comunicacions

This thesis is submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy (PhD)

Efficient, End-to-End and Self-Supervised Methods for Speech Processing and Generation

by Santiago Pascual de la Puente

Advisor: Antonio Bonafonte Cávez
Co-advisor: Joan Serrà Julià
Barcelona, December 2019

"Out of the night that covers me,
Black as the pit from pole to pole,
I thank whatever gods may be
For my unconquerable soul.

In the fell clutch of circumstance
I have not winced nor cried aloud.
Under the bludgeonings of chance
My head is bloody, but unbowed.

Beyond this place of wrath and tears
Looms but the Horror of the shade,
And yet the menace of the years
Finds and shall find me unafraid.

It matters not how strait the gate,
How charged with punishments the scroll,
I am the master of my fate,
I am the captain of my soul. "

William E. Henley

Abstract

Deep learning has affected the speech processing and speech generation fields in many directions. First, with end-to-end architectures that allow for injection and synthesis of waveform samples directly. Secondly, with the exploration of efficient solutions that allow for deep learning to be applied, with its effectiveness, to computationally restricted environments like mobile devices. And finally, with latest trends that explore speech and audio data to learn novel representations with the least amount of supervision. In this work we explore these three directions in both speech generation and speech analysis systems. First, we make use of the latest pseudo-recurrent structures, such as self-attention models and quasi recurrent neural networks, to build text-to-speech acoustic models. The resulting quality and efficiency of the proposed systems is then studied in comparison to the recurrent baseline. Our study shows that, with the proposed pseudo-recurrent acoustic model based on quasi recurrent neural networks, named QLAD, we achieve a 11.2 times synthesis speedup on CPU and 3.3 times on GPU with respect to the purely recurrent baseline model. Moreover, such model is able to preserve the synthetic speech quality to the same level as in the original recurrent model, both competitive with state of the art in vocoder-based statistical parametric speech synthesis models. Secondly, we propose a speech enhancement generative adversarial network, the SEGAN. This is able to perform speech-to-speech mappings in one single inference as a fully convolutional structure, which also involves an efficiency increase with respect to existing end-to-end auto-regressive generative models. We show prominent results in noise removal with respect to other classic and deep regression baselines. This is first reflected in objective results, where the best performing model we propose reaches the highest quotas of segmental signal-to-noise-ratio and intelligibility scores. In subjective terms, the model also shows advantage over all other selected baselines in terms of noise removal and, also importantly, least distortion injection in the speech regeneration. The proposed model also proves to be efficient in its transferability to new languages and noises. Departing from an English pre-trained model, the system achieves comparable objective performance to the English model itself on Catalan and Korean with only 24 s of adaptation data, with unseen speakers and noise. This would specially fit the requirements of having a model for low resource environments. Finally, we unveil the generalization capability of SEGAN towards other noises and distortions, hence truly taking advantage of the generative capacity of the model with the so called generalized speech enhancement. The first task tackled under this shift towards palliating distortions on the speech signals is a whisper-to-voiced conversion. This part of the work aims to improve an existing regression based approach with recurrent networks that, for clinical purposes, tries to recover natural speech from aphonia. Results show that our model can indeed enhance the naturalness of the resulting speech in terms of intonation, as well as proving the need for the adversarial training component upon the failure of

a regression-based method in time domain. Then, to finish our proposal on the use of SEGAN for generalized speech enhancement, a new paradigm shift is proposed where several distortions are combined and have to be palliated. This involves a new trend where deep generative models are to be used for general purpose speech enhancement, where reconstructed naturally sounding utterances have to emerge out of damaged ones even if it involves a change of prosody and low level features with respect to the original signal. To make our SEGAN effective on the first approach we take for this task, we propose the insertion of perceptually related losses to build a multi-task setup in the GAN discriminator, as well as a two-stage training strategy. Acoustic features are then predicted in the deepest part of the classifier, which enriches the features learned in the adversarial training, which in turn makes the generator yields better quality for the regenerated speech. And secondly, the two-stage training is proposed to keep the equilibrium in the synergy between both GAN networks. With these two additions objective and subjective results show a substantial increase of quality of our proposed GAN compared to its vanilla version. Finally, we propose a problem-agnostic speech encoder, named PASE, as well as a training framework for it that makes it work unsupervisedly. PASE is a fully convolutional structure that yields compact embeddings, which contain highly abstract features such as speaker identity, prosodic traits and spoken contents. We also propose a novel self-supervised multi-task learning approach in order to train the encoder. Training without requiring manual labeling makes it specially attractive for exportability purposes to whichever dataset and to leverage large amounts of data. Once PASE is trained during the self-supervised phase, it is exported into other tasks and datasets to assess its transferability and the quality of its learned representation. The encoder proves to be effective to do speaker, emotion and speech recognition. Specifically, if we are able to fine-tune it with the end-task classifier in the target dataset, the representation quality increases and systematically surpasses well tuned classic features such as MFCCs and FBANKs, which are extensively used in current state of the art speaker recognition and speech recognition systems, as well as itself without fine-tuning. Interestingly, we also show that this holds for noisy acoustic conditions to do speech recognition, even if the encoder was trained only on clean data as it is the case. Moreover, the self-supervised multi-task components importance is evaluated in an ablation study, which shows that all the proposed tasks matter across the three evaluated problems. Finally, we provide preliminary results on the effectiveness of the learned embeddings for text-to-speech. In this case the encoder is used as an acoustic feature descriptor for speaker identities, where the yielded frames for an utterance are averaged to obtain the global representation. Objective results show faster convergence and higher speech quality when using our embeddings than with typical trainable one-hot discrete codes. Additionally, using an acoustic feature extractor as descriptor allows for out-of-corpus speech identities to be synthesized, hence doing speaker adaptation without the need for retraining. We also show our first objective results that support that we can indeed perform this speaker adaptation, specially when we have enough samples of the target speaker.

Acknowledgements

En aquesta tesi hi ha recopilades moltes pàgines de propostes, dissenys i resultats. Però a part d'això, i de manera no palpable ni fàcilment apreciable a la vista d'aquests fulls, hi ha un creixement personal important per part meva. Ha estat un camí llarg, dut endavant per il·lusions alimentades de curiositat. I dins la frustració que genera no obtenir el que un espera més sovint del necessari, aquí esta el resultat de la perseverança i la paciència per aconseguir quelcom del que quedar satisfet. Nogensmenys, un no sol suportar les frustracions sense companyia en la que recolzar-se i amb qui sentir que les coses milloraràn. Tampoc fora possible obtenir un treball de qualitat amb diversitat de propostes i resultats sense col·laboracions més que enriquidores. Així doncs, hi ha un gran col·lectiu de gent a qui agraïr el suport i la feina feta.

Primer de tot, tirar endavant la feina esmentada no seria possible sense una mentoria adeqüada. Així doncs, vull agraïr enormement als meus dos directors de tesi, l'Antonio Bonafonte i el Joan Serrà, la dedicació que han tingut envers a mi i la meva formació. El seu recolzament constant a les meves propostes m'ha ajudat envers el desenvolupament de noves idees. Els agraeixo també els esforços que han fet per aconseguir-me els recursos necessaris per dur a terme recerca cada cop més exigent, especialment en termes computacionals. És important remarcar que si em sento més científic, tant en saber proposar innovacions, com en el desenvolupament experimental i l'escriptura, és gràcies als seus consells i la seva incansable guia.

También agradezco la dedicación de mi tutora, Asunción Moreno, para escuchar y darme consejos de escritura, y por brindarme su inestimable ayuda con los sortilegios burocráticos.

Agradezco a mi família primero de todo su cariño, y que me hayan apoyado en los momentos difíciles. También les agradezco enormemente que siempre me hayan empujado a estudiar y construirme un futuro donde la educación fuera un pilar fundamental. Es para mi una gran satisfacción alimentar su orgullo con el hecho de culminar mi educación superior como doctor. Además, debo agradecer de manera especial a mi madre, Maria Angeles, su eterno esfuerzo en trabajar sin reparo para que sus hijos tuvieramos oportunidades y buen ejemplo. Als meus germans, Toni i Jordi, els transmeto un especial agraïment per fer-me seguir els bons passos que m'obligàven a formar-me i no abandonar. També agraeixo al pare, l'Antonio, el fet que des de petit m'aportés reflexió i afecció pel coneixement.

Vull agraïr a la Cristina el suport que m'ha donat en tot moment per calmar els ànims quan els resultats eren adversos i els processos de la recerca generàven ansietat. També per alegrar-se de cada petita o gran victòria que pogués publicar i celebrar, i per fer-me sentir que esta orgullosa del que faig mentres avanço per aquest camí trontollós que es diu recerca. També agraïr a la seva familia el resguard de les escapades al poble amb les que alliberar els pensaments i respirar aire fresc.

També vull agraïr als meus companys i professors els moments d'entreteniment que m'han brindat, així com els coneixements que haguem pogut compartir al llarg del desenvolupament de la recerca que es presenta aqui. Gràcies a l'Amaia Salvador,

el Xavi Giró, el Miquel India, l'Enric Monte, el Jose Adrián Rodríguez Fonollosa, el Jordi Luque, el Carlos Segura, el Ferran Diego Andilla i el Jordi Pons.

També agraeixo els bons moments que els meus amics m'han brindat, per totes les vegades que em puc escapar per veure'ls, que no són tantes com esperaria, i lliurar-me del *mare magnum* de divagacions científiques que tant absorbeixen. Gràcies especialment al Carles Valencia, el Nacho Ripoll, la Carla Hortigüela, el Pedro Yuste, el Maties Pons, la Carmen Quintana i el Rafa Valencia.

I also want to acknowledge the efforts and work by my close collaborators, which participated in certain parts of this extense work. I want to thank Maruchan Park and Prof. Kang-Hun Ahn for their work on SEGAN adaptation to Korean, and for the excellent time I had getting to know Korea. A piece of the "Pali pali" culture remains in me. I also thank José A. González for his work on whispered-to-voice conversion with SEGAN, and for facilitating us the data. I also thank Mirco Ravanelli and Yoshua Bengio for their work and collaboration to make PASE, their fantastic ideas to improve it and its current development. Also in this line of research, I want to thank Pawel Swietojanski for our insightful conversations on many topics related to speech processing and generation in the end of my PhD, in addition to his prominent work with PASE for ASR. I also want to thank Jianyuan Zhong and Joao Monteiro for their work on improving PASE and working so hard on refactoring the open source framework. I also thank Maurizio Omologo and the JSALT 2019 oragnizers for gathering such great team with which I could learn so much about a broad set of concepts related to speech recognition, deep architectures and learning strategies. I also thank David Álvarez for his experimentation on merging PASE features with the SampleRNN acoustic model.

I want to deeply thank all those people that took the several subjective evaluations with patience. The satisfaction of successful results coming out of this work would not have happened without their willingness to help. Also, special thanks to all the reviewers, both for the publications that are part of the development of this thesis and for the thesis itself. They really helped me to substantially improve the quality of my manuscripts.

I acknowledge that this thesis would not have been possible without the financial assistance of the VEU group at UPC and the Spanish Government through the FPI grant and the project TEC2015-69266-P (MINECO/FEDER, UE).

Contents

List of Figures

List of Tables

List of Abbreviations

ASR	**A**utomatic **S**peech **R**ecognition
DNN	**D**eep **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
HMM	**H**idden **M**arkov **M**odel
HNM	**H**armonics plus **N**oise **M**odel
IoT	**I**nternet **o**f **T**hings
IPA	**I**ntelligent **P**ersonal **A**ssistant
LPS	**L**og **P**ower **S**pectrum
LSTM	**L**ong **S**hort **T**erm **M**emory
MCD	**M**el **C**epstral **D**istortion
MDN	**M**ixture **D**ensity **N**etwork
MFCC	**M**el **F**requency **C**epstral **C**oefficients
MCEP	**M**el **C**epstral **C**oefficients
MHA	**M**ulti **H**ead **A**ttention
MLP	**M**ulti **L**ayer **P**erceptron
NLP	**N**atural **L**anguage **P**rocessing
PASE	**P**roblem **A**gnostic **S**peech **E**ncoder
PDF	**P**robability **D**ensity **F**unction
PReLU	**P**arametric **Re**ctified **L**inear **U**nit
QRNN	**Q**uasi **R**ecurrent **N**eural **N**etwork
RMSE	**R**oot **M**ean **S**quared **E**rror
RNADE	**R**eal-valued **N**eural **A**utoregressive **D**ensity **E**stimator
RNN	**R**ecurrent **N**eural **N**etwork
SEGAN	**S**peech **E**nhancement **G**enerative **A**dversarial **N**etwork
SPSS	**S**tatistical **P**arametric **S**peech **S**ynthesis
STS	**S**peech **T**o **S**peech
TTS	**T**ext **T**o **S**peech
US	**U**nit **S**election

Chapter 1

Introduction

1.1 Motivation

Speech is the most prominent way of exchanging messages between human beings. It allows us to be expressive as well as communicative, thus transmitting a message and, at the same time, conveying other information alongside linguistics that enrich the message. This is related to para-linguistics, which can be understood as everything that can be found in the speech signal that cannot be described only in strictly phonetic and/or linguistic terms (Schuller et al., 2013). Examples of acoustic phenomena embedded in speech, which are non related or loosely related to linguistics could be coughs, laughter or filled pauses. They can denote (health) state, emotion/mood, speaker idiosyncrasies, and the like. Moreover, high pitch can be an indication of anxiety and breathy voice can indicate attractivity, and they are also modulated onto the verbal message (Schuller et al., 2013).

Such richness makes speech signals to be present in a myriad of situations related to communications (e.g. phone, television, etc., and consequently in multimedia systems too). As an inheritance of human to human interaction, it has become a natural way of communication with an automated system like an intelligent personal assistant (IPA), either in the form of a software application in the mobile phone or a hardware appliance like a smart speaker. We refer to human-to-human communication as the process happening when a message is transmitted/received between human beings, regardless of the medium. It can hence happen through a telephonic channel or in a meeting room. In contrast, we refer to machine-to-human communication as the process happening when a message is transmitted/received between a human being and a machine. Current smart speakers feature voice-activated digital assistants and often operate as home automation hubs. These devices are usually part of a company's existing product stack[1]. Precisely, the introduction of these assistants, like Amazon Alexa, Google Home or Apple Siri, is a perfect match with the internet of thinsg (IoT) concept. We can play music, order products or interact with other home appliances thanks to the massive interconnection of IPAs with other devices. Actually projections stipulate that the smart speaker market is in the middle of a major growth cycle, and that the introduction of 5G mobile connectivity, with a massive data transfer capacity, will allow the IPAs market to grow even faster than now[2] [3].

Both scenarios, one where humans communicate amongst themselves and another with human-machine interactions, demand algorithms that process these signals to extract information from them, enhance them or even mimic their generation process to make machines produce speech. These processing techniques allow us

[1] https://whatis.techtarget.com/definition/smart-speaker
[2] https://www.alliedmarketresearch.com/smart-speaker-market
[3] https://www.visualcapitalist.com/smart-speaker-market-share-fight/

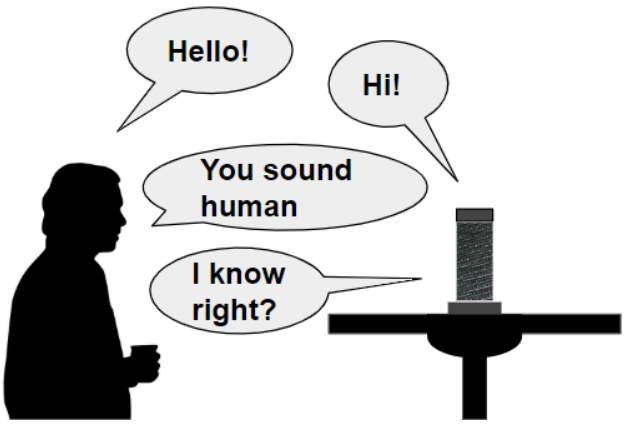

FIGURE 1.1: Example communication between a person and a smart speaker.

to communicate in any condition with other people or machines, even if such condition is adverse. This means we might be contacting someone through a phone call from a noisy environment or we may ask commands to an IPA in a real home environment, hence doing distant automatic speech recognition (ASR; Barker et al., 2018). But not only distant recordings can be troublesome to recognize the linguistic contents. Motor impairments may provoke a loss of intelligibility and naturalness in some people's speech too, so recomposing a signal with better quality with the aid of some device might be helpful in that situation.

An important component of modern technologies is the constant connectivity of the communication devices with processing machines, which is a concept aligned with the aforementioned IoT. These machines can in turn do data crunching on portions of conversations. For instance, in the case of IPAs, this process often aims to improve even further the future response of these devices, due to the fact that there is a feedback loop between the processing algorithms and the new incoming data to improve the algorithm itself. However, in order for the algorithm to properly take advantage of the massive data generated, we need to leverage the potential of algorithms that learn automatically from data.

Deep learning is a framework of algorithms and tools to learn automatic patterns from data, hence it forms a subset within machine learning techniques (Goodfellow et al., 2016). The deep learning popularity has grown exponentially during the latest years, as it offers complex statistical models that usually outperform previously existing shallower and less computationally demanding ones. The term *deep* describes the fact that these models are built as multiple levels of a hierarchy that breaks down an abstract problem (e.g. classification of audio events) into sub-levels of features and transformations. And usually the performance of our models raise with the increase of hierarchy levels. This implies that not only the input-output mapping is learned in the hierarchy of concepts, but also the feature extraction processes. In order for these models to work effectively though, massive amounts of data are necessary, specially in end-to-end scenarios, which means that signals are directly injected as raw data. This is contrary to previous (and normally shallower) machine learning solutions, where a key factor was the so called feature engineering or feature design (Huang et al., 2001; Goodfellow et al., 2016). The end-to-end strategy has been successful in fields like computer vision, where current visual models work with raw pixel data to solve several tasks; including classification, semantic segmentation, object recognition, and image generation among others (Krizhevsky et al., 2012; He et al., 2016; Garcia-Garcia et al., 2017; Zhao et al., 2019; Radford et al.,

2016; Brock et al., 2019). Natural language processing (NLP) is another field that currently tackles its most challenging tasks by directly processing words, characters or byte-pair encodings (Sennrich et al., 2016). These raw textual tokens are encoded with simple discrete identifiers and fed to deep models to get state of the art in language modeling, machine translation, and natural language understanding among others (Yang et al., 2019; Devlin et al., 2019; Vaswani et al., 2017; Young et al., 2018).

Regardless of the application field, in order to work effectively with deep models, massive amounts of data are necessary. And this need specially grows with the depth and complexity of the model. Precisely, the first great advantage nowadays to improve the speech processing pipelines is the availability of massive amounts of data to exploit with these automatized learning algorithms. Secondly, the spread of the GPU component and the democratization of its usage for high-end computations with open source platforms like PyTorch (Paszke et al., 2017) simplifies the building process of these complex models for everyone[4].

With the advent of deep learning, speech technologies are suffering a drift into the end-to-end paradigm, similarly to the aforementioned computer vision and NLP cases. This implies that the deep model is exposed to large amounts of raw speech data, with the least possible signal pre-processing or prior knowledge about the structure of speech. The model is then trained to solve a certain task by consuming the data in a state as raw as possible, such that the model itself can learn the best intermediate features that lead to the optimum response. This trend is gaining popularity across the different disciplines of speech technologies, and end-to-end modelling usually satisfies to raise state-of-the-art results across many tasks. Examples of these tasks are automatic speech recognition (ASR) (Graves and Jaitly, 2014), speech synthesis (van den Oord et al., 2016b), speech enhancement (Pascual et al., 2019b), spoken emotion recognition (Trigeorgis et al., 2016) or speaker recognition (Snyder et al., 2018). Nonetheless, these breakthroughs usually happen when the amount of computation time and data have been enough to fulfil the search of the optimum parameters for our algorithms.

As mentioned, this end-to-end scheme contrasts with shallower solutions that rely on feature engineering. Fig. 1.2 depicts simplified schematics of substitution of classic pipelines for end-to-end ones for text-to-speech (TTS) and speech enhancement tasks. Classic pipelines usually require important notions and prior knowledge about signal processing, as well as a clear conception of what features from the input signal are important to derive a proper response at the model output. We can exemplify the importance of feature selection by contrasting the classic feature selection used in speech recognition and emotion recognition. The former one works effectively upon mel frequency cepstral coefficients or filter banks (Hinton et al., 2012), whereas emotion recognition typically relies on additional prosodic features like pitch contours, energy, or voicing factors, among others (El Ayadi et al., 2011). Hence models based on feature engineering may pose three main possible drawbacks in the resulting performance: sub-optimal feature selection, cumulative errors in cascaded systems, and more complex processing pipelines.

1.1.1 Sub-optimal Feature Selection

The feature engineering process may lead to representation limitations for the algorithms. This means that we might be imposing an initial bias in the outcomes by discarding input information. So there might be still exploitable features hidden in

[4]https://missinglink.ai/guides/computer-vision/complete-guide-deep-learning-gpus/

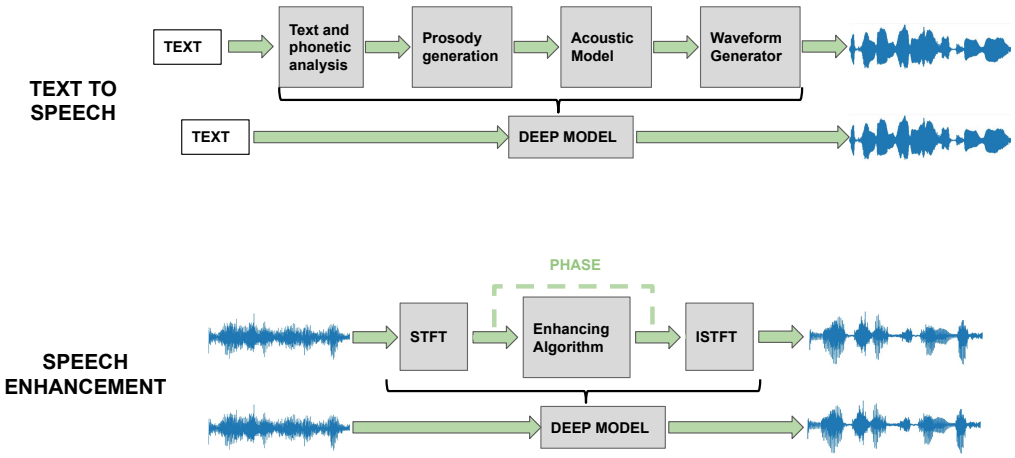

FIGURE 1.2: Example end-to-end replacements for text-to-speech and speech enhancement applications.

the raw data that we do not consider as important, perhaps because they were never studied within our field of interest. This can lead to sub-optimal results in the end. A clear example of this happens in TTS. The vocoding process, widely used in old statistical systems, discards information of the original speech waveform, hence biasing the resulting quality to a worse level even in the real data, where the model does not take effect yet. Fig. 1.3 depicts the results of subjective evaluations taken under MUSHRA tests for different TTS systems from Pascual (2016). There it is shown that the vocoder system, labelled as Ahocoder, has a degraded quality with respect to the natural speech, with an average score of approximately 82.7 with respect to the 97.7 achieved by natural speech.

1.1.2 Cumulative Errors in Cascaded Systems

The traditional pipelines are built with different specialized blocks that are trained separately, and cascaded afterwards. For instance we may have four main blocks in a TTS pipeline: (1) Text processing and linguistic feature extraction, (2) Prosodic feature prediction, (3) acoustic feature prediction, and finally (3) Waveform generation. Each block can be deterministic or learned with a statistical model, but these blocks are trained independently, which can make it unaware of possible mistakes made at the beginning of the whole process. Developing an end-to-end approach tunes every block for the final task jointly, hence it is less prone to cumulative errors that may be present in classic pipelines. Following the TTS example, the advances of the latest years shifted from separated blocks cascaded after training (Arik et al., 2017; Sotelo et al., 2017), until almost end-to-end models like Tacotron where only the final acoustic mapping is decoupled for computational limitations (Wang et al., 2017). Even when Tacotron is not fully end-to-end (it does not produce waveforms, which are the end goal of speech synthesis) it is still considered to be an end-to-end system, due to the fact that it achieves a direct mapping between raw textual features and an acoustic representation without any break down of sub-blocks that connect the

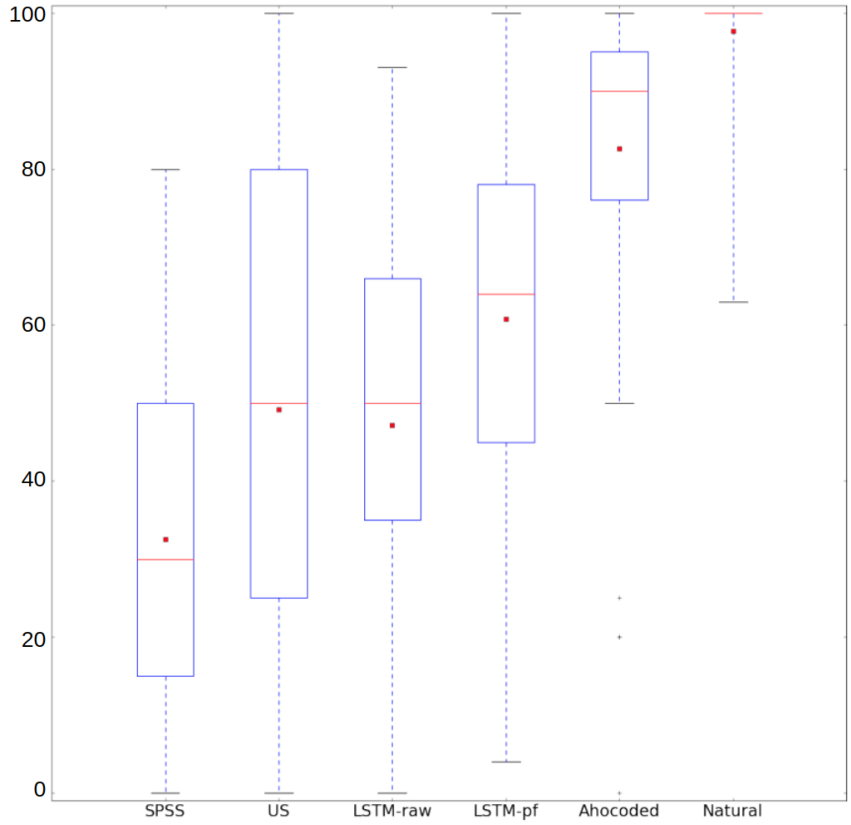

FIGURE 1.3: MUSHRA results from different speech synthesis models from Pascual (2016) Fig. 4.21. SPSS: HMM-based Statistical-Parametric Speech Synthesis (see section 2.1.2). US: Unit Selection Concatenative Speech Synthesis (see section 2.1.1). LSTM-pf and LSTM-raw: SPSS models based on recurrent neural networks with and without post-filtering respectively. Ahocoded: vocoded speech. Natural: raw recordings.

two domains. Hence in the end-to-end case the intermediate linguistic representations, the alignment between linguistic and acoustic features, the required prosodic traits and the final acoustic features are learned altogether without intermediate labeling per sub-task. The Tacotron 2 demo page[5] shows several examples of a TTS that captures a broad generalization of vocabulary, connected to appropriate pronunciations and prosodic traits that can be intuitively extracted from raw text, like intonation stress with capital letters, or the change from a statement into a question after a question mark.

1.1.3 Processing Pipeline Complexity

An end-to-end scheme can simplify the construction of models that solve complex tasks, as they do not require breaking down the big problem to be solved into smaller ones that are cascaded. It can even imply, as mentioned earlier, less need for prior knowledge in the design of the modeling pipeline. Hence, it can be a better fit for application/production purposes, as a single model is deployed to perform all the intermediate tasks, from the raw input until the expected output. This may also simplify the documentation and usability of our deployed system, due to the reduction of blocks to be used and interactions among them. However, this comes at a cost too, which is a loss in the interpretability of the model and the controllability

[5] https://google.github.io/tacotron/publications/tacotron2/index.html

of its outcomes. Evidently this happens because the model is more opaque and the optimization process drives its decisions, which cannot be interpreted as responses to sub-tasks that we would usually solve after breaking down the problem. Again, the TTS research field has quickly tackled the end-to-end transition during the last 3 years, as in the aforementioned Tacotron case (Wang et al., 2017). However, after the success of end-to-end TTS, the pursue of a better controllability on the final result still remains, with recent studies on prosodic and expressiveness transfer that can be injected into Tacotron-like systems for instance (Wang et al., 2018; Kenter et al., 2019).

1.1.4 The Deep Paradigm Shift in Speech Processing

We are in the middle of a paradigm shift in many fields related to signal processing, and one of them is speech processing. Going end-to-end can be advantageous in terms of performance and simplicity, as long as we have enough data and computational resources to train them. It is evident that end-to-end systems, as they have to integrate many different stages within one structure, have to be deep and complex although monolithic. Nonetheless, leveraging our prior knowledge on the structure of speech can also be an effective strategy, as it still maintains the controllability over our model outcomes. We pragmatically use the term "prior knowledge" to refer to classic speech feature extraction procedures, like the mel-frequency cepstral coefficients (MFCCs).

Deep generative speech models are those that mimic the real generation process that composes speech signals. They have been often used in the speech synthesis and TTS field for years (Taylor, 2009). The end-to-end shift is particularly significant in this field, as deep models like the WaveNet (van den Oord et al., 2016b) proved to suffice to directly generate waveform samples. This pushed the naturalness boundary beyond existing feature-based approaches by far. Moreover, other than text-conditioned generative models as in TTS, speech-conditioned generative models are gaining momentum too, specially thanks to this end-to-end paradigm shift. These models are applied to speech enhancement (Pascual et al., 2019b) and voice conversion (Serrà et al., 2019; van den Oord and Vinyals, 2017) with different types of generative models applied directly on the raw waveform too.

1.2 Deep Speech Representation and Generation

Speech signals are present in many types of scenarios related to human-to-human and machine-to-human communications. Evidently the machine must convey some form of speech interfacing such that it can "listen" the human enquiries, and then "speak" in order to transmit the responses. Similarly, if the human-to-human communication has to be interpreted by a machine to enhance the signal quality or transcribe the conversation as well as recognizing who speaks, some acoustic representation has to allow it.

In any type of communication, the flow of speech is either categorized (linguistic contents, speaker identity, emotion, etc.), or generated. A particular case of generation is regeneration, in case of enhancing some qualities of corrupted captured speech. Regeneration can be inserted as a post-processing stage of either receiving or emitting side. Hence in general terms for both communication devices and humans, we first have the recognition interface as the receiver, and the synthesis interface as an emitting process. In the middle of both interfaces there are other

components related to language understanding and intent recognition. In the case of humans, this language understanding is naturally available if the receiver understands the emitter language. In the case of machines, this obviously requires systems that map the captured linguistic spoken units from an ASR system into some conceptual categories that involve actions and responses. This process is called spoken language understanding (Tur and De Mori, 2011). But aside from the intrinsics of the language understanding core, and regardless of whether communication happens between human beings or a machine also interacts, the speech interfaces are critical components to establish proper interactions whenever some communication tool is used to carry the messages (e.g. phone, capturing device, etc.). In the following the speech representations concept will be described, as well as the importance of building compact and high-level speech descriptors to solve many tasks where speech is the input signal. Then, a brief description of speech generation is given, and its applicability to different applications where speech is synthesized out of different source conditioning types.

1.2.1 Speech Representations

Firstly, if we communicate a message and we build an automatized and abstract representation of the speech, it can be used to recognize patterns or even reconstruct damaged parts of the acoustics if it is rich enough. The patterns can comprise the uttered content, the speaker identity, and the emotion embedded in the speech. Importantly, these characteristics should be extracted also in adverse conditions. So the representation should be robust either in the presence of background noise, reverberation or pathologically damaged speech. Hence it is our interest to have a representation versatile enough to cover a broad spectrum of qualities about the speech as the described ones, as we can then couple in a single model different systems that are usually built and trained separately. We could think that the waveform already contains all these qualities, however a more abstract representation would ideally disentangle them, yielding frames that contain the different aforementioned factors encoded in different dimensions with a reduced intersection amongst them.

The advantage of merging representations and tasks has been already discussed previously to avoid high levels of complexity in the processing pipeline, including the avoidance of cumulative errors in the cascade of separate blocks. In fact, speech recognition systems can improve their performance by conditioning information about the acoustic environment or the speaker identity (Karafiát et al., 2011; Saon et al., 2013) by means of i-vector representations (Dehak et al., 2010). Similarly, phonetic and speaker variabilities affect the performance of speech emotion recognition systems, although the latter seem to be more critical (Sethu et al., 2008). As such, conditioning on speaker identity cues helps improving the performance of emotion recognition systems, to filter out the large acoustic variabilities among speakers (Kim et al., 2011; Ding et al., 2012).

Robust speech representations can even go beyond detecting those signal qualities explicitly, and use the mixture of the aforementioned signal qualities implicitly in an end-to-end fashion to regenerate other versions of the utterance. For instance enhanced versions of the speech can be generated (Pascual et al., 2017; Pascual et al., 2019b), or the voice identity can be converted between two speakers by learning to factorize the signal qualities and recomposing at inference with changed identities (van den Oord and Vinyals, 2017; Serrà et al., 2019). Besides all the applications

to speech tasks themselves, current deep representations can even drive the generation of face images that match the voices in the supplied utterances (Duarte et al., 2019; Oh et al., 2019).

1.2.2 Speech Generation

Speech generation, also known as speech synthesis, refers to the computerized generation of speech. This may happen after conditioning the generation on some additional source of information. TTS, for instance, is the method that generates the speech out of textual information (Taylor, 2009). Hence when a user types text, the computer interprets the linguistic contents and generates a speech waveform matching them. Another possible source of information could be a speech signal itself. We could name this approach as speech-to-speech (STS). Speech enhancement is the task where a corrupted signal, either by additive noise or reverberation, is cleaned up and restored to be more natural and intelligible (Loizou, 2013). This application then follows the STS paradigm, where an input signal is modified and reconstructed in the output with different components. The TTS research field has been rapidly moving after the first applications of deep models in acoustic models (Zen and Senior, 2014). Nevertheless, with the introduction of the WaveNet, as mentioned earlier, and also thanks to the sequence-to-sequence structures (Sutskever et al., 2014), the end-to-end model pursue accelerated exponentially in this field. As such, many very deep models have recently been proposed with the availability of large amounts of speech and text data, as well as powerful computational resources to train them. Examples of these models are the different Tacotron versions (Wang et al., 2017; Wang et al., 2018; Shen et al., 2018), the different DeepVoice versions (Arik et al., 2017; Gibiansky et al., 2017; Ping et al., 2018), the Char2Wav (Sotelo et al., 2017), the Transformer TTS (Li et al., 2019) or the ClariNet (Ping et al., 2019). These models are expensive to train, and as such their construction is often carried by the current big companies leading the deep learning research. On the other hand, however, there is hesitation on making speech generation models efficient too. The WaveNet was the proof that the waveform sampling process can be learned with current technologies and achieves the best quotas of synthetic speech quality (comparable to real speech), but it is known that it is an expensive model to sample from (Kalchbrenner et al., 2018). To that end, several models pursue improving the waveform generation efficiency, either working with a simple autoregressive parameterization than that of the WaveNet (Valin and Skoglund, 2019; Jin et al., 2018; Mehri et al., 2016; Kalchbrenner et al., 2018), or with other parallel approaches (van den Oord et al., 2017; Donahue et al., 2019). Hence, efficiency is an important matter in speech generation applications, pursued specially due to the need to embed these systems within the user devices for rapid and personalized responses (like IPAs).

Regarding speech enhancement, its main applications are usually related to improving the quality of communications in noisy environments. However, we also find applications related to hearing aids and cochlear implants, where enhancing the signal before amplification can significantly reduce discomfort and increase intelligibility (Yang and Fu, 2005). Furthermore, speech enhancement is also applied as a pre-processing stage in speech recognition and speaker identification systems, usually to denoise or dereverberate the recorded signals (Ortega-Garcia and Gonzalez-Rodriguez, 1996; Yu et al., 2008; Maas et al., 2012; Donahue et al., 2018). Beyond removing noisy conditions, speech enhancement may also refer to the recovery of missing signal components, which in turn aims to improve the signal naturalness

and expressiveness. For instance, whispered speech refers to a form of spoken communication in which the vocal folds do not vibrate and, therefore, there is no periodic glottal excitation. This can be intentional (e.g., speaking in whispers), or as a result of disease or trauma (e.g., patients suffering from aphonia after a total laryngectomy). The lack of pitch reduces the expressiveness and naturalness of the voice. Moreover, it can be a serious impediment for speech intelligibility in tonal languages (Chen et al., 2018) or in the presence of other interfering sources (i.e., cocktail party problem (Popham et al., 2018)). The conversion from whispered to voiced speech (we refer to it as dewhispering), either by reconstructing partially existent pitch contours or by generating completely new ones, is an area of research that not only has relevant practical and real-world applications, but also fosters the development of advanced speech conversion systems (Pascual et al., 2018b). Speech dewhispering turns out to be an example of the need of speech enhancement in a distortion agnostic framework, which may be plausible to achieve with the current deep learning techniques and with enough data. So generalizing the speech enhancement concept to palliate with any speech distortion provoked at any point of any speech processing pipeline is an interesting matter. To exemplify possible distortions, we may have low-rate speech coding losses, signal amplitude clippings or even sample loss (unexpected zeroed out values in the signal).

1.3 Objectives and Outline of the Thesis

This thesis has 3 goals related to the application of deep architectures in different facets of speech generation and speech processing: (1) improving efficiency of recurrent speech generation models, (2) exploring the applicability of generative adversarial networks (GANs) to make a non-autoregressive end-to-end STS system, and (3) investigate the use of self-supervision to do a generalizable speech encoder that can be applied to a myriad of speech processing tasks. Throughout the development of the end-to-end speech generation GAN and the self-supervised modules, efficiency is never abandoned to restrict the design of the proposed models.

Therefore, we begin working on the modeling efficiency concept in terms of synthesis efficiency of state of the art pseudo recurrent structures in TTS. Owing to the recent trend of end-to-end TTS, where computationally expensive models are built, this is aligned with the aforementioned recent advances in efficient designs in neural vocoders (Kalchbrenner et al., 2018; Valin and Skoglund, 2019; Jin et al., 2018; Mehri et al., 2016). We explore the use of pseudo-recurrent structures such as self-attention and quasi RNNs (see chapter 3), that leverage the sequential dependencies of the signals similarly to RNNs, hence obtaining similar synthetic speech quality while increasing parallelization.

Secondly, we tackle a speech-to-speech problem with the first deep generative end-to-end design of its kind, working at the waveform level and built as a generative adversarial network. The proposed system is applied to speech enhancement. With this design, our study makes a shift towards end-to-end deep generative modeling for speech generation. We focus on STS to apply our model, and we design it with little prior knowledge apart from its convolutional structure. Then, its limits are explored by augmenting the number of transformations to be carried or by adapting to new languages and domains. Nonetheless, we show that even when the plain model is designed with the least prior knowledge in terms of signal description, using some speech processing enhances the model design to guide its learning strategy and perceptually weight it, specially when several distortions have to be

palliated. Interestingly, this shows that end-to-end models for STS, where an input waveform is processed to output the resulting waveform, benefit from using extra signals driven by prior knowledge, like the acoustic losses we attach to our model's discriminator. This indicates that we can potentially obtain a rich set of features from the waveform through deep models, specially if we provide learning signals indicating that the well known speech features like mel-frequency cepstral coefficients are present in our signal. This is connected to the last exploration we do in this thesis, as the use of auxiliary signals with which the model predicts transformed versions of the input signal is denominated self-supervised learning, a frorm of unsupervised learning.

Therefore, in the third part of this thesis we face the challenges of applying unsupervised learning to discover speech representations that can be useful for many end-tasks that require speech as input. As introduced, speech processing prior knowledge is leveraged from the beginning, but still allowing the model to do a full exploration of the raw data (hence injecting waveforms in the input). An important characteristic of such representation is that it must keep information of different high-level features of speech (i.e. linguistic contents, speaker identity and prosody), but without the need for any label. The main objective of building such a system is its potential exportability to any task where novel speech feature extraction models might be helpful to begin the development. To achieve this, we focus on simple yet powerful approaches to learn such a versatile representation without the need of any hand-labeling.

In chapter 2 we proceed with a comprehensive literature review for speech generation, speech enhancement and speech unsupervised learning. The speech generation part, developed in section 2.1, comprises a study of speech synthesis as a paradigm where textual and spoken inputs lead to a synthetic speech signal. First, concatenative speech synthesis is reviewed in section 2.1.1. After briefly reviewing its strengths and weaknesses, the statistical parametric approach is described in section 2.1.2. This review begins with the need to overcome the limitations of concatenative synthesis in terms of adaptability and scalability. Then a thorough study is made to follow the path that leads to the current deep models, which successfully generate the highest level of synthetic speech quality with the greatest degree of adaptability to new speaker conditions and content conditionings. Eventually, the latest deep generative model trends applied to speech generation of either kind are reviewed and detailed.

In chapter 3 we introduce the different blocks that are used to compose deep learning models. Concretely, we focus in the description of the basic blocks that can be composed to build the structures used throughout this thesis, like convolutional neural networks with their design patterns, recurrent neural networks and attention models. Finally, in section 3.2 we describe deep generative models and their current taxonomy. In sections 3.2.1 and 3.2.2.1 we describe likelihood-based models and generative adversarial networks with detail owing to their impact with respect to the work developed in this thesis.

In chapter 4 we develop the first part of the thesis, where we pursue the development of pseudo-recurrent structures to make TTS systems efficient without loss of quality. To that end, we leverage the sequential processing capabilities of quasi recurrent neural networks and self-attention models, presented in sections 4.2 and 4.3 respectively.

Then, we work on the development of an end-to-end speech enhancement generative adversarial network (SEGAN) in chapter 5. A thorough exploration of architecture variations of this model are performed through ablation studies in section 5.3.1. Moreover, in section 5.4 we study the transferability of this proposed model towards new languages and noises unseen during training, and what is the least amount of adaptation data required to work well in those new conditions departing from an English model. Another dimension of study is the use of SEGAN to make whispered-to-voiced conversions in a pursuit of a generalized enhancement systems thanks to the potential of deep generative models. This is partly motivated by the application of reconstructing natural speech from whispering data synthesized from an articulator motion capture device (Fagan et al., 2008) used to aid people that suffered a total laryngectomy. Finally, after showing the capabilities of this model to both denoise and reconstruct missing speech components, a more generalized speech enhancement approach is proposed in section 5.6. The introduction of this mixture of severe adverse conditions on the signal, discussed in section 5.6, requires an upgrade of the model to palliate audible distortions that are perceptually unpleasant. This brings the acoustic losses, described in section 5.6.1, which are attached in the end of the discriminator to establish a multi-task that weights perceptually the quality of the generated signals. Finally, a summary of the proposed contributions in this part of the thesis is detailed in section 5.7.

Chapter 6 is devoted to two main proposals: a problem-agnostic speech encoder and the self-supervised framework to train it. We begin proving that this encoder conveys high level abstractions from the speech waveforms, such as speaker identities, emotional cues and spoken content classification. We also prove which parts of the self-supervised learning are effective with an ablation study. Then, we return to the speech generation problems by first applying this encoder to work as a speaker and content identifier. Concretely, in section 6.4 we provide an example of usage of PASE as a speaker identifier for acoustic models in TTS. Moreover, we also apply it to do speaker adaptation without any model training thanks to the generalization of PASE to out-of-corpus identities.

Finally, chapter 7 presents a summary of the research presented in this thesis, as well as a detailed list of the key contributions. There is also a set of future perspectives on where could the presented research lead to in section 7.2.

Chapter 2

Literature Review

We use the term speech processing to refer to all those techniques that extract and encode information from the speech signals at different levels (e.g. emotion, content, etc.), as well as those that generate speech signals. This chapter comprises a review of techniques to perform different facets of speech processing. Owing to the fact that this work comprises an exploration of deep generative models and unsupervised learning techniques for speech processing tasks, as stated in chapter 1, the following review rapidly moves into the latest techniques framed within the usage of deep neural models. In the following, three main topics are reviewed for the proper contextualization of the different facets of this work; these are speech generation, speech enhancement and speech representation learning. The first two will be related under the speech-to-speech paradigm introduced in chapter 1. Nonetheless, a separate review of speech enhancement techniques is done, specially focused on deep learning applications, due to its importance in the development of the speech enhancement system proposed in chapter 5.

2.1 Speech Generation

Over time there have been several proposals of techniques to perform speech generation, both for text-to-speech (TTS) and speech-to-speech (STS) approaches. Specially TTS and voice conversion research brought the most innovative techniques to obtain flexible and scalable models, together with new levels of naturalness in the generated voice, until current systems that practically sound indistinguishable from natural speech in some cases. What follows is an overview of classic and recent techniques to generate speech. This overview mixes concepts from TTS and STS systems, as we abstract the conditioning factor and pay attention to the speech signal generation techniques themselves.

2.1.1 Unit Selection Concatenative Speech Synthesis

The classic option we find in speech generation techniques is the so called unit selection concatenative speech synthesis (US). It is based on the concatenation of small speech units selected from a large database (Hunt and Black, 1996), basically pasting pieces of real voice, which can already sound natural if their connection is continuous and smooth. However, if the concatenation is not effectively performed it yields glitches and audible discontinuities, this technique loses naturalness. The units can be phones as they serve as an elemental spoken unit. Nonetheless, a typical choice are diphones, which are constructed as the transition between two phonemes (i.e. from half of the left-side phone till half of the right-side phone). The reason for this is to find a structure which is least affected by co-articulation, which is the influence of a phone context to itself (Hardcastle and Hewlett, 2006). As such, they are

preferable even in scenarios with scarcity in the number of units (Hunt and Black, 1996).

During the construction of the waveform, two dimensions of decision have to be considered for the unit selection: the appropriate units depending on the target content, and the concatenation cost. As mentioned earlier, the content may come from a textual input as in TTS (Hunt and Black, 1996), or from a spoken input as in STS (Sundermann et al., 2006; Jin et al., 2016).

A US system is comprised of a training part, where the datasets of speech units and their contexts are built out of aligned transcriptions. During synthesis, the phonetic cues to compose the utterance are inferred out of text. Then these are used to select the proper units, which in turn depend on their contextualized units. After units are selected, some signal processing is applied to smooth the signal transitions. As mentioned, US allows us to construct the most natural speech in terms of signal quality, but the global naturalness of the full utterance might be broken by bad concatenation decisions. This means that selecting the proper continuity matching units and even post-processing the signal adequately is of essential importance. But selecting the proper units first means having enough spoken contexts represented to select units from, which cannot happen unless we have a large database. It is known that US systems require big datasets of units to achieve a human level of naturalness, both in terms of waveform low level features but also in fluency and prosody (Hinterleitner et al., 2016; Clark et al., 2007). For instance, works experimenting with varying corpus sizes, as in Clark et al. (2018), state that a corpus of around 36,000 phones (1.4 h of speech) provides an intelligible voice, but the system performs much better with much bigger corpora, and the authors still experiment quality improvements when using a corpus of 7 h. The signal processing block can also transform the signal to express certain emotional acoustic cues (e.g., sound happy, sound sad, etc.). Nevertheless, the main way to achieve this expressiveness is via expressive databases. This is called expressive speech synthesis (Cahn, 1989; Bulut et al., 2002). We may even like to transform the acoustic units to change the speaker identity if we could. A proper adaptation with some simplistic processing would make this speech synthesis system scalable to any desired condition. Nonetheless, US systems are not flexible enough in terms of ease of adaptation to different spoken conditions; This is because for each condition change, either new recordings are needed to properly pick contextualized units, or specialized signal processing techniques are required to transform the existing units. These transformations, however, have a limit in the trade off between adapting the speech and distorting it. Another potential issue for US systems is the fact that the speech data has to be carried around with the synthesis system, as units themselves are the key component needed to build streams of speech (Black et al., 2007).

2.1.2 Statistical Parametric Speech Synthesis

The second methodology used to do speech generation is the so called statistical parametric speech synthesis (SPSS). This is grounded under the concept of building statistical models that generate speech signals, instead of retrieving them from a dataset. If we happen to have a statistical model that learns the structure of speech and we can generate new signals from it, then we can palliate the big drawbacks of US that limit the system's flexibility (Black et al., 2007). Formally, we can express that a speech signal is a stream of acoustic frames $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t, ..., \mathbf{x}_T]$, where $\mathbf{X} \in \mathbb{R}^{T \times A}$, being T the number of frames, and $\mathbf{x}_t \in \mathbb{R}^A$, being A the number of acoustic features per frame. Depending on the type of acoustic features for which

our model learns the distribution, we have a trade-off between T and A parameters. If we happen to model the waveform samples directly, then $A = 1$ and T is the total number of samples we are modeling. Considering these variables, we can build the statistical model Φ expressed as

$$\Phi(\mathbf{X}) \approx p(\mathbf{X}) = p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T), \tag{2.1}$$

thus resembling as close as possible the true underlying distribution $p(\mathbf{x})$ that generated the composition of frames $\mathbf{x}_t$ that lead to the full signal $\mathbf{X}$. Importantly, if our model Φ actually resembles the generation process that builds speech signals, the sampling process can be conditioned to some additional factors $\mathbf{h} \in \mathbb{R}^F$, where F is the number of conditioning features. These can express contents factorized from some other signals (e.g. text (TTS) or speech (STS)), speaker identities, or prosodic changes among others. Our model then learns a conditioned distribution

$$\Phi(\mathbf{X}|\mathbf{h}) \approx p(\mathbf{X}|\mathbf{h}) = p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T|\mathbf{h}), \tag{2.2}$$

by which we achieve a great degree of flexibility for our speech generation system based on some external control we inject at sampling time. The fact that we can easily manipulate the characteristics of speech at generation time motivates the development of the aforementioned SPSS (Zen et al., 2009; Ling et al., 2015). Additionally, the memory footprint of these systems can be much smaller than that of US. This happens because the model Φ is learned from a training dataset $\mathcal{X}$ of examples $\mathbf{x}_n \subset \mathcal{X}$, where $\mathbf{x}_n$ would be an utterance's acoustic representation. Out of these samples $\mathbf{x}_n$, our model learns to approximate the aforementioned distribution, and then we can just ship our speech generator model without the dataset. It is thus encoding the spoken contents, the identity, and the prosody factors inside the model.

2.1.2.1 Hidden Markov Model Based Statistical Parametric Speech Synthesis

Hidden Markov model (HMM) based SPSS (Zen et al., 2009) was the first approach to model the speech generation process in a statistical way, heavily used in TTS. These systems do not work at waveform level, but with an intermediate acoustic representation. The speech signal is typically encoded with a vocoder to generate acoustic feature frames that describe the spectral envelope and its intonation, where the entire utterance is processed with a sliding window shifted every τ ms. We may typically use $\tau = 5\,ms$, which is a shift of 80 samples at 16 kHz sampling rate. Then we would obtain a collection of $T/80$ frames (following the T notation of equation 2.1), each one containing typically 40 to 60 parameters representing the spectral envelope, the value for the $F0$, and 5 parameters to describe the spectral envelope of the aperiodic excitation in the case of a source-filter vocoder (King, 2011) and a single parameter described the voiced frequency for an harmonics-plus-noise (HNM) vocoder (Erro et al., 2011). The dynamic representations of these features are also included to form the acoustic parameters of a series of phonemes in certain phonetic and prosodic contexts, which are then represented by context-dependent phoneme HMMs, typically with single Gaussian state-output probability density functions (PDFs). The contexts of the phonemes are defined with decision-trees based on rules asserted with linguists' prior knowledge (Zen et al., 2009). The trees end up clustering similar HMM output PDFs attending to the phonetic and prosodic features. Figure 2.1 depicts the general scheme for a statistical parametric speech synthesis system. Importantly and in contrast with US systems, the SPSS model is

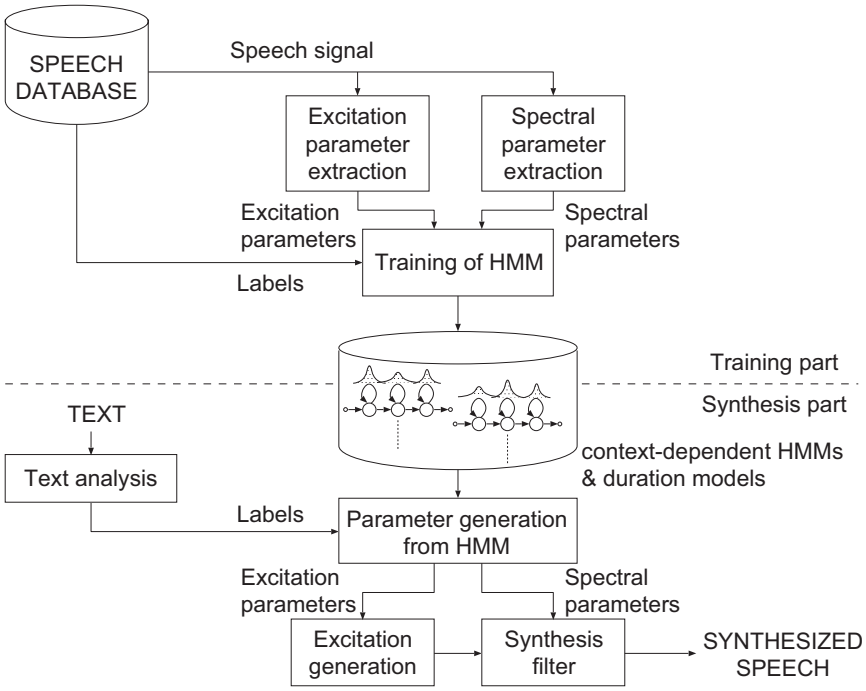

FIGURE 2.1: SPSS framework schematic (from Zen et al. (2009) Fig. 3).

learning the PDFs that can generate the acoustic parameters of speech in any context, whereas in US we use the parameters to compute the cost of concatenating two units, but the speech is retrieved as-is from the dataset of small waveform units. In SPSS, once we are in the synthesis part, we retrieve the desired acoustic models to generate the utterances mapped from the textual analysis. In this case then the linguistic and prosodic features make us pick the models that generate the different factors that compose the final utterance. First, we need a duration model that specifies how many frames will the current phoneme last. Secondly, we need an acoustic model that predicts the features themselves, generating as many frames as indicated by the duration model prediction. This is called a two-stage TTS approach. After the sequence of acoustic frames is generated, a speech parameter generation algorithm finds the most probable acoustic feature trajectories under the constraints between the predicted static and dynamic features (Tokuda et al., 2000), and the vocoder algorithm that we used to extract the acoustic parameters reconverts the parameterization of frames into waveforms of synthesized speech. Examples of vocoders used in SPSS are Ahocoder (Erro et al., 2011), WORLD (Morise et al., 2016), STRAIGHT (Kawahara, 2006) and Vocaine (Agiomyrgiannakis, 2015).

Machine learning was first used in speech generation with the aforementioned SPSS systems, built upon HMMs with single Gaussian state-output PDFs. This first application of machine learning seemed effective to build flexible and scalable models, but the generated voice quality did not surpass the quality of US approaches. This happened mainly due to three factors: the vocoder lossy compression, the modeling accuracy, and the so called over-smoothing effect (Black et al., 2007). The latter is introduced because of the averaging effect of the statistical model during training. Averaging together many frames of speech, each with slightly differing spectral properties, will have the effect of widening the formant bandwidths and reducing the dynamic range of the spectral envelope (King, 2011). During synthesis the model

tends to produce over-smooth spectral envelopes and over-smooth trajectories, although the former seems to be the one mainly affecting the generated speech quality (Zhang et al., 2008). This effect translates into producing speech that sounds muffled.

2.1.2.2 Deep Learning Based Statistical Parametric Speech Synthesis

In early SPSS works, where deep learning began to give good results, the multi-layer perceptron (MLP) was used as an acoustic model for SPSS (Zen et al., 2013). Here, the decision tree for contextualization with the Gaussian distribution of HMM based SPSS is substituted by a MLP that generates the acoustic parameters, both static and dynamic, on a frame-by-frame basis. MLPs allowed to obtain a better modeling of the complex context dependencies, outperforming the more classic approach of decision trees in a first approach. Other works pointed in similar directions by using MLPs for this mapping (Lu et al., 2013a; Qian et al., 2014; Kang et al., 2013), differing in the studied training methods like using a pre-trained deep belief network, varying the size of the proposed and baseline models, or using different input linguistic representations. Other research directions with the introduction of deep networks explored also the use of different target acoustic representations that could improve the perceptual results (Hu et al., 2015), as well the computation of losses in acoustic domains perceptually more relevant than those of vocoder parameters (Valentini-Botinhao et al., 2015).

These MLP models predict the acoustic frames with the perspective of a regression model, which typically takes the form of a mean-squared error regression between the output features of the training data and the predicted values (Zen et al., 2013). This works under the assumption that the acoustic data lays over a Gaussian distribution (Bishop, 1994), which can easily be a biased assumption (specially for complex data such as speech acoustic parameters). But even if it is the case, the predictions will tend to converge around the mean of the assumed Gaussian, ignoring the actual variance. Additionally, the distributions of acoustic features given linguistic features can be multimodal since humans can speak the same text in many different ways (Zen and Senior, 2014). The regression approach might then be problematic as the average of the acoustic features may actually be close to none of the modes of the distribution. To address these limitations, Zen and Senior (2014) investigated the use of mixture density networks (MDNs; Bishop, 1994), which can model full PDFs over real-valued output features conditioned on the corresponding input features. This type of model predicts the parameters of a Gaussian mixture model (GMM) instead of the acoustic parameter values, and it is trained based on the maximum likelihood criteria. Once the GMM parameters are obtained, the means and variances can be used by the speech parameter generation algorithm (Tokuda et al., 2000) to obtain the acoustic trajectories that will be fed into the vocoder. Zen and Senior (2014) demonstrate that predicting variances and multiple mixture components increased the model prediction accuracy and improved the synthetic speech quality significantly. In a similar approach Uria et al. (2015) used a real-valued neural autoregressive density estimator (RNADE) (Uria et al., 2013). The objective of this method is similar to that of the MDN, but the main difference is that RNADE predicts each dimension within an acoustic frame sequentially. This imposes the RNADE to also capture dependencies between the different acoustic features in a frame.

The recurrent neural network (RNN) has also been applied to speech generation due to the fact that we face a sequence generation problem. Recurrent networks are

specially a good fit due to their wide context coverage through time for each prediction (see section 3.1.4). For instance, in TTS, RNNs are used for prosodic and acoustic parameter prediction for vocoded TTS with different settings (Chen et al., 1998; Fernandez et al., 2014; Zen and Sak, 2015). In this case, HMMs are not used anymore because the RNNs can handle the temporal dynamics. Also because of this, dynamic acoustic features are not predicted as RNNs smooth the output acoustic trajectories. A main variation among these different works using RNNs is the application of diverse recurrent structures like the long-short term memory (LSTM; Hochreiter and Schmidhuber, 1997) or the gated recurrent unit (GRU; Chung et al., 2014), with one temporal direction or two (bidirectional). Besides, Wu and King (2016) explicitly investigate the effect of the different gating components embedded in the advanced recurrent structures and propose an optimized architecture for TTS. Also other works cover more recent pseudo-recurrent structures that save computation compared to the purely recurrent ones, reaching a comparable generated speech quality level to that of RNNs (Pascual et al., 2019a).

An important feature mentioned earlier about modeling the generation process statistically is its ease of adaptation to new conditions, hence flexibility. A first interesting trait that we can change during generation is the speaker identity. In equation 2.2 it is shown how some conditioning information embedded in a vector $\mathbf{h} \in \mathbb{R}^F$ can control what is generated and how. It has been detailed that typically we can condition either in a TTS or an STS approach, but we can additionally control a factor like the speaker identity at generation time. Wu et al. (2015) do a deep neural network (DNN) based speaker adaptation by adding the speaker identity information to the input features with discrete tokens. They also apply learned hidden unit contributions (Swietojanski and Renals, 2014), and make output feature space transformations. They showed different combinations of results by adding and subtracting the different adaptive components. Fan et al. (2015) and Pascual and Bonafonte (2016a), on the contrary, built multi-speaker capabilities in the generator structure.

2.1.2.3 Towards End-to-End Speech Generation

As introduced earlier in this section, modeling a PDF explicitly with neural networks is what deep generative models are about. We have some data points from a trainset $\mathcal{X}$ and we want to know how were those points $\mathbf{x}_n$ generated. WaveNet is a deep convolutional neural network designed to predict waveform samples autoregressively by explicitly modeling the joint PDF $p(\mathbf{x})$ with a product of conditionals (van den Oord et al., 2016b). This decomposition allows the WaveNet to learn to mimic the generating process that composes valid acoustic samples, which sound like the ones in the trainset $\mathcal{X}$. Because of its importance as an explicit likelihood-based deep generative model, this mechanism is detailed in chapter 3. Remarkably for TTS, the first prominent contribution of this work is that it tackles directly the waveform generation (hence the name WaveNet), without intermediate acoustic parameterizations that may lead to lossy compressions (as with vocoders). Another important contribution of this work is related to MDNs. Differently than predicting a parameterized PDF as in MDNs, the authors of WaveNet build a classifier trained on a maximum likelihood criteria. They first quantize the waveform amplitude with 8 bits after a μ-law encoding that ensures to reduce the quantization error. Reducing the quantization resolution to 8 bits simplifies the problem, as they dramatically reduce the number of possible classes from 32,768 that would correspond to 16 bit signals to 256, corresponding to 8 bit ones. This means that WaveNet is trained to predict, at each time-step t, the index of its quantized histogram, out of which the mean

value of that range is selected as the sample value in that waveform time instant. Then, at synthesis time the resulting PDF yielded at the model output if sampled to get a likely class out of the 256 possible ones. Note that this differs from the MDN commented earlier, as it rather used the speech parameter trajectory generation algorithm. This turns out to yield a very flexible generative model, surpassing the distortions introduced by acoustic parameterizations as it works upon waveforms. Also, its naturalness goes even beyond than the one for their baseline US system (van den Oord et al., 2016b). This type of model is an important breakthrough, as it surpasses the naturalness of US TTS at no expense of flexibility, so it can model different linguistic contents, intonations and speaker identities at a time. The fact that the waveform can be directly generated by a neural model turned the attention of the speech generation research fields into end-to-end perspectives, where one can potentially go directly from raw input representations like text or speech waveforms to output speech waveforms by training a single network to perform all the required intermediate steps via backpropagation. WaveNet demonstrated it could work as an intermediate end-to-end step to TTS by injecting aligned linguistic features based on the ones used for MLP-based SPSS systems. Additionally, successive works showed that conditioning WaveNet on acoustic parameters extracted from a vocoder allows to recover speech with higher quality in comparison to the vocoder reconstruction itself (Tamamori et al., 2017). This type of WaveNet conditioned on the acoustic parameters is called neural vocoder, and typically the input features take the form of mel spectrum bins. This involved the possibility of tackling STS problems like voice conversion by targeting directly the waveform, as a step towards an end-to-end transformation solution as done in the TTS WaveNet (Kobayashi et al., 2017; Niwa et al., 2018).

The main issue of WaveNet is known to be its efficiency due to that fact it takes several computation steps (across the deep structure) to predict just one sample of the waveform, and the waveform is predicted one sample at a time. This involved a new branch of research within speech generation devoted to improving the efficiency of deep autoregressive generative models. First, a technique called *probability density distillation* (Hinton et al., 2015) allowed to build a faster WaveNet, which learns to generate samples in parallel by predicting the parameters of a mixture of logistics (similar to the MDN parameterization). This parallel WaveNet is trained by matching its output likelihood to the one yielded by a pre-trained WaveNet classifier network, in addition to other losses that assess the correctness of the output signal's power (power loss), or the intelligibility of the spoken contents (speech recognition loss). In this case the WaveNet was turned into a parallel system with a change in the generative modeling perspective, becoming an inverse autoregressive flow generative model (Rezende and Mohamed, 2015), although neural distillation was used to avoid the computationally expensive training process of inverse autoregressive flows.

Kalchbrenner et al. (2018) also worked on improving the efficiency of autoregressive speech generation by turning the deep convolutional WaveNet structure into a simpler recurrent one, named WaveRNN. Kalchbrenner et al. (2018) designed an architecture based on the GRU, which predicts each sample at a time. In this case they propose to use 16 bit quantization to build a classifier again at the output. They predict the most significant and least significant part of the two bytes separately in two branches of 256 classes each. This way they achieve a much higher resolution of the waveform amplitude at the expense of just doubling the number of outputs, rather than exponentially increasing them. WaveRNN also comes with different techniques

to accelerate its operations such as sparsification of the network parameters and different sub-sequential conditioning schemes. Nevertheless, the vanilla WaveRNN proposed as a more compact form of the WaveNet already reaches the same naturalness than WaveNet, but being much faster (Kalchbrenner et al., 2018). WaveRNN can also become a more convenient option over the parallel WaveNet mainly due to its facility to be trained under the maximum likelihood criteria, as in the vanilla WaveNet. WaveNet and WaveRNN are then interchangeable, as the two systems generate waveforms autoregressively and can be conditioned on features that drive the sample generation. Hence, WaveRNN can also be used as a neural vocoder. The neural vocoder research has become very trendy during the latest years, focusing on compactness, ease of training and efficiency in the form of inference speed. Some well known systems designed as neural vocoders contemporary to WaveNet or compacting and parallelizing its design are the SampleRNN (Mehri et al., 2016), the FFTNet (Jin et al., 2018), the LPCNet (Valin and Skoglund, 2019) and the WaveGlow (Prenger et al., 2019). The SampleRNN, the FFTNet and the LPCNet are efficient multi-scale structures that also predict the waveform samples autoregressively with structures that resemble that of the WaveRNN. The WaveGlow, on the other hand, is based on real non-volume preserving normalizing flows (Rezende and Mohamed, 2015). This sort of model allows to parallelize the operations and still compute an exact likelihood model, at the expense of needing several layers of transformation and requiring long training sessions to converge. Nevertheless, at synthesis time it is a much more efficient option in comparison to WaveNet, generating at a rate of 4850 kHz on an NVIDIA V100 GPU[1]. Similarly to the aforementioned application of WaveNet to STS and making a step forward towards end-to-end voice conversion, Blow is proposed as a flow-based model to perform unparallel voice conversion directly between frames of waveforms (Serrà et al., 2019). Note that in this case no intermediate acoustic representation is used to carry the conversion task, hence the identity and content are implicitly extracted within the model structure from the raw waveform to obtain the target raw waveform.

So far the end-to-end drift has been described in the output of the models for both TTS and STS paradigms. However, all the reviewed systems work with signal alignments known ahead of time. Hence for TTS, this means that a duration model precedes an acoustic model, as introduced earlier in section 2.1.2.1. The former one predicts the number of acoustic frames to be generated, and the latter predicts them. Contrarily to this approach we have the recent models based on sequence-to-sequence structures (see section 3.1.6), which directly predict the acoustic signal from the linguistic contents without an intermediate duration prediction step. This is done through a learnable alignment model called attention mechanism, which learns the relations between the input sequence of features and the output acoustic features, hence adapting the sequential resolution difference. Tacotron TTS models (Wang et al., 2017) follow this sequence-to-sequence design, and in contrast to WaveNet they meant a shift towards end-to-end models that directly processed raw text to output the acoustic representation. Unlike previous work, Tacotron models do not need hand-engineered linguistic features like WaveNet, or complex components such as an HMM aligner as in non sequence-to-sequence models. Additionally, Tacotron models predict acoustic parameters like mel spectrum frames, hence making it a frame-based approach like the acoustic models prior to WaveNet. This makes their inference substantially faster than waveform-level autoregressive methods (Wang et al., 2017). The way to recover the waveform in the original Tacotron model was

[1] https://github.com/NVIDIA/waveglow

by means of the Griffin-Lim phase reconstruction algorithm for simplicity. Nonetheless, successive versions of the system used neural vocoders like WaveNet to achieve the state of the art quality of waveform-based TTS systems (Shen et al., 2018). Other modifications of the model included style tokens and prosodic features that drive the Tacotron outputs to be more controllable in terms of prosody (Wang et al., 2018; Skerry-Ryan et al., 2018). The effectiveness of these models depends on the availability of large amounts of speech and text data, as well as powerful computational resources to train them. For instance, Tacotron was originally trained on 24.6 hours of speech spoken by a professional female speaker (Wang et al., 2017). Other sequence-to-sequence variants are the DeepVoice models (Arik et al., 2017; Gibiansky et al., 2017; Ping et al., 2018), the Char2Wav (Sotelo et al., 2017), the Transformer TTS (Li et al., 2019) or the ClariNet (Ping et al., 2019). Although there is a trend towards end-to-end models, no model is yet applied directly to the conversion of characters to speech waveforms without intermediate representations, like Tacotron or WaveNet do. On the STS side, the sequence-to-sequence paradigm has also been applied to surpass the assumption that input and output sequences are aligned on a frame basis. This sequence-to-sequence modeling allowed the construction of STS solutions for voice conversion that rely on structures similar to Tacotron's, like Parrotron (Biadsy et al., 2019). In this case an input spectrogram is directly mapped to a target spectrogram thanks to the alignment learned by the attention model, and a vocoder reconverts the acoustic targets to a waveform.

Another remarkable deep generative model is the generative adversarial network (GAN; Goodfellow et al., 2014), detailed in chapter 3. This type of system allows to generate samples in parallel, but there is no explicit modeling of the loss function to learn our data distribution $p(\mathbf{x})$. Generative adversarial networks have also been applied to raw waveform speech generation. In STS they have been used for waveform-based speech enhancement (Pascual et al., 2017), spectrum-based voice conversion (Kameoka et al., 2018), and efficient neural vocoding (Kumar et al., 2019). They have also been applied in TTS (Donahue et al., 2019; Bińkowski et al., 2019), achieving recently state of the art results compared to autoregressive approaches in Bińkowski et al. (2019), hence becoming a new breakthrough in terms of quality and inference efficiency.

2.2 Speech Enhancement

Speech enhancement aims to improve the intelligibility and quality of speech contaminated by additive noise (Loizou, 2013). Its main applications are related to improving the quality of communications in noisy environments. However, we also find applications related to hearing aids and cochlear implants, where enhancing the signal before amplification can significantly reduce discomfort and increase intelligibility (Yang and Fu, 2005). Speech enhancement has also been successfully applied as a preprocessing stage in speech recognition and speaker identification systems (Ortega-Garcia and Gonzalez-Rodriguez, 1996; Yu et al., 2008; Maas et al., 2012).

Most of the current speech enhancement systems are based on the short-time Fourier analysis/synthesis framework, where only the spectral magnitude is treated to remove contaminating artifacts (Loizou, 2013). Recovering the signal is, in that case, a matter of recombining the cleaned-up magnitude with the input phase. This

approach is common practice, as it is often claimed that short-time phase is not important for speech enhancement (Wang and Lim, 1982). Nonetheless, other studies show that significant improvements of speech quality are possible, particularly when a clean phase spectrum is known (Paliwal et al., 2011). Classic speech enhancement includes spectral subtraction (Berouti et al., 1979), Wiener filtering (Lim and Oppenheim, 1978), statistics-based methods (Ephraim, 1992) such as the minimum mean squared error (MMSE), and subspace algorithms (Dendrinos et al., 1991; Ephraim and Van Trees, 1995).

Neural networks are a recent and successful trend for this task, although they were initially applied in the 1980s by Tamura and Waibel (1988), and later by Parveen and Green (2004). Recent widely used architectures typically work in the spectral domain, as with classic techniques, to learn a regression to the clean spectrum, typically in the form of a denoising autoencoder (DAE; Lu et al., 2013b; Xu et al., 2015). Other approaches work by predicting masks with deep neural networks that palliate noisy spectral regions (Narayanan and Wang, 2013; Williamson and Wang, 2017; Wang et al., 2014). Recurrent neural networks (RNNs) are also used, owing to their success in modeling sequential processes. Research shows that RNNs can predict a better contextualized set of frames or masks (Maas et al., 2012; Weninger et al., 2015; Weninger et al., 2014; Erdogan et al., 2015). The use of dropout, postfiltering, and perceptually motivated metrics is also effective. Xia and Bao (2013) propose to use a weighted DAE, altering the mean squared error loss function by assigning weighting factors to each spectral component. Furthermore, Shivakumar and Georgiou (2016) use a loss function that considers the perceptual quality of speech, and Fu et al. (2018) use an intelligibility loss to obtain better scores than those of plain regression losses. Williamson and Wang (2017) use a deep neural network (DNN) in the spectral domain, including the phase, by working with complex masks.

Convolutional neural networks (CNNs) are also known to perform well for locally correlated data, such as speech waveforms or spectrograms. As such, we have used them for one of the first speech enhancement systems working with the raw audio signal, hence shifting the enhancement paradigm into an end-to-end modality (Pascual et al., 2017). Other contemporary studies use deep convolutional structures for this task in the form of regression architectures, such as the work by Park and Lee (2017), who emphasize the need for reduction in model size (typically achievable through CNNs), or the denoising WaveNet (Rethage et al., 2018). Other approaches use improvements in the adversarial setup in the form of a Wasserstein GAN with gradient penalty (Gulrajani et al., 2017; Qin and Jiang, 2018). Moreover, adversarial losses have been used in the speech enhancement field to work without parallel corpora of aligned pairs (Higuchi et al., 2017). The adversarial framework also appeared as a methodology to combine speech enhancement together with automatic speech recognition systems, either in the waveform or the spectral domain (Donahue et al., 2018; Meng et al., 2018).

SEGANs were first applied to the denoising problem, and were later adopted to reconstruct speech components from what is called whispered speech, where aphonic speech is corrected to attain the intonation and identity back (Pascual et al., 2018b; Pascual et al., 2019b). Following this trend, SEGAN has been extended to be applied upon a more generalized speech enhancement concept, where the system must recover cleaner speech out of severely distorted utterances from a myriad of combined degrading transformations. These include whispered speech, clipped signals, sample losses in the form of zeroed-out sections of speech and bandwidth reductions at different scales (Pascual et al., 2019c). As stated in previous works, this generalization trend of speech enhancement to palliate different distortions at once

is interesting due to its direct applicability to modern communication technologies, where connections are interrupted (sample losses), and voice processing pipelines can distort the signals through amplifiers, codecs, etc.

2.3 Unsupervised Speech Representation Learning

Finally, on the other side of the speech processing pipeline there is the analysis section, where information is retrieved from an input signal. This is directly related to speech recognition, speaker recognition, or any task requiring speech as input for an analysis of high-level factors. Examples of these factors could be the speaker identity, the spoken emotion, the spoken content, etc. These problems can be tackled whenever we have matching annotations with our speech data. Nonetheless, in the following we analyze a broader concept with regard to learnable speech representations by means of deep unsupervised learning.

The success of deep learning techniques strongly depends on the quality of the representations that are automatically discovered from data. These representations should capture a hierarchy of different levels of concepts, features, or latent variables, and are commonly learned in a supervised way using large annotated corpora (Goodfellow et al., 2016). Even though supervised learning is still the dominant paradigm, some crucial limitations arise. First, collecting large amounts of annotated examples, for instance, is very costly and time-consuming. Moreover, if not learned with a large pool of tasks (Serrà et al., 2018), supervised representations are likely to be biased towards the considered problem, limiting their exportability to other problems and applications (Rosenstein et al., 2005).

A natural way to mitigate these issues is unsupervised learning (Bengio, 2011). Unsupervised learning attempts to extract knowledge from unlabeled data, hence first avoiding the efforts of manual annotations. It can potentially discover representations that capture the underlying structure of such data. Several approaches have been proposed for unsupervised learning in the last decade. Notable examples are deep autoencoders (Bengio et al., 2006) and restricted Boltzmann machines (Hinton et al., 2006). In speech processing these can be employed as pre-training stages for subsequent supervised tasks like speech recognition (Dahl et al., 2012) or speech synthesis (Kang et al., 2013). More recent techniques, introduced in chapter 3, include variational autoencoders (Kingma and Welling, 2014; van den Oord and Vinyals, 2017), likelihood models (van den Oord et al., 2016b; Kalchbrenner et al., 2018; Kingma and Dhariwal, 2018) and generative adversarial networks (Goodfellow, 2016).

A sub-field of unsupervised learning that is gaining popularity due to its simplicity and effectiveness is the so called self-supervised learning. This modality is especially trendy nowadays for computer vision and NLP tasks. In self-supervised learning, the targets are extracted from the input signal itself in an automatic fashion. For instance, Doersch and Zisserman (2017) predicts the relative position between two image patches, colorizes grayscale images, predicts whether two images are transformed under the same function (e.g. rotation) and predicts what pixels will move in time between two video frames. All these tasks are hence useful to build a generalizable image representation. Similarly, Gidaris et al. (2018) use a convolutional encoder to predict the amount of rotation applied to the input image, and Misra et al. (2016) predict whether an input video sequence has the right temporal order of frames. In NLP, another well known case of self-supervision is the BERT model learning strategy (Devlin et al., 2019). In this case a bidirectional language model is built, where it has to predict what tokens of an input sentence

have been masked out, as well as which are the neighbour sentences. This learning strategy reduces the need for heavily-engineered task-specific architectures, and the pre-trained BERT achieves state of the art performance on several sentence level and token level tasks (Devlin et al., 2019). Some other attempts have also been done to extend self-supervised learning to different modalities (Arandjelović and Zisserman, 2018; Owens et al., 2018) or to audio representations only (Jansen et al., 2018; Chorowski et al., 2019; van den Oord et al., 2018; Ravanelli and Bengio, 2019; Chung et al., 2019). With this regard, a recent trend consists of learning speech representations using a neural network encoder followed by a binary discriminator that determines whether audio features are close in context or not (van den Oord et al., 2018; Hjelm et al., 2019; Ravanelli and Bengio, 2019). Also aligned with this trend, regression tasks are plugged on top of the encoder for feature discovery (Chung et al., 2019; Chorowski et al., 2019). These regressors take the form of decoders that reconstruct the input signal with a likelihood based generative model (see section 3.2.1).

Chapter 3

Deep Learning Review

Deep learning is a subset of tools in machine learning (Goodfellow et al., 2016). Statistical algorithms are used to learn features and functions automatically from data. In the case of deep learning, these algorithms are grounded in the use of neural networks. These models are based on stacks of neural layers, hence forming a deep structure. The use of these deep structures allows for multiple transformations to happen between the input data and the output of the model, which usually capture optimized features at different levels (Erhan et al., 2009), ranging from the lowest level (raw data or close to it) until the most abstract one (e.g. concepts like speaker identity, spoken contents, emotion, etc.). We will refer to network topology or network architecture to describe the connection pattern of the neurons in a layer and between layers.

3.1 Stacking in Depth: Fundamental Neural Blocks

In this section, a brief overview of the different basic neural blocks that compose neural networks is given. As mentioned, deep neural models are composed of a hierarchy of computational blocks named neural layers. The composition of these layers forms the neural network. Throughout the course of this work, four basic neural blocks are used to compose different deep models depending on the context in which they are used. These basic blocks are now reviewed.

3.1.1 Fully Connected Layer

This is the most basic type of layer that operates over a group of input features arranged as a D-dimensional vector $\mathbf{x} \in \mathbb{R}^D$. Fig. 3.1 depicts this scheme, where we have a fully connected neuron on top, also named artificial neuron, neural unit or neuron. This is the basic unit in the fully connected layer. This unit operates as

$$h = \tanh(\mathbf{w}^T\mathbf{x} + b), \tag{3.1}$$

where $\mathbf{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}^1$. Note that this is a sum of the weighted input vector and a bias with an additional non-linear activation, like the tanh. As such, it is usually formulated as the dot product between $\mathbf{w}$ and $\mathbf{x}$ and the addition of the bias, as in equation 3.1. The weights and the bias are the learnable parameters of the neuron. The output of a neuron is named as an activation. If we arrange H neurons together we build a fully connected layer. The extension from one neuron to H neurons is done by converting the weight vector $\mathbf{w} \in \mathbb{R}^D$ into a matrix $\mathbf{W} \in \mathbb{R}^{H\times D}$ and the bias scalar to a bias vector $\mathbf{b} \in \mathbb{R}^H$ in algebraic terms. This way when the dot product is applied, as well as the addition of the bias and the non-linearity as in

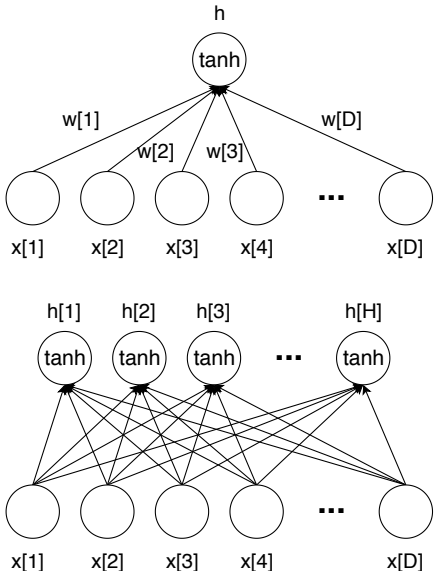

FIGURE 3.1: Top: fully connected neuron. Bottom: fully connected layer.

$$\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{3.2}$$

a vector of activations $\mathbf{h} \in \mathbb{R}^H$ is returned in the output of the layer. Note that the name fully connected comes from the fact that all the D inputs are connected to all the H units. These layers are stacked to form a deep structure, and the full stack becomes a multi-layer perceptron (MLP) or a deep neural network (DNN). Fully connected layers are also very useful in combination with other topologies, especially to make bottleneck sections in our data flow that enforce a compression of features to a lower dimension than the input one (Hinton and Salakhutdinov, 2006).

3.1.2 Convolutional Layer

Convolutional layers are a more specialized kind of neural layer for processing data that has a known grid-like topology (Goodfellow et al., 2016; LeCun, 1989). Examples of this are time-series, which is a 1-D grid taking samples at regular time intervals, or images, which are 2-D grids of pixels. This can obviously scale beyond these dimensions, to 3-D and 4-D grids, including volumetric representations or video. As we deal with discrete-time signals throughout this work, we only describe the discrete representation of the neural operators. The discrete 1-D convolution is defined as

$$y[n] = \mathbf{x} * \mathbf{w} = \sum_{m=-\infty}^{\infty} x[m] \cdot w[m-n], \tag{3.3}$$

where n is the discrete signal time-index. This means we have a filter $\mathbf{w}$ (also named kernel) that slides over the signal $\mathbf{x}$, performing a weighted sum at each sliding instant m. This operation is denoted with $*$.

Fig. 3.2 depicts a convolutional kernel with 3 weights sliding over a signal of 12 samples. The outcome $h[n]$ at every time-step is the result of applying the neuron operation (as in equation 3.1) with a certain width in time L over the input signal piece of the same length $\mathbf{x}_{n:n+L-1}$, hence

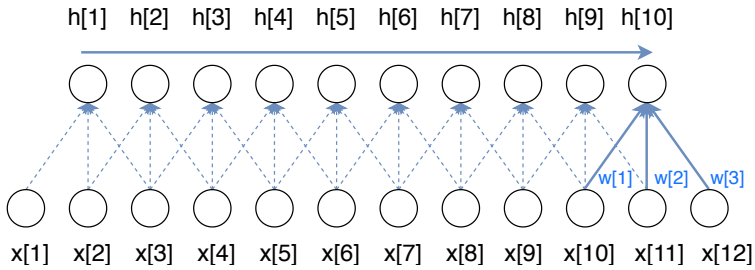

FIGURE 3.2: Example of a convolutional neural filter of size 3 sliding over the input signal.

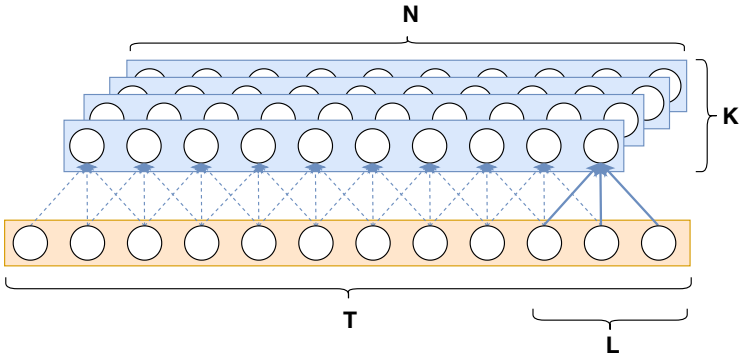

FIGURE 3.3: Example of a convolutional neural layer with $K = 4$ feature maps and $L = 3$ kernel width. The input signal has $T = 12$ samples, and each feature map $N = 10$ activations.

$$h = \sigma(\mathbf{x}_{n:n+L-1}^{T} \cdot \mathbf{w} + b), \tag{3.4}$$

with an appropriate non-linearity σ at the output. ReLU is often applied in the case of convolutional units, where $\sigma(a) = \max(a, 0)$, due to their usually better and faster convergence effects when compared to saturated activations like tanh (Jarrett et al., 2009; Krizhevsky et al., 2012). Similarly as with fully connected layers, we normally arrange multiple neural units altogether to extract multiple features per network layer. In the case of convolutional networks, we refer to K as the number of convolutional kernels, and the resulting collection of features with time-span are the feature maps. For an arbitrary convolutional layer we also refer to the collection of K feature-maps as output channels, whereas the input channels C would be the number of input feature maps to be processed. Fig. 3.3 shows an example arrangement of $K = 4$ convolutional kernels of length $L = 3$. The input signal has one channel with $T = 12$ samples, so the result of sliding the window has length $N = T - L + 1 = 10$, with one channel per k-th kernel. Convolutional layers can also consume inputs sequences with multiple channels C, which makes them able to be stacked together and connect the feature maps of a precedent layer with the current layer, and similarly with the subsequent layer too. This allows for the conjunction of kernels to be grouped together in tensors $\mathbf{W} \in \mathbb{R}^{L \times K \times C}$, and the conjunction of biases behaves exactly the same as in fully connected layers with arrangements of K real values. Note how signal lengths T and N do not describe any dimension of the weights tensor, which is agnostic to the time-dimension as it operates locally per sliding step with fixed input windows. We will denote the convolutional layer operation as

$$h = \text{ReLU}(\mathbf{x} * \mathbf{W} + b). \tag{3.5}$$

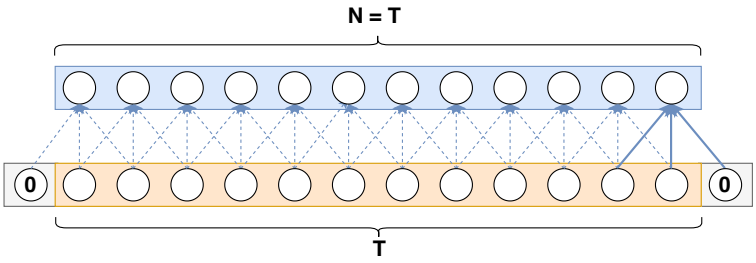

FIGURE 3.4: Padding the input signal so that the output feature map has the same length N as the input signal T.

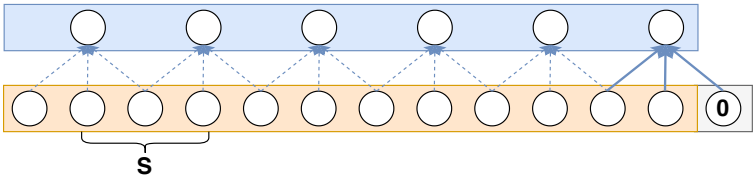

FIGURE 3.5: Example of a convolutional kernel striding by a factor $S = 2$, hence obtaining a feature map with decimated time resolution, from $T = 12$ to $N = 6$.

Apart from the basic operation described about the convolutional layer with the sliding windows and its parameters L, C and K, there are four important additional details that can be tuned to operate differently in the interaction with the data: padding, striding, grouping and dilation.

Padding allows to obtain a certain time-length in the output feature maps by injecting additional P values (normally zero) in our input signal beginning and end. Fig. 3.4 shows an example where the convolution operates upon a padded signal, with which the output feature map is enforced to fit in the same time-length as the input signal. Padding is used extensively when designing convolutional nets to ensure that the data flow contains specific amounts of information from layer to layer, and no decimation happens at signal extremes when $L > 1$. Besides, it is the key to converting a regular convolution into a causal convolution if we happen to pad only the left-side of the signal with $P = L - 1$ values. This is done in works that emulate autoregressive operations with convolutions like the WaveNet (van den Oord et al., 2016b) or the quasi recurrent neural network (Bradbury et al., 2017) to build pseudo-recurrent mechanisms whilst avoiding the burden of recurrent connections, which are reviewed further below.

Striding refers to the amount of samples by which the convolutional kernel is shifted at each time-step of analysis. We will refer to this factor as S. When $S = 1$, the sliding operation described above is performed, with each convolutional kernel moving one step ahead at a time. However, when $S > 1$, the signal is decimated by that S factor (if padding is set correctly). So the convolutional layer will perform a signal analysis and decimate at the same time. Fig. 3.5 exemplifies the use of a kernel striding by a factor $S = 2$. Note that padding is still necessary to preserve an exact decimation factor of 2 due to the kernel of length $L = 3$. It is usual to decimate and increase the number of kernels while growing in depth (LeCun, 1989; Krizhevsky et al., 2012), as it requires much less memory and accelerates the deeper layer convolutions because they slide through reduced lengths. We pragmatically refer as strided convolutions as those which have a factor $S > 1$.

Grouping refers to grouped convolutions (Krizhevsky et al., 2012). It is based on dividing the total number of input channels C into sub-groups of equal size G, hence

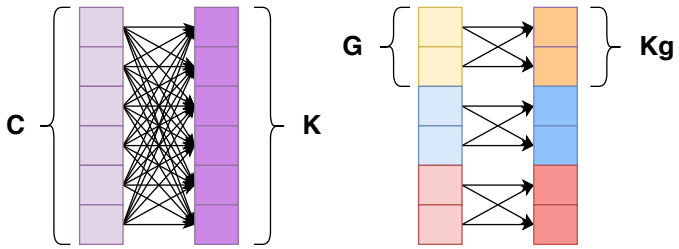

FIGURE 3.6: Comparison between input-to-output channel connectivity patterns in a normal convolution (left) and a grouped convolution (right). C is the number of input channels, K is the number of output channels, $G = 2$ is the group size and K_g is the number of output channels connected to its group of inputs. The representation is a slice across the kernel length, so time axis is not represented.

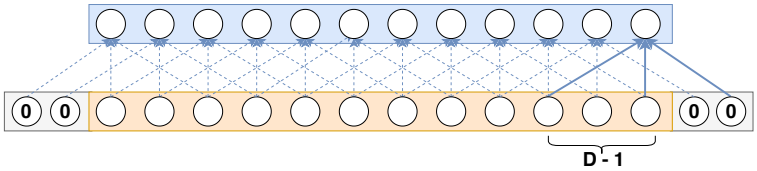

FIGURE 3.7: Dilated convolutional kernel with $D = 2$. In each time-step, the convolutional parameters have a time-span of 5 samples, although only the $L = 3$ weights are applied to the signal.

getting $\Gamma = C/G$ blocks of input channels. Then each block is connected to its own set of K_g convolutional kernels. Fig. 3.6 shows the different connectivity patterns that a normal convolution and a grouped convolution have in channel dimensions. In fact, the normal convolution operation is the special case of the grouped convolution when $G = 1$. Another extreme is that when $G = C$, in which case each input channel is connected to only one feature map, which is the cheapest option computationally. The grouping design was proposed originally to distribute the computation of large convolutional layers into different GPUs (Krizhevsky et al., 2012). Nonetheless, it has also been found to yield good results in computer vision under appropriate setups (Xie et al., 2017), as well as to be a key component to efficient neural designs focused on accelerating the model performance on embedded devices (Zhang et al., 2018b; Huang et al., 2018).

Finally, dilation is the addition of intermediate processing spaces within the kernel of length L, which makes it virtually longer than it really is, being the new length $\Lambda > L$. Fig. 3.7 depicts this design with a kernel of length $L = 3$, which can virtually be converted into a kernel of length $\Lambda = 5$ with a dilation factor $D = 2$. The dilation factor is translated into $D - 1$ void spaces between kernel weights. Due to these voids, the amount of parameters does not increase with this approach.

The receptive field is an important concept in convolutional networks design. It is the amount of context that a neuron from an arbitrary layer sees in the input to predict its output. The main advantage of dilated convolutions is enlarging the receptive field of the neuron without growing in parameters. This is depicted in Fig. 3.8, where different receptive fields are shown for networks of 2 convolutional layers. The first one, as a result of a regular stack of convolutions, grows linearly with depth. The second one, as a result of a stack where a strided convolution is present, grows exponentially. The same exponential receptive field increasing effect is yielded by a design with dilated convolutions (Yu and Koltun, 2015; van den Oord et al., 2016b). The choice of either decimation or dilation depends on the memory to

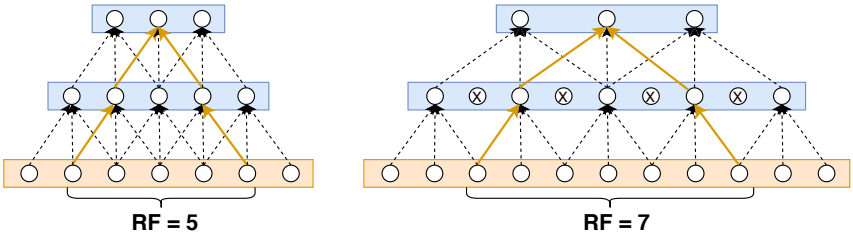

FIGURE 3.8: Recetive fields of a regular stack of a two convolutional neural layers (left) and a stack with a strided convolution with $S = 2$ (right). Crossed out units in the right-side indicate decimated samples ignored because of the doubled striding.

be consumed (dilation requires more) or the amount of signal details that have to be preserved (decimation removes more)[1].

To conclude with the description of convolutional layers, we can relate them to the previously presented fully connected layers. Concretely, a fully connected layer is also denominated 1×1 convolution, where a convolutional layer with $S = 1$ an $L = 1$ transforms the number of input channels C into K features per time-step (Szegedy et al., 2015). The 1×1 denomination for fully connected layers refers to the sliding process with which the neuron operation is applied, following equation 3.1, without time-dependency.

3.1.3 Transposed Convolutional Layer

The transposed convolution is a backward operation with respect to the convolution described before. Instead of summing up weighted contribution from different inputs, it scatters one input into weighted output portions. Fig. 3.9 depicts the transition from a convolution (as presented earlier) to a transposed convolution, which acts as a way back to the same original resolution. It can then be seen as a learnable interpolation. Each output element from the transposed convolution can be defined as

$$\mathbf{i}[t] = \mathbf{U}_t \mathbf{y} + b, \tag{3.6}$$

where $\mathbf{U}_t \in \mathbb{R}^{F \times K}$ is the t-th weight of the kernel, mapping the K inputs from $\mathbf{y} \in \mathbb{R}^K$ to $\mathbf{i}_t \mathbb{R}^F$ with F output features per time-step. Note that t denominates the output position in time for the case of sequential feature map interpolations. Therefore, contrarily to the contraction of convolutions from L inputs into one output, transposed convolutions scatter one input into L outputs. A non-linear activation may be plugged into the output of this operation as well. The transposed convolution is often called deconvolution too for short, although it may not be a good description of its actual operation. Transposed convolutions also include padding and striding mechanisms. The former is used, as in convolutions, to establish the proper output dimensions. The latter is used to control the interpolation factor[2].

3.1.4 Recurrent Layer

The recurrent neural network (RNN) leverages temporal correlations in its inputs, hence modeling the dynamics of sequential data. The way this is introduced in the neural architecture is through a new set of connections departing from the hidden

[1] https://tinyurl.com/santty128-rfconvs
[2] https://tinyurl.com/santty128-deconv

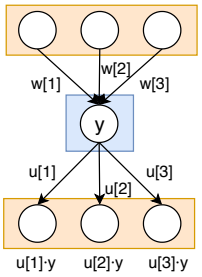

FIGURE 3.9: Transition from a convolution operation (top) to a transposed convolution (bottom). The **w** values denote the convolution weights, while the **u** values denote the transposed convolution weights.

layer and leading to all of its neurons again. Hence, the formulation is very similar to that of fully connected layers, but with the addition of the feedback connections. This adds a new level of abstraction to the equation 3.2 as

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}), \tag{3.7}$$

where temporal indices t are introduced given the sequential evolution of $\mathbf{x}_t$ and $\mathbf{h}_t$ signals, and the recurrent matrix $\mathbf{U} \in \mathbb{R}^{H \times H}$ expresses the connectivity pattern of all neurons interconnected to themselves within the layer. Nowadays vanilla RNNs are rarely used due to their known issues with respect to back-propagating gradients through time (Hochreiter et al., 2001), and their quickly vanishing memory owing to the multiplicative operation of matrix $\mathbf{U}$. This is why more advanced recurrent architectures like the long-short term memory (LSTM; Hochreiter and Schmidhuber, 1997) or the gated recurrent unit (GRU; Chung et al., 2014) seem more popular in current works. These two models share a common design principle named the gating mechanism, which reformulates the vanilla RNN equation 3.7 like

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_\mathbf{i}\mathbf{x}_t + \mathbf{U}_\mathbf{i}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{i}) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_\mathbf{f}\mathbf{x}_t + \mathbf{U}_\mathbf{f}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{f}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_\mathbf{o}\mathbf{x}_t + \mathbf{U}_\mathbf{o}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{o}) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_\mathbf{c}\mathbf{x}_t + \mathbf{U}_\mathbf{c}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{c}) \\
\mathbf{c}_t &= \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh \mathbf{c}_t
\end{aligned} \tag{3.8}$$

3.1.5 Quasi Recurrent Neural Layer

The quasi recurrent neural network (QRNN) was proposed as an efficient alternative to the vanilla RNN, by removing the affine projections from recurrent connections (Bradbury et al., 2017). The temporal dependency, introduced in equation 3.7 with $\boldsymbol{U}$ connections, dramatically limits parallelism and makes RNNs unwieldy for very long sequences. This happens because $\boldsymbol{U}$ matrix projects up to T times each $\boldsymbol{h}_t$, T being the total sequence duration. In contrast to this, QRNNs alternate convolutional layers, applied in parallel across time-steps, with minimalist recurrent pooling functions that apply in parallel across channels. Despite having no trainable recurrent connections (i.e., removing $\boldsymbol{U}$), QRNNs proved to be very effective in different NLP tasks, in comparison to their LSTM counterpart, when enough QRNN layers are stacked (Bradbury et al., 2017). The first QRNN stage operates with a series of causal convolutional layers, which can process all input time-steps in parallel. We

formulate the convolutional projections as

$$\begin{aligned} \mathbf{Z} &= \tanh(\mathbf{W}_z * \mathbf{X}) \\ \mathbf{F} &= \sigma(\mathbf{W}_f * \mathbf{X}) \\ \mathbf{O} &= \sigma(\mathbf{W}_o * \mathbf{X}), \end{aligned} \tag{3.9}$$

where $\mathbf{W}_z$, $\mathbf{W}_f$ and $\mathbf{W}_o$ are convolutional filter banks that predict the cell activation, the forget gate activation, and the output gate activation, respectively. The sizes of these filter banks are $\mathbb{R}^{K \times H \times M}$, being K each kernel width, H the number of input channels, and M the number of convolutional kernels. Note that $\mathbf{Z}$, $\mathbf{F}$, and $\mathbf{O}$ are of dimension $T \times M$, thus all T time-steps are processed in parallel by each filter bank. The causality of each convolution operation is enforced by padding on the left side of the signal with $K - 1$ zeros. After obtaining these activations, we can pass them to the aforementioned recurrent pooling, defined as

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c_{t-1}} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t \\ \mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t, \end{aligned} \tag{3.10}$$

where $\odot$ denotes element-wise multiplication. Note that each vector, indexed by t, is a time-slice of each of the previously defined activation tensors $\mathbf{Z}$, $\mathbf{F}$, and $\mathbf{O}$. These operations are similar to the ones defined in LSTMs (equation 3.8), but note how we have no affine transformation between time-steps in this case. Hence, the key to QRNN efficiency is that all learnable parameters are conveyed in the parallel computation phase. Moreover, the QRNN offers more flexibility than LSTMs in what gates are modeled, as the authors proposed originally 3 possible modeling strategies (Bradbury et al., 2017). We work with the often used and effective forget-output configuration (*fo*-pooling), which is the one formulated in the equations 3.9 and 3.10. This is also the default configuration in the official implementation[3]. Fig. 3.10 shows a detailed diagram of a QRNN layer. There it is shown that the convolutional projections allow all operations to happen in parallel, as formulated in equation 3.9. On the other hand, recurrent pooling contains a time dependency that only requires element-wise vector interactions, as formulated in equation 3.10.

3.1.6 Sequence to Sequence Model

RNNs and their derivations are a good fit to process sequences. The formulation in equation 3.7 yields a hidden vector of features per input time-step, which shows that RNNs can transform an input sequence $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ into another sequence $(\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T)$, which in turn can result into a final sequence $(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_T)$ after some additional level of transformation. This can be done by plugging a fully connected layer that operates independently in time to adjust the dimension of our desired network outputs. However, this model imposes that the alignment between input and output sequences must be known ahead of time, and both must have the same length T. Sequence to sequence models overcome this limitation by using two different blocks that interact with two sequences of arbitrary lengths T and T' for input and output respectively.

The first block is denominated encoder, which consumes the input sequence $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ to yield a single final vector $\mathbf{c} \in \mathbb{R}^H$ that summarizes the full contents of the input sequence. Normally a RNN handles this task due to its capacity to find interactions among elements in the sequence as time-steps go forward. Hence,

[3] https://github.com/salesforce/pytorch-qrnn

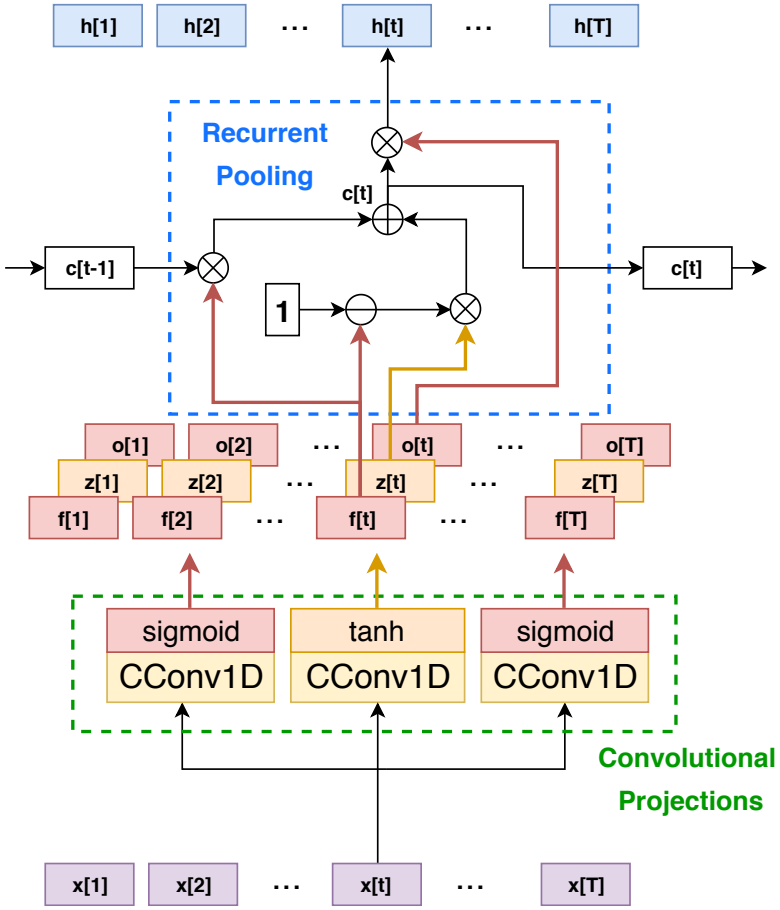

FIGURE 3.10: Schematic of a QRNN layer, following the operations of equations 3.9 for the convolutional projections and 3.10 for the recurrent pooling. This corresponds to the *fo*-pooling type from the original QRNN proposal (Bradbury et al., 2017).

the final state of the encoder RNN layer after processing the whole input sequence, $\mathbf{h}^e_T$, can be taken as the summary vector, such that $\mathbf{c} = \mathbf{h}^e_T$ (Sutskever et al., 2014; Cho et al., 2014).

The second block is the decoder, and it takes this representation $\mathbf{c}$ and generates the target sequence $(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_T)$ autoregressively. The concrete formulation for this operation is

$$\mathbf{h}^d_n = f(\mathbf{h}^d_{n-1}, \mathbf{y}_{n-1}, \mathbf{c}) \tag{3.11}$$

where $\mathbf{h}^d_n$ is the current time-step n hidden activation of the decoder RNN, $\mathbf{h}^d_{n-1}$ is the one from the previous time-step, $\mathbf{y}_{n-1}$ is the previous network prediction (after additional transformations on top of the RNN) and f is the RNN layer itself operating with the different inputs. Fig. 3.11 depicts this process as in the proposal by Cho et al. (2014).

3.1.7 Attention Layer

An attention layer or function relates the importance of a set of features with respect to another set. As stated in Vaswani et al. (2017), this function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors of features. Each output is computed as a

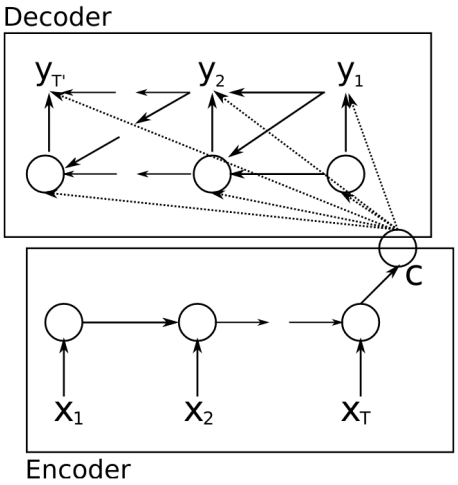

FIGURE 3.11: RNN based encoder-decoder model (from Cho et al. (2014)).

weighted sum of the values, and the weight assigned to each value is computed by a compatibility function of the query with the matching key (Vaswani et al., 2017). In the context of this thesis, this compatibility function takes the form of a dot product, as proposed by Vaswani et al. (2017). Hence assuming we have sequences of queries and keys of dimension d_k each, and values of dimension d_v, we define the scaled dot-product attention function Δ as

$$\Delta(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \sigma\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3.12}$$

where $\mathbf{Q} \in \mathbb{R}^{T' \times d_k}$, $\mathbf{K} \in \mathbb{R}^{T \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{T \times d_v}$. The parameters T' and T' denote the lengths of the different sequences of features. The outcome of the dot product between $\mathbf{Q}$ and $\mathbf{K}$ is the attention map, which is weighted by a softmax function σ to convert it to a PDF. Then, the resulting attention weights are applied over $\mathbf{V}$ to get the final sequence.

Attention models were introduced to improve sequence-to-sequence models on neural machine translation (Bahdanau et al., 2015). This component relieves the encoder from performing huge compressions of information (specially when the input sentence gets long) when building the summary vector $\mathbf{c}$ introduced in section 3.1.6, and allowed the decoder to learn to select the important parts of the encoder sequence to predict the next token in the output $\mathbf{y}_t$, following equation 3.11. In this case, considering a sequence of encoder hidden states as $(\mathbf{h}_1^e, \mathbf{h}_2^e, \cdots, \mathbf{h}_T^e)$ and the previous decoder output $\mathbf{y}_{n-1}$, we can construct an attention map when using $\mathbf{y}_{n-1}$ as a query. Then, each encoder state $\mathbf{h}_\tau^e$ will serve as a key, and the compatibility function of our choice (e.g. scaled dot product as in equation 3.12) will score their similarity. Finally, each encoder state will also act as a value, that will in turn be weighted by the compatibility score between the query and the key. In other words, the previous output $\mathbf{y}_{n-1}$ specifies what encoder states are more relevant by means of the compatibility function to weight the encoder states and use them to predict $\mathbf{y}_n$. Fig. 3.12 exemplifies the resulting grid that we may obtain in the attention responses (of size $T \times T'$), after we obtain our T' outputs. There, every column represents the previous output (query), and the lighter and more yellowish squares (weights from the compatibility function) are the most important ones for the next prediction. As such, the resulting thought vector at that instant is the dot product between the

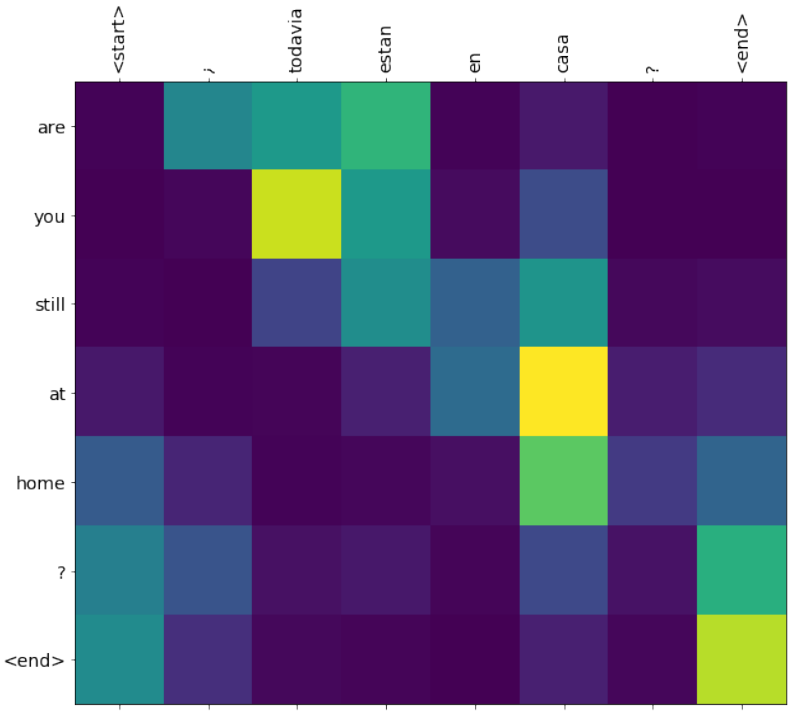

FIGURE 3.12: Example attention map that relates two sentences for machine translation, with an input English sentence "Are you still home?" and an output Spanish sentence "¿Todavía estan en casa?" (from TensorFlow neural machine translation with attention tutorial[4]).

weights and their corresponding encoder states (values). Note that the sequence-to-sequence attention operates autoregressively, due to the nature of the decoder module. Nonetheless, equation 3.12 contemplates operating on full sequences at once arranged in the matrices $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$. This means we can relate full sequences of arbitrary lengths to learn relations between them, and this includes the possibility of finding intra-sequence relations if both sequences are the same sequence. This specific application of the attention model is named self-attention, introduced in Vaswani et al. (2017) to replace the recurrent modules in the sequence-to-sequence architecture with something that still finds correlations within the sequence, but has no expensive recurrent connections.

Since their first introduction, attention models have been extensively used in fields like computer vision (Lu et al., 2017), speech recognition (Chan et al., 2016), speech synthesis (Wang et al., 2017; Pascual et al., 2019a), speaker recognition (India et al., 2019) and language modeling (Devlin et al., 2019; Irie et al., 2019), among others. In Vaswani et al. (2017) it was found beneficial to include not only one attention block, but a group of them arranged in parallel to build the so called multi-head attention (MHA; Vaswani et al., 2017), formulated as

$$\begin{aligned} MHA(\mathbf{Q},\mathbf{K},\mathbf{V}) &= C(H_1,...,H_h)\cdot\mathbf{W}^O \\ H_i &= \Delta(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V), \end{aligned} \tag{3.13}$$

with projection matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{model}\times d_v}$ and $\mathbf{W}^O \in \mathbb{R}^{hd_v\times d_{model}}$. Hence each of the 3 branches (query, key and value) contains a projection prior to the attention, and we can have up to h different attention paths.

[4]https://www.tensorflow.org/tutorials/text/nmt_with_attention

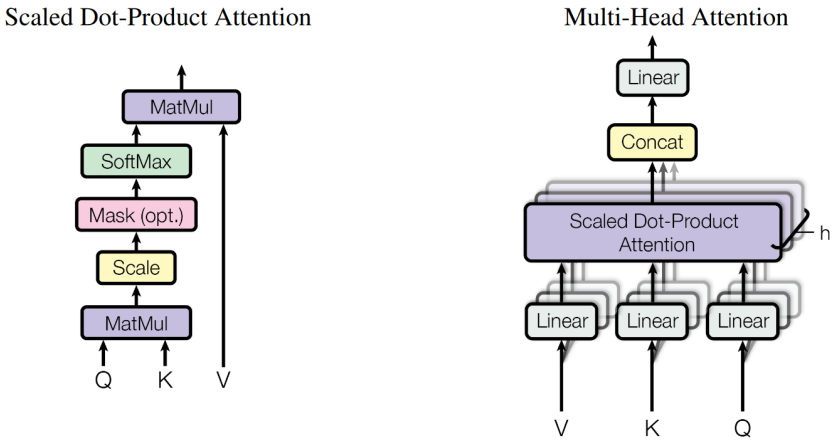

FIGURE 3.13: Left: Scaled dot-product attention function diagram. Right: multi-head attention consisting of h parallel attention functions (from Vaswani et al. (2017) Fig. 2).

Then all of them are merged back into a single representation with a concatenation operation C and a projection through $\mathbf{W}^O$. In other words, MHA applies h parallel attention layers, which can have a more versatile feature extraction by allowing the model to jointly attend to information from different representation subspaces at different positions (Vaswani et al., 2017).

3.2 Deep Generative Models

In this work, the terms generative model and deep generative model are interchangeable. This term, following the definition of Goodfellow (2016), refers to a model that takes a training set, consisting of samples drawn from a distribution p_{data}, and learns to represent an estimate of that distribution, resulting in a probability distribution p_{model}. Depending on the generative model we build, it estimates p_{model} explicitly or implicitly. Explicitly corresponds to directly parameterizing p_{model}. Implicitly corresponds to directly sample from p_{model}, without knowing its parameterization. What follows is a review of the state of the art deep generative models related to this work, as well as their taxonomy within the deep generative framework.

3.2.1 Explicit Density Models

The first type of generative models of great success in several applications are defined as explicit density functions $p_{model}(\mathbf{x}, \theta)$. In this notation, $\mathbf{x}$ refers to the data modeled and θ refers to the model parameters. For these models, maximization of the likelihood is straightforward, as model definition of the PDF is inserted into the likelihood expression and the negative log-likelihood gradient is followed downhill. Nonetheless, the main difficulty with explicit densities is designing a model that can capture all the data complexity, with inter-dimension dependencies, and still maintain a tractable computational approach. There are two main strategies used to tackle this challenging design: (1) careful construction of models whose structure guarantees their tractability, and (2) models that admit tractable approximations to the likelihood and its gradients (Goodfellow, 2016). In the former type of strategy we find mainly two models as the state of the art: autoregressive models and flow-based models. In the latter type we find variational autoencoders, which are based on variational methods.

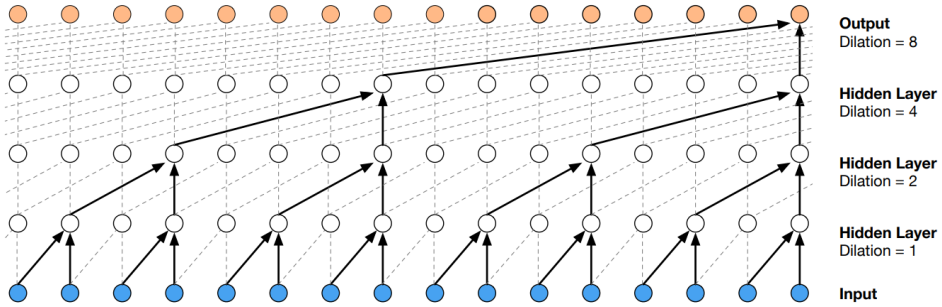

FIGURE 3.14: Schematic of the deep causal convolutional stack that forms the core of WaveNet (from van den Oord et al. (2016b) Fig. 3).

3.2.1.1 Autoregressive models

A joint probability distribution can be effectively and tractably modelled as a product of conditional distributions, following the probability chain rule. Fully visible belief networks (FVBFs Neal, 1992; Frey et al., 1996) are models that use this rule to decompose the probability distribution over a T-dimensional vector $\mathbf{x}$ as an autoregressive prediction such that

$$p_{model}(\mathbf{x}) = \prod_{t=1}^{T} p_{model}(x_t | x_1, ..., x_{t-1}). \tag{3.14}$$

FVBNs build the basis for the WaveNet model (presented in section 2.1), its derivations like the WaveRNN (Kalchbrenner et al., 2018), and its vision counterparts, the PixelRNN and the PixelCNN (van den Oord et al., 2016a). The core structure of the WaveNet and PixelCNN models is shown in Fig. 3.14. As noted, this is a stack of causal convolutional layers with growing dilation factors that increase the receptive field exponentially. This allows every prediction, formulated as each product in equation 3.14, to take a large context T into account to predict the next sample at time-step t. The main drawback of this modeling approach is the generation process, as it happens one entry at a time. In the case of WaveNet this gets prohibitively expensive due to its depth, which is around 60 layers, whereas models like WaveRNN try to reduce this by stacking less depth and working with subsequence generation, hence predicting in parallel downsampled sequences and merging in the end (Kalchbrenner et al., 2018). Nevertheless, this hits a limit of computational efficiency that can be surpassed only by non autoregressive models, where full parallelization is available.

3.2.1.2 Flow-based models

Flow-based generative models learn a bijective transformation from input samples $\mathbf{x} \in \mathbb{R}^T$ to latent representations $\mathbf{z} \in \mathbb{R}^T$ such that $\mathbf{z} = f(\mathbf{x})$ and $\mathbf{x} = f^{-1}(\mathbf{z})$. This invertible mapping f is called a normalizing flow (Rezende and Mohamed, 2015), and it can be parameterized with a neural network. This function can further be composed of k invertible transformations $f = f_1 \circ f_2 \circ ... \circ f_k$. The relationship tied between $\mathbf{x}$ and $\mathbf{z}$ is hence

$$\mathbf{x} \triangleq \mathbf{h}_0 \overset{f_1}{\longleftrightarrow} \mathbf{h}_1 \overset{f_2}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{f_k}{\longleftrightarrow} \mathbf{h}_k \triangleq \mathbf{z}. \tag{3.15}$$

We want to model the probability distribution $p(\mathbf{x})$ in order to build a generative model. The previous autoregressive approach makes this tractable at the expense of sampling efficiency (i.e. it is slow to generate new $\mathbf{x}$ samples). However, with the defined bijective transformation f, we can also model the exact log-likelihood as

$$L(\mathsf{X}) = \frac{1}{|\mathsf{X}|} \sum_{i=1}^{|\mathsf{X}|} \log\left(p\left(\mathbf{x}_i\right)\right). \tag{3.16}$$

For a single sample $\mathbf{x}$, with a change of variables following Rezende and Mohamed (2015), and applying the inverse function theorem, compositionality, and logarithm properties, we can write

$$p\left(\mathbf{z}\right) = p\left(f\left(\mathbf{x}\right)\right) = p\left(\mathbf{x}\right) \left|\det\left(\frac{\partial f^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right)\right|, \tag{3.17}$$

which is derived from the change of variables formula. By the inverse function theorem, we can work with the Jacobian of f,

$$p\left(\mathbf{z}\right) = p\left(\mathbf{x}\right) \left|\det\left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right)\right|^{-1} \tag{3.18}$$

and, taking logarithms and rearranging, we reach

$$\log\left(p\left(\mathbf{x}\right)\right) = \log\left(p\left(\mathbf{z}\right)\right) + \log\left|\det\left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right)\right|. \tag{3.19}$$

Finally, since f is a composite function (Sec. 3), we can write the previous equation as (Kingma and Dhariwal, 2018; Serrà et al., 2019):

$$\log\left(p\left(\mathbf{x}\right)\right) = \log\left(p\left(\mathbf{z}\right)\right) + \sum_{i=1}^{k} \log\left|\det\left(\frac{\partial f_i(\mathbf{h}_{i-1})}{\partial \mathbf{h}_{i-1}}\right)\right|. \tag{3.20}$$

In fact this is the expression used to optimize the normalizing flow. The density $p(\mathbf{x})$ is then tractable if the density $p(\mathbf{z})$ is tractable and the determinant of the Jacobian of f^{-1} is tractable. This means that a simple distribution over $\mathbf{z}$ combined with a transformation f that warps space in complicated ways can yield a complicated distribution over $\mathbf{x}$ (Goodfellow, 2016). This allows us to sample $\mathbf{z}$ values at one end and obtain new $\mathbf{x}$ values at the other end of the flow after all transforms are applied. And more importantly, we know that the exact likelihood criteria can be used to learn these transformations f if they are parameterized as neural networks, as it was the case with the earlier autoregressive decomposition.

Models with nonlinear functions f date back at least to Deco and Brauer (1995), with recent notable advances with systems like the RealNVP (Dinh et al., 2017), Glow (Kingma and Dhariwal, 2018), the parallel WaveNet (van den Oord et al., 2017), WaveGlow (Prenger et al., 2019) and Blow (Serrà et al., 2019). The RealNVP and Glow are applied to image generation. On the other hand, parallel WaveNet and WaveGlow are used in TTS applications, whereas Blow is a voice conversion system.

Finally, as stated in Goodfellow (2016), to avoid some of the disadvantages imposed by the design requirements of models with tractable PDF, other models were proposed that still provide an explicit PDF but use one that is intractable, requiring

the use of approximations to maximize the likelihood. With the use of deterministic approximations, this means using variational methods. These define a lower bound

$$L(\mathsf{X}) \leq \log p_{\text{model}}(\mathbf{x}). \tag{3.21}$$

A learning algorithm that maximizes L is guaranteed to obtain at least as high a value of the log-likelihood as it does of L (Goodfellow, 2016). For many families of models, we can define a tractable L even when the log-likelihood is not. Currently, the most prominent approach to variational learning is the variational autoencoder (Kingma and Welling, 2014).

3.2.2 Implicit Density Models

Implicit density models can be trained without the need to explicitly define a density function. Hence they typically offer a way to be trained by interacting indirectly with p_{model}, usually by sampling from it (Goodfellow, 2016). In this family of models, we have a very salient type nowadays that yields outstanding results: the generative adversarial network.

3.2.2.1 Generative Adversarial Networks

Generative adversarial networks (GANs) (Goodfellow et al., 2014) are generative models that learn to map samples $\mathbf{z}$ from some prior distribution $\mathcal{Z}$ to samples $\mathbf{x}$ from another distribution $\mathcal{X}$, which is the one of the training instances (e.g., images or audio). The component within the GAN structure that performs the mapping is called the generator network (G), and its main task is to learn a function whose outcomes can imitate some real data distribution. In this way, we can generate novel samples related to those of the training set. Importantly, G does so not by memorizing input-output pairs but by mapping the data distribution characteristics to the manifold defined in our prior $\mathcal{Z}$. Thus, there is an inherent stochastic component (in this case the sampling from $\mathcal{Z}$) that implies a different outcome for every generated prediction.

Adversarial training is the key component with which G learns to perform the aforementioned mapping. In this configuration, we have another component, called the discriminator network (D), which is typically a binary classifier. Its inputs are either real samples, coming from the dataset, or synthetic samples, entirely made up by G (which in turn imitates real samples). The adversarial characteristic comes from the fact that D has to classify the samples coming from $\mathcal{X}$ as real, whereas the samples coming from G, $\hat{\mathcal{X}}$, have to be classified as synthetic. This condition leads to G trying to fool D, and the way to do so is that G adapts its parameters such that D classifies the G output as real. During back-propagation, D improves at finding realistic features in its input; in turn, G corrects its parameters to move towards the real data manifold described by the training data (Fig. 3.15). This adversarial learning process is formulated as a minimax game between G and D, with the objective

$$\begin{aligned} \min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} \left[\log D(\mathbf{x})\right] + \\ + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})} \left[\log\left(1 - D\left(G\left(\mathbf{z}\right)\right)\right)\right]. \end{aligned} \tag{3.22}$$

The original GAN formulation of Goodfellow et al. (2014) included a hyperparameter that controlled an arbitrary amount of D update iterations k prior to updating G. This was done so that D could capture better features that would then be

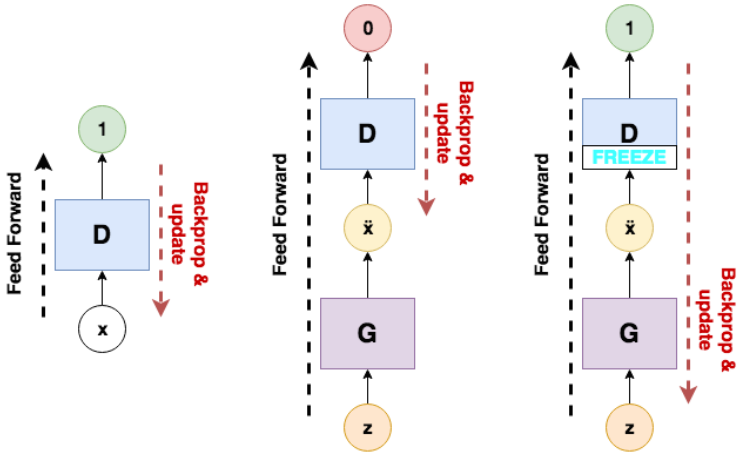

FIGURE 3.15: Schema of the GAN training process. First, D back-props a batch of real examples (left). Then, D back-props a batch of synthetic examples that come from G and classifies them as synthetic (middle). Finally, the D parameters are frozen, and G back-props to make D misclassify the examples (right).

leaked to G in the gradient flow. Nonetheless, this approach can be expensive, as it needs to make the forward-backward learning step per k-th iteration. The hyper-parameter k is tuned to stabilize the learning strategy as well as accelerating the generator convergence. Recently, this multiple batching update rule in the discriminator has been replaced by the so called two-timescale update rule (TTUR; Heusel et al., 2017), where the learning rate of the discriminator is higher than that of the generator (usually between 2 to 4 times). Importantly, this allows D to learn features in a quicker way per training step while operating a single forward-backward step per D update, hence being more efficient.

The previously described model would learn to generate novel samples that could resemble random real points, but it is often interesting to add a conditioning factor that can be used for a specific task. We can thus work with a conditioned version of GANs, where we have some additional information in G and D to perform mapping and classification (see Isola et al., 2017, and references therein). For instance in the speech enhancement task, we can condition the generation to a contaminated input utterance such that G has to output a clean version of it. The minimax game formulation is then modified to include a conditioning input vector $\tilde{\mathbf{x}}$:

$$\begin{aligned}\min_G \max_D V(D,G) = &\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}\left[\log D(\mathbf{x},\tilde{\mathbf{x}})\right] + \\ &+ \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}\left[\log\left(1 - D\left(G\left(\mathbf{z},\tilde{\mathbf{x}}\right),\tilde{\mathbf{x}}\right)\right)\right].\end{aligned} \tag{3.23}$$

Note that D is also receiving the conditioning vector $\tilde{\mathbf{x}}$ such that the information flowing back from D to G during training incorporates tied descriptions of both reality $\mathbf{x}$ (clean signal) and its conditioning reference $\tilde{\mathbf{x}}$ (noisy or corrupted signal).

There have been a number of improvements to the classifier structure of the discriminator to stabilize the overall adversarial training. These developments make D learn better features, with a better gradient flow in some cases relative to the classical formulation, which in turn improves the training of G, as it receives better error signals. The binary classification output in D can suffer from vanishing gradients due to the sigmoid cross-entropy loss used for training. To solve this problem, the least squares GAN (LSGAN) approach (Mao et al., 2017) replaces the cross-entropy loss with the least squares function and an output linear unit. The formulation in

equation 3.23 changes to

$$\begin{aligned}\min_G V(G) &= \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[(D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-c)^2],\\ \max_D V(D) &= \frac{1}{2}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}[(D(\mathbf{x},\tilde{\mathbf{x}})-b)^2]\\ &\quad+\frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[(D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-a)^2].\end{aligned} \tag{3.24}$$

where a, b, and c must fulfill the condition to minimize Pearson χ^2 divergence, i.e. $b-c=1$ and $b-a=2$ (Mao et al., 2017). In practice, we can select either ($a=-1, b=1, c=0$) or the binary encoding ($a=0, b=c=1$). The authors claim that both formulations reach similar results in practice (Mao et al., 2017).

Similarly to the development of LSGAN, other loss variants appeared to be used as stable counterparts to the vanilla GAN loss, like the Wasserstein GANs (Arjovsky et al., 2017; Gulrajani et al., 2017), the boundary equilibrium GAN (Berthelot et al., 2017) or the geometric GAN (Lim and Ye, 2017). Moreover, other mechanisms have been designed which are not not directly related to the loss functions but to bounding the gradient flow between the two networks (i.e. avoiding severe spiking shifts in the *G-D* learning equilibrium). Examples of this are both the gradient penalty (Gulrajani et al., 2017) and the spectral normalization (Miyato et al., 2018; Zhang et al., 2018a). Nonetheless, Lučić et al. (2018) conducted a neutral, multi-faceted large-scale empirical study on state-of-the art models and evaluation measures, finding that most GAN models can reach similar scores with enough hyper-parameter optimization and random restarts. They then suggest that further improvements in GANs performance may arise from a higher computational budget rather than from fundamental algorithmic changes. In fact, the recent work by Brock et al. (2019) demonstrates that GANs especially benefit from scaling, as well as from training models with 2 to 4 times as many parameters and 8 times the batch size compared to prior art.

Finally, it is worth mentioning a recent approach applied specifically to speech and audio generation. In this case, to increase the effectiveness of the generator learning process, a set of randomly sampled windows from real and fake speech signals are fed into an ensemble of discriminators. This seems to dramatically increase the synthetic samples quality, due to the data augmentation effect of the random sampling and the richness of features extracted by the ensemble of models (Bińkowski et al., 2019). This strategy proved so far to be effective under a high demand of resources, continuing the line of research of Brock et al. (2019).

Chapter 4

Efficient Neural Acoustic Modeling in Text-to-Speech

In the literature review we discussed the importance of sequential mechanisms to do speech synthesis. As such, recurrent neural networks (RNNs) are one of the most prominent architectures used in linguistic to prosodic and acoustic mapping. In section 2.1, we described how this happens for two-stage TTS (Chen et al., 1998; Fernandez et al., 2014; Zen and Sak, 2015), as well as for end-to-end TTS (Wang et al., 2017; Arik et al., 2017; Sotelo et al., 2017). In this chapter we investigate the performance in terms of both prediction efficiency and voice quality of three types of sequential processing architectures for linguistic–acoustic mapping in two-stage TTS systems. And we specifically focus our analysis on the linguistic–acoustic mapping as it tends to require fewer data and resources to be trained than end-to-end systems. Additionally, we can decouple the actual effects of acoustic modeling from the linguistic alignment in this way.

Despite the success of RNNs in the sequential modeling paradigm, recent investigations proved the effectiveness of other mechanisms that rely on structures different from the classic recurrent ones. Other recurrent-like variants were proposed to overcome the computation burden that projected recurrent connections impose. For instance, the quasi-RNN (QRNN), introduced in section 3.1.5, moves these projections to a feed forward case while maintaining a memory vector that gets updated with element-wise interactions (Bradbury et al., 2017). Hence, QRNNs may be used as a replacement for LSTMs. Their effectiveness in different NLP tasks compared to LSTMs has been demonstrated (Bradbury et al., 2017; Merity et al., 2018), with empirical evidence showing they can speed up sequence processing by 16 times with respect to LSTMs. They have also been used in end-to-end TTS systems, such as the aforementioned DeepVoice (Arik et al., 2017), as conditioning to processors to upsample linguistic and prosodic features to the waveform resolution used by a WaveNet generator. Another system is the Transformer network, which conceptualizes sequential dependencies differently from RNNs, avoiding any sort of recurrent connection. The Transformer network was designed as a sequence-to-sequence model with attention for machine translation (see sections 3.1.6 and 3.1.7), where typically RNNs are applied to deal with the conversion between the two sequences (Sutskever et al., 2014; Bahdanau et al., 2015). Vaswani et al. (2017) specifically introduced the self-attention mechanism described in section 3.1.7, which can relate elements within a single sequence regardless of the sequential order by using a compatibility function. Nonetheless, having an order awareness allows for better exploitation of the sequential context, and it is one of the main motivations for using RNNs rather than only self-attention modules to seek intra-sequential interactions. Thus, in Transformer, ordering is imposed with positioning codes that serve as

time-stamps. This makes input features determine the current state in the sequential context instead of having an intrinsic structure with feedback connections.

In our work we depart from an RNN reference model that uses LSTM cells, as proposed in previous works (Zen and Sak, 2015; Pascual, 2016). Then, we use a QRNN-based model named QRNN linguistic–acoustic decoder (QLAD), based on the previous mentioned evidence of more efficient computations than LSTM structures, albeit being comparable in performance across different sequential tasks. QRNNs have been used in the conditioning branch of end-to-end TTS, as mentioned above, but we instead propose them to actually generate the acoustic parameter trajectories of our decoder. We also consider a Transformer decoder without attention to any encoder, because we are dealing with a mapping between two sequences that have the same time resolution. Hence our decoder blocks only contain the self-attention and feed-forward network modules. Precisely a key difference between our Transformer module and the recently proposed Transformer TTS is that we do not work with the full Transformer structure as other end-to-end solutions do (Li et al., 2019). We thus call this system the self-attention linguistic acoustic–decoder (SALAD) (Pascual et al., 2018a). Acoustic models that improve the speech synthesis efficiency over the existing RNN-based ones are a suitable target for low resource environments as long as the generated speech quality is not degraded. This specially fits embedded devices, such as mobile phones, with which the TTS system can then run locally. Furthermore, the experimentation carried out in this work is applicable to further settings where text is not the input signal but speech is the synthesized output, e.g., speech enhancement or voice conversion (Sun et al., 2015; Erdogan et al., 2015), where speech signals are converted into other speech signals. Moreover, some of these architectural changes can also be extended for more end-to-end waveform generation solutions similar to those of WaveNet (van den Oord et al., 2016b) or WaveRNN (Kalchbrenner et al., 2018). Nonetheless, our focus is on exploring these variations on the vocoder acoustic parameter generation for ease of experimentation with respect to computational requirements, and as an incremental development upon our previous acoustic modeling works (Pascual, 2016).

In the following sections we find that both QLAD and SALAD models are competitive in terms of efficiency for training and inference. Additionally, both models show competitive objective results against those of the RNN. Nonetheless, QLAD models are the only ones that subjectively match, or even improve in some setting, the voice naturalness of RNNs.

4.1 Recurrent Linguistic–Acoustic Decoder

To study the introduction of our proposed networks into a TTS system, we employed our previous multiple speaker adaptation (MUSA) framework (Pascual and Bonafonte, 2016a; Pascual, 2016; Pascual and Bonafonte, 2016b). This is a two-stage RNN model influenced by the work of Zen and Sak (Zen and Sak, 2015), in the sense that it uses unidirectional LSTMs to build the duration model and the acoustic model without the need of predicting dynamic acoustic features. Therefore the RNN predicts a vocoder frame per time-step. The details on the vocoder used and its features are explained in section 4.4.2.

However, for the current work, we did not use this multiple speaker capability and focused on just single speaker models (one model per speaker) for the new architecture design on improving the acoustic model. This way we can decouple

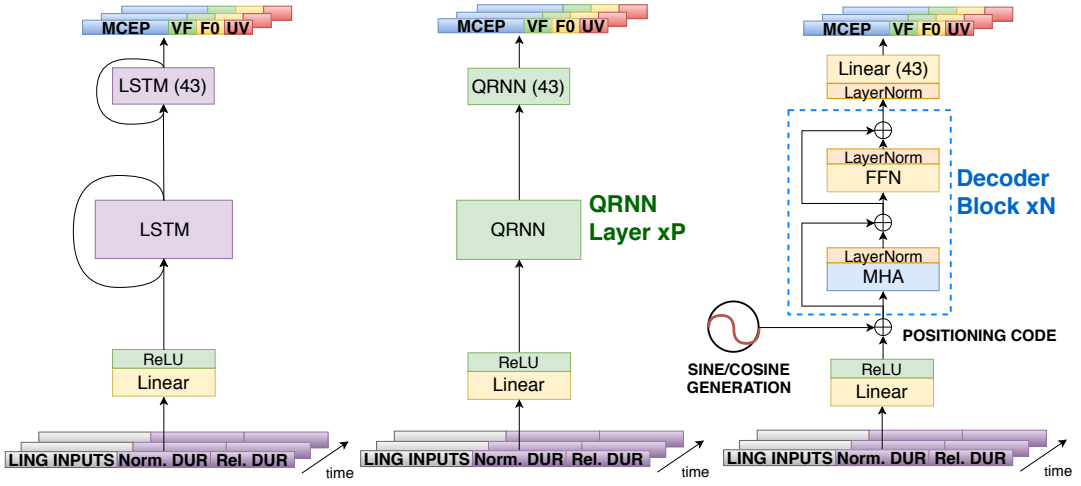

FIGURE 4.1: Comparison of architectures for: (**a**) RNN/LSTM acoustic model (RNN baseline); (**b**) QLAD; and (**c**) SALAD. The embedding projections are the same. The QRNN layer is stacked P times in the hidden structure. In SALAD, the positioning encoding introduces sequential information. The SALAD decoder block is stacked N times to form the hidden structure. FFN, Feed-forward Network; MHA, Multi-Head Attention; CConv1D, Causal 1D convolution.

architectural effects from shared representations, training each speaker model with a comparable amount of data and the same number of parameters.

Fig. 4.1 depicts the linguistic–acoustic decoder across the different structures explored in this work. The input to the model is a sequence of frames, each one containing a mixture of linguistic and duration features. Then, each structure outputs a frame of 43 acoustic parameters per time-step. Both linguistic and acoustic features used in this work are detailed in Section 4.4.2. The RNN baseline is shown in Fig. 4.1a. As a first step, we have a pre-projection, performed by a fully-connected layer with a ReLU activation. This embeds the mixture $\mathbf{x}_t$ of different input types into a common dense representation $\mathbf{h}_t$ in the form of one vector per time step t. Hence, the transformation $\mathbb{R}^L \rightarrow \mathbb{R}^H$ is applied independently at each time step t as

$$\mathbf{h}_t = \max(0, \mathbf{W}\mathbf{x}_t + \mathbf{b}), \tag{4.1}$$

where $\mathbf{W} \in \mathbb{R}^{H \times L}$, $\mathbf{b} \in \mathbb{R}^H$, $\mathbf{x}_t \in \mathbb{R}^L$, and $\mathbf{h}_t \in \mathbb{R}^H$. This is a 1×1 convolution as defined in section 3.1.2. After this projection, we have the recurrent core formed by an LSTM layer of H cells and an additional LSTM output layer. The output is recurrent, as this prompted better results than using dynamic features to smooth cepstral trajectories in time when using RNNs (Zen and Sak, 2015).

4.2 QRNN Linguistic–Acoustic Decoder

Quasi RNNs are introduced in section 3.1.5. As denoted there, they were proposed as an efficient alternative to RNNs, and they proved to be very effective in different NLP tasks, in comparison to their LSTM counterpart, when enough QRNN layers are stacked. We propose the substitution of LSTM layers with QRNNs, hence applying them after the embedding layer introduced in section 4.1. We hence have an input tensor $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_T] \in \mathbb{R}^{T \times H}$ containing the embeddings at all time-steps, and these are processed through a stack of P QRNN hidden layers and a QRNN output layer. This is depicted in Fig. 4.1b. A QRNN layer is contained within the green

dashed line, with its gate activations and their interactions described in section 3.1.5 (equations 3.9 and 3.10). Finally, analogous to the RNN version, a second QRNN is stacked as output layer. This architecture can thus be seen as an intermediate step between full recurrence and no recurrence at all, and we name it QLAD.

4.3 Self-Attention Linguistic–Acoustic Decoder

Based on the Transformer architecture (Vaswani et al., 2017), we also consider a pseudo-sequential processing network that can leverage distant element interactions within the input linguistic sequence to predict acoustic features (Pascual et al., 2018a; Pascual et al., 2019a). This is similar to what an RNN does, but discarding any recurrent connection. This allows us to process all input elements in parallel at inference, hence substantially accelerating the acoustic predictions. In our setup, we do not face a sequence-to-sequence problem as stated previously, so we only use a structure similar to the Transformer encoder, which we call a linguistic–acoustic decoder.

The SALAD architecture begins with the same embeddings of linguistic and prosodic features from Section 4.1, followed by a position encoding system. As we have no recurrent structure, and hence no processing order, this position encoding allows the upper parts of the network to locate their operating point in time, such that the network knows where it is inside the input sequence. Concretely following the proposal by Vaswani et al. (2017), a positioning code $\mathbf{c} \in \mathbb{R}^H$ is a combination of harmonic signals of varying frequency:

$$\begin{aligned} c_{t,2i} &= \sin\left(t/10000^{\frac{2i}{H}}\right) \\ c_{t,2i+1} &= \cos\left(t/10000^{\frac{2i}{H}}\right), \end{aligned} \tag{4.2}$$

where i represents each harmonic, and H is the embedding dimensionality. At each time-step t, we have a unique combination of signals that serves as a time stamp. We can expect this to generalize better to long sequences than having an incremental counter that marks the position relative to the beginning, owing to the cyclic nature of sine functions. Each time stamp $\mathbf{c}_t$ is added to each embedding $\mathbf{h}_t$, and this is input to the decoder core, which can leverage this ordering information even though it does not have recurrent connections.

The decoder core is built with a stack of N blocks, depicted within the dashed blue rectangle in Fig. 4.1c. These blocks are the same as the ones proposed in the encoder of Vaswani et al. (2017). The most salient part of this type of block is the multi-head attention (MHA) layer, introduced in section 3.1.7. We precisely introduced attention as a function that relates two sequences of feature vectors, one representing queries and another representing keys and values. A special type of attention is the one where the sequence of queries and the sequence of keys and values are the same, named self-attention (see section 3.1.7). Here we concretely use this variant, and the sequences processed by our MHA are two copies of the input. Hence the MHA block looks for relations among close and distant tokens within the same sequence, and returns sequences that contain information on intra-sequential dependencies. This is a similar process to the one followed by an RNN, where there is a temporal connectivity among hidden states, nonetheless it happens to be unsorted with self-attention and no feedback connection is needed.

Following with the block structure shown in Fig. 4.1c, a feed-forward network (FFN) comes right after the MHA. This FFN composed of two fully-connected layers.

The first layer expands the attended features into a higher dimension d_{ff}, and this gets projected again to the embedding dimensionality H. Finally, the output layer is a fully-connected dimension adapter such that it can convert the hidden dimensions H to the desired amount of acoustic outputs. In this case, we may slightly degrade the quality of predictions with this output topology, as recurrence helps in the output layer capturing better the dynamics of acoustic features. Nonetheless, this may suffice for our objective of having a highly parallelizable and competitive system. Finally, in the decoder block both MHA and FFN have residual connections (He et al., 2016) and normalization layers (Ba et al., 2016) to favour the gradient flow, as in the original Transformer (Vaswani et al., 2017).

Note that, in this model, contrary to QLAD, we do not even have a state vector $\boldsymbol{h}_t$ that carries dynamic features over time. As such, it can operate fully parallel during training and inference at all stages. Nonetheless, self-attention modules require computing large matrix dot products which depend on the sequence length. This may impose a limit in the inference speed and even a performance mismatch with the QRNN structures at a certain sequence length.

4.4 Experimental Setup

In this section we describe the dataset we use. Then, we describe the features used both in the input and the output of our models. The model details are also given, including their layer sizes, training strategies and optimization parameters. Finally evaluation metrics and results are presented and discussed.

4.4.1 Dataset

For the experiments presented in this section, we used utterances of speakers from the Technology and Corpora for Speech to Speech Translation (TC-STAR) project dataset (Bonafonte et al., 2006) following the experimental development of our previous works (Pascual and Bonafonte, 2016a; Pascual and Bonafonte, 2016c; Pascual, 2016). The purpose of these text sources is twofold: enrich the vocabulary and facilitate the selection of the sentences to achieve good prosodic and phonetic coverage. For this work, we chose the same male (M1) and female (F1) speakers as in our previous works (Pascual and Bonafonte, 2016a; Pascual, 2016; Pascual and Bonafonte, 2016b). Their data are balanced with approximately the following durations per split for both: 100 min for training, 15 min for validation, and 15 min for test.

4.4.2 Linguistic and Acoustic Features

The decoder maps linguistic and prosodic features into acoustic ones. This means that we must first extract the hand-crafted features out of the input text. These follow a context-dependent label format (Zen, 2006). Concretely, these features contain the phonetic transcription of few windowed phonemes, information about stressed syllables, position of the phoneme inside the current syllable, and position of the syllable in the word, among others. Each phoneme is encoded as a string containing the following symbols, described in detail in Table 4.1:

```
p1^p2$-$p3$+$p4$=$p5~p6_p7/A:a1_a2_a3/B:b1-b2-b3~b4-b5 ...
&b6-b7#b8-b9$b10-b11!b12-b13;b14-b15|b16/C:c1+c2+c3/D:d1_d2 ...
/E:e1+e2~e3+e4&e5+e6#e7+e8/F:f1_f2/G:g1_g2/H:h1=h2~h3=h4|h5 ...
/I:i1_i2/J:j1+j2-j3.
```

TABLE 4.1: Linguistic and prosodic features of the context-dependent label format.

label format	
Symbol	Description
p1	phoneme identity before the previous phoneme
p2	previous phoneme identity
p3	current phoneme identity
p4	next phoneme identity
p5	the phoneme after the next phoneme identity
p6	position of the current phoneme identity in the current syllable (forward)
p7	position of the current phoneme identity in the current syllable (backward)
a1	whether the previous syllable is stressed or not (0; not, 1: yes)
a2	whether the previous syllable is accented or not (0; not, 1: yes)
a3	number of phonemes in the previous syllable
b1	whether the current syllable stressed or not (0: not, 1: yes)
b2	whether the current syllable accented or not (0: not, 1: yes)
b3	the number of phonemes in the current syllable
b4	position of the current syllable in the current word (forward)
b5	position of the current syllable in the current word (backward)
b6	position of the current syllable in the current phrase(forward)
b7	position of the current syllable in the current phrase(backward)
b8	number of stressed syllables before the current syllable in the current phrase
b9	number of stressed syllables after the current syllable in the current phrase
b10	number of accented syllables before the current syllable in the current phrase
b11	number of accented syllables after the current syllable in the current phrase
b12	number of syllables from the previous stressed syllable to the current syllable
b13	number of syllables from the current syllable to the next stressed syllable
b14	number of syllables from the previous accented syllable to the current syllable
b15	number of syllables from the current syllable to the next accented syllable
b16	name of the vowel of the current syllable
c1	whether the next syllable stressed or not (0: not, 1:yes)
c2	whether the next syllable accented or not (0: not, 1:yes)
c3	the number of phonemes in the next syllable
d1	gpos (guess part-of-speech) of the previous word
d2	number of syllables in the previous word
e1	gpos (guess part-of-speech) of the current word
e2	number of syllables in the current word
e3	position of current word in the current phrase (forward)

e4	position of current word in the current phrase (backward)
e5	number of content words before the current word in the current phrase
e6	number of content words after the current word in the current phrase
e7	number of words from the previous content word to the current word
e8	number of words from the current word to the next content word
f1	gpos (guess part-of-speech) of the next word
f2	number of syllables in the previous word
g1	number of syllables in the previous phrase
g2	number of words in the previous phrase
h1	number of syllables in the current phrase
h2	number of words in the current phrase
h3	position of the current phrase in utterance (forward)
h4	position of the current phrase in utterance (backward)
h5	Phrase modality (question, exclamation, etc.)
i1	number of syllables in the next phrase
i2	number of words in the previous phrase
j1	number of syllables in this utterance
j2	number of words in this utterance
j3	number of phrases in this utterance

Each of the features in Table 4.1 is numerically encoded depending on whether it contains several discrete categories (i.e. one-hot codes), binary categories, or real values. Then real valued inputs are normalized to have zero mean and unit variance.

For an input sentence with L words, after the phonetic transcription we obtain $M \geq L$ phonetic units encoded into label vectors, each with 362 dimensions. To inject these into the acoustic decoder, we needed an extra step. As mentioned, the MUSA testbed follows a two-stage structure with the amount of frames specified in the first stage: (1) duration prediction; and (2) acoustic prediction. Here, we only worked with the acoustic mapping, thus we enforced the duration with labeled data. For this reason, we replicated the linguistic label vector of each phoneme as many times as dictated by the ground-truth annotated duration, appending two extra dimensions to the 362 existing ones. These two extra dimensions corresponded to: (1) absolute duration normalized between 0 and 1, given the training data; and (2) relative position of current phoneme inside the absolute duration, also normalized between 0 and 1.

We parameterized the speech with a vocoded representation using the Ahocoder from Erro et al. (2011). Ahocoder is a harmonic-plus-noise high quality vocoder, which converts each windowed waveform frame into three types of features: (1) mel cepstral coefficients (MCEP); (2) log-F0 contour; and (3) voicing frequency (VF). Note that F0 contours have two states: either they follow a continuous envelope for voiced sections of speech, or they are 0, for which the logarithm is undefined. Because of that, Ahocoder encodes this value with -10^9, to avoid numerically undefined values. This result would be a cumbersome output distribution to be predicted by a neural net using a quadratic regression loss. Therefore, to smooth the values out and normalize the log-F0 distribution, we linearly interpolated these contours and create an extra acoustic feature, the unvoiced-voiced flag (UV), which is the binary

TABLE 4.2: Different layer sizes of the different models and total number of model parameters. Emb/H: linear embedding layer size and SALAD hidden size (in all layers but FFN ones); RNN/QRNN: recurrent or quasi-recurrent layer size in the hidden layers; d_{ff}: dimension of the feed-forward hidden layer inside the FFN in SALAD models.

Model	**Emb/H**	**RNN/QRNN**	**d_{ff}**	**#Params**
Small RNN	128	450	-	1.17M
Small QLAD	128	360	-	1.01M
Small SALAD	128	-	1024	1.04M
Big RNN	512	1300	-	9.85M
Big QLAD	512	1150	-	10.04M
Big SALAD	512	-	2048	9.66M

flag indicating the voiced or unvoiced state of the current frame. We then had an acoustic vector with 40 MCEP, 1 log-F0, 1 VF, and 1 UV. This totaled 43 features per frame, where each frame window has a stride of 80 samples over the waveform. Real-numbered linguistic features were Z-normalized by computing statistics on the training data. In the acoustic feature outputs, all of them were normalized to fall within $[0, 1]$ following the speaker-dependent training data statistics.

4.4.3 Model Details and Training Setup

We had three main structures: the baseline RNN, QLAD, and SALAD. The RNN took the form of an LSTM network for their known advantages of avoiding typical vanilla RNN pitfalls in terms of vanishing memory and bad gradient flows (Hochreiter and Schmidhuber, 1997; Hochreiter et al., 2001). Each of the three different models had two configurations, small (Small RNN / Small QLAD / Small SALAD) and big (Big RNN / Big QLAD / Big SALAD). This was intended to show the performance difference with regard to speed and distortion between the proposed model and the baseline, but also their variability with respect to their capacity (RNN, QLAD, and SALAD models of the same capacity have an equivalent number of parameters although they have different connection topologies). Fig. 4.1 depicts all these models' structure, where only the size of their layers (LSTM, embedding, MHA, FFN and CConv1D) changes with the mentioned magnitude. Table 4.2 summarizes the different layer sizes for all types of models and magnitudes, as well as their total number of parameters.

Additionally, all models are trained with dropout (Srivastava et al., 2014) in certain parts of their structure to avoid over-fitting easily to the available training set per speaker. The RNN and QLAD models have it after the hidden LSTM or QRNN blocks. The SALAD model has many dropouts in different parts of its submodules, replicating the ones proposed in the original Transformer encoder (Vaswani et al., 2017). The RNN and QLAD dropouts are 0.5, and SALAD has a dropout of 0.1 in its attention components and 0.5 or 0.2 in FFN and after the positioning codes, depending on the model performance based on a validation criterion. Note that both QLAD and SALAD models also can have P and N number of blocks, respectively, which turned out to be $P = N = 3$ for all performed experiments.

All models were trained with batches of 32 sequences of 120 symbols. The training consisted of stateful-structured batches, such that we carried the sequential state between batches over time (that is, the memory state in the RNN or QRNN and the position code index in SALAD). To achieve this, we concatenated all training frames

into a very long sequence, and then chopped it into 32 long pieces. We then used a non-overlapped sliding window of size 120, so that each batch contained a piece per sequence, continuous with the previous batch. This made the models learn how to deal with sequences longer than 120 outside of train, learning to use a conditioning state different from zero in training. All models were trained for a maximum of 300 epochs, but they triggered a break by early-stopping with the validation data. The validation criterion for which they stop was the mel cepstral distortion (MCD, as discussed in Section 5.3) with a patience of 20 epochs.

Regarding the optimizers, we used Adam (Kingma and Ba, 2015) for the RNN and QLAD models, with the default parameters in `PyTorch` ($\lambda = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$). For SALAD, we used a variant of Adam with adaptive learning rate, already proposed in the Transformer work, called Noam (Vaswani et al., 2017). This optimizer is based on Adam with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$ and a learning rate scheduled as

$$\lambda = H^{-0.5} \cdot \min(s^{-0.5}, s \cdot w^{-1.5}) \tag{4.3}$$

where we have an increasing learning rate for w warmup training batches, decreasing afterwards proportionally to the inverse square root of the step number s (number of batches). We used $w = 4000$ in all experiments. The parameter H is the inner embedding size of SALAD, which was 128 or 512, depending on whether it was the small or big model, as noted in Table 4.2. We also tested Adam on the big version of SALAD, but we did not observe any improvement in the results, thus we stuck to Noam following the original Transformer setup.

4.4.4 Evaluation Metrics

What follows is a description of the 3 types of measures: (1) objective metrics, (2) subjective test and (3) efficiency. With these we compare the aforementioned models both in terms of synthesized voice quality and synthesis efficiency. In the case of objective metrics, they easily allow us to evaluate the small and big variants per model. Then, as the objective results and qualitative listenings by the authors and different colleagues suggest, we proceed with the big variant of each model to carry a subjective test due to their higher level of naturalness. Finally, the efficiency evaluation is also taken with the big variants, as we can then take the worst case scenario in consideration, where we have more parameters to do synthesis. This way we know small models should run faster due to their reduced set of parameters.

4.4.4.1 Objective Evaluation

To assess the distortion introduced in the synthetic speech by the different models, we took 3 different objective evaluation metrics that follow the same formulations as in our previous works (Pascual and Bonafonte, 2016a; Pascual, 2016; Pascual and Bonafonte, 2016b). First, we have the mel cepstral distortion (MCD) measured in decibels (Kubichek, 1993), which tells us the amount of distortion in the prediction of the spectral envelope, defined as

$$\text{MCD} = \frac{10\sqrt{2}}{T \ln 10} \sum_{t=0}^{T-1} \sqrt{\sum_{n=0}^{39} (\kappa_{t,n} - \hat{\kappa}_{t,n})^2}, \tag{4.4}$$

where T are the amount of frames, $\kappa_{t,n}$ is each ground-truth cepstral component and $\hat{\kappa}_{t,n}$ the predicted one. Then, we have the root mean squared error (RMSE) of the F0

prediction in Hertz ($\hat{\phi}_{0t}$) for the same T amount of frames, defined as

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=0}^{T-1}(\phi_{0t} - \hat{\phi}_{0t})^2}. \tag{4.5}$$

Finally, as we introduced the binary flag that specifies which frames are voiced or unvoiced, we measured the unvoiced–voiced classification error (UV error) as the number of failed hits over total outcomes, where unvoiced and voiced classes are balanced by nature. These metrics are indicatives of convergence towards the overall speech components of each speaker. However, very low objective scores are not an indicator of a natural sounding voice. In fact, a model with increased variance in its acoustic predictions, which in turn increments speech naturalness, is an objectively inferior model (Henter et al., 2018). Hence, it is appropriate to run into a subjective evaluation, where human listeners can rate in a specific scale how natural does an utterance sound, either generated by the TTS or coming from the test set.

4.4.4.2 Subjective Evaluation

We designed a naturalness subjective test where 18 subjects listened to 32 utterances in total (four systems evaluated for each of the eight test utterances) and rated the mean opinion score (MOS), which ranges from 1 (totally unnatural) to 5 (completely natural). The four systems rated were the real test utterance and each of the different synthesized versions: the big RNN, QLAD, and SALAD. The big version of each model is taken due to the higher level of naturalness in the generated speech when compared to its small counterpart version. Both objective results, available in table 4.3, and qualitative listenings indicated this. The eight utterances were a subset selected randomly per subject out of a pool of 24 possible ones. The subjects were presented the four systems simultaneously and the order of the different systems was randomized such that no repetitive pattern was salient by an ordered position.

4.4.4.3 Efficiency Evaluation

To assess the speed of each model, we measured the wall clock inference time per test utterance three times, and then average these realizations per utterance. Measurements were taken both on CPU and GPU hardware setups. CPU inferences were all done on an Intel i7 processor with `PyTorch` framework version 1.0.0 (Paszke et al., 2017). This is a different setup than that of our previous work, so absolute measured timings differ mainly for library and hardware changes. Nevertheless, results are consistent in terms of relative speed improvements. GPU measurements were made on an NVIDIA GTX Titan X.

4.4.5 Results

Fig. 4.2 depicts the evolution of test distortions with respect to the available training set size for each model and speaker. Each training set portion is shown in percentage, such that 5% means 5 min of training data. We can observe how SALAD struggles in the low data regimes, where only 5%, 25% or 50% of the training set is available. This indicates that it might reach a more comparable performance if we used even more training data. Nonetheless, the recurrent-connected architectures are a better fit for these low data regimes, as they can potentially capture the sequential nature of the acoustic data easily by carrying the cell states forward. Finally, the 100% data

TABLE 4.3: Male (top) and female (bottom) objective results. Err, unvoiced–voiced classification error. Lower values are better.

Model	MCD [dB]	F0 [Hz]	Err [%]
Small RNN	5.18	13.64	5.1
Small QLAD	5.18	13.95	5.2
Small SALAD	5.92	16.33	6.2
Big RNN	5.15	13.58	5.1
Big QLAD	5.15	13.77	5.2
Big SALAD	5.43	14.56	5.5
Small RNN	4.63	15.11	3.2
Small QLAD	4.86	16.66	3.8
Small SALAD	5.25	20.15	3.6
Big RNN	4.73	15.44	3.1
Big QLAD	4.62	16.00	3.3
Big SALAD	4.79	16.02	3.2

regime distortion measurements are shown in table 4.3 for all models. All models seem to perform similarly by only looking at the objective results in this case, with the exception that SALAD models are the ones having systematically higher errors of log $F0$ prediction and MCD in some cases, specially for the male speaker. We can also see that increasing the amount of parameters specially boosts SALAD models, indicating that the sequential task might be more challenging for a structure that contains no recurrent structure.

In table 4.4, we show the results for the subjective evaluation. First, we can see a gap between the natural MOS value and the best model (big RNN) of about 1 point in the "both" case. This is usual for vocoder-based two-stage TTS, where usually both the vocoder lossy compression and the over-smoothing effect contribute to this for approximately half point each (Shen et al., 2018). Albeit objective results show comparable performance for QLAD and SALAD models with respect to RNNs, we can see remarkable evidence in the subjective results determining SALAD to yield a worse performing synthesis in terms of naturalness. The SALAD MOS degradation is potentially provoked by mistakes in the predicted temporal dynamics of the acoustic contours, as we can easily hear prosodic defects or barely intelligible sections in a preliminary listening test. In addition, the non-recurrent output layer degrades frame transition quality, even when positioning codes are available. On the other hand, the big QLAD model is very close (if not equivalent) to the big RNN one. The recurrent structure of QRNNs is then an important ingredient to match the RNN performance, but pre-projecting the activations all at once for all time-steps is also possible to speed up the synthesis and not degrade performance. Qualitative results in the form of audio samples are also available online (`http://veu.talp.cat/efftts/`).

We can see the inference speed profile of each system in Fig. 4.3. These inference speeds are shown for the different big models depending on the synthesized waveform length, for both CPU and GPU systems. Each dot represents an utterance inference at a certain utterance length. After we collected the dots, we can first see a linear growth in the computational cost for RNN and QRNN models depending on the sequence length. On the other hand, SALAD models grow quadratically.

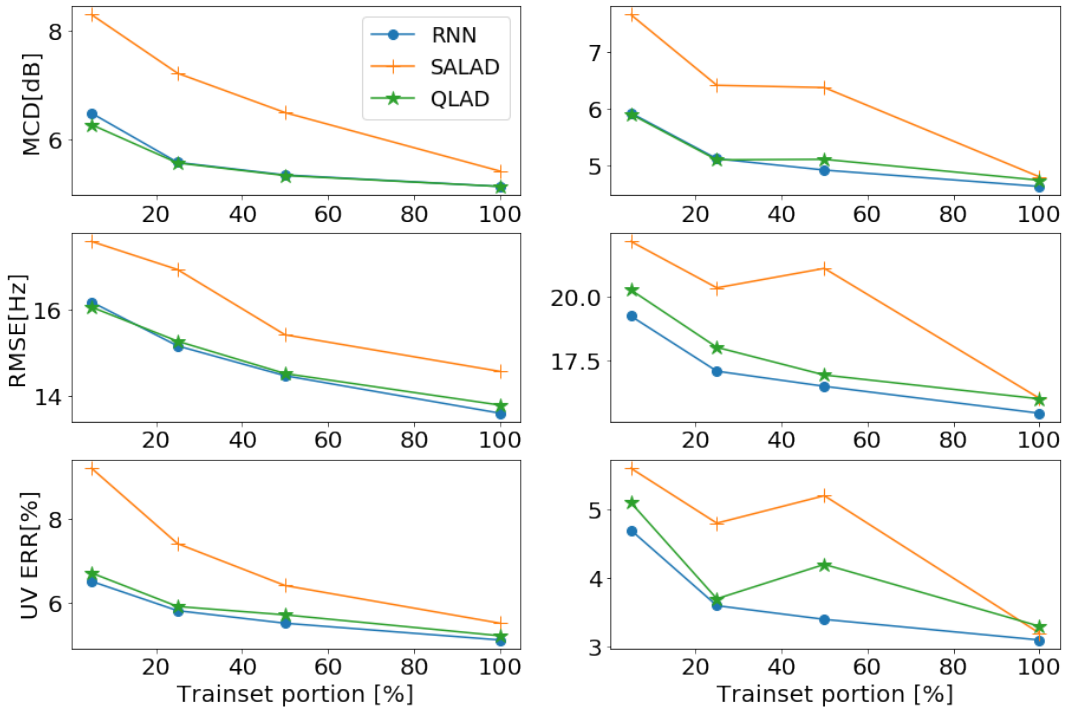

FIGURE 4.2: Evolution of test distortions depending on the available amount of training data (in percentage), for both male (**left**) and female (**right**) big models.

TABLE 4.4: Subjective mean opinion scores, for both speakers, male and female. Higher values are better.

	Natural	**Big RNN**	**Big QLAD**	**Big SALAD**
Both	4.67	3.53	3.52	2.12
Male	4.90	3.55	3.66	2.44
Female	4.37	3.49	3.34	1.71

This happens, as stated in Vaswani et al. (2017), because self-attention layers connect all the positions of the inputs sequences with a constant number of sequentially executed operations O(1), whereas a recurrent layer requires O(n) sequential operations. Then, self-attention layers are faster than recurrent layers when the sequence length is smaller than the representation dimensionality (H in Table 4.2). We confirmed these trends by performing linear regressions for RNN and QLAD models and then quadratic regressions for SALAD models (with `Scikit-learn` algorithms (Pedregosa et al., 2011)). Each model line shows the latency uprise trend with the generated utterance length up to 45 s. They also confirm the aforementioned linear and quadratic behaviors, which follow the observations by Vaswani et al. (2017). Interestingly, and as expected, the RNN model is slower on both CPU and GPU than the other two models on average. Especially on CPU, it can be seen to be slower regardless of the signal length, however on GPU it is rather equivalent to the other two models under the 10 s regime. After that, the inference time significantly increases compared to the other pseudo-recurrent models.

Moreover, even though SALAD seems to be faster on GPU than QLAD, they reach a crossing point when they generate utterances of approximately 45 s. At this point, both models are between 3 and 3.3 times faster than the RNN, as shown in Table 4.5. On the CPU, however, QLAD models are the fastest ones, mainly due to

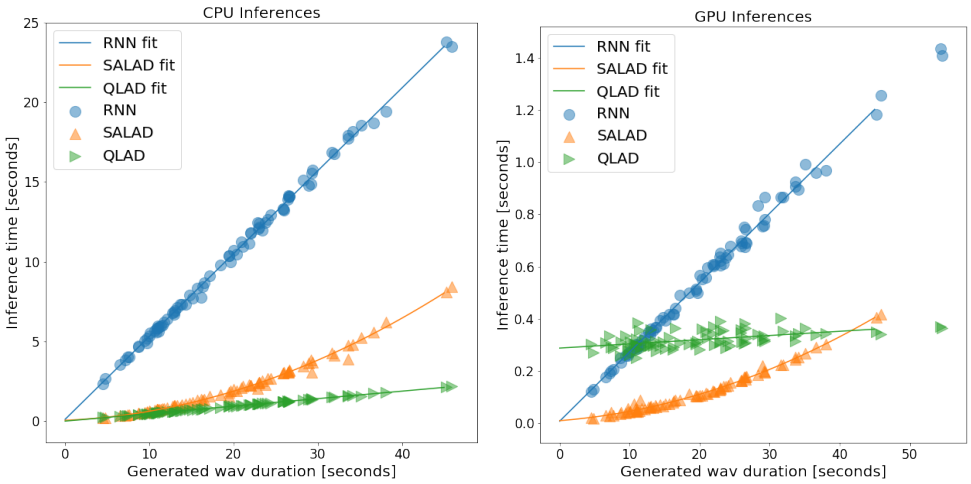

FIGURE 4.3: Inference time for the three different studied models: (**Left**) CPU speed curves; and (**Right**) GPU speed curves.

TABLE 4.5: Maximum inference latency with linear fit for RNN and QLAD models, and quadratic fit for SALAD. All measurements are taken at 45 s utterances.

Model	**Max. CPU Latency [s]**	**Max. GPU Latency [s]**
Big RNN	23.52	1.2
Big QLAD	2.09	0.36
Big SALAD	8.03	0.40

cheaper matrix operations with small convolutions of kernels of length one. They are thus 3.8 times faster than SALAD and 11.3 times faster than RNN on CPU, which are non-negligible speedups. Again, earlier results on QRNNs used in NLP showed their empirical speed advantage over vanilla RNNs (Bradbury et al., 2017), something we can also verify in this work for a continuous acoustic data prediction task.

4.5 Discussion and Conclusion

In this chapter, we present two competitive and fast acoustic model replacements for our RNN TTS baseline. The first proposal, QLAD, is a quasi-recurrent neural network that moves the computational burden of learnable feedback connections to feed-forward ones, while maintaining a simpler recurrent pooling. This allows a comparable modeling of temporal dynamics to that of LSTMs for the speech synthesis task. SALAD is based on the Transformer network, where self-attention modules build a global reasoning within the sequence of linguistic tokens to come up with the acoustic outcomes. In this case, positioning codes ensure the ordered processing in substitution of the ordered injection of features that RNN has intrinsic to its topology. Both systems converge to comparative amounts of distortion compared to the RNN baseline. However, only QLAD reaches the same level of subjective naturalness. On the other hand, both QLAD and SALAD are more efficient than LSTMs in large utterance generation regimes, in both CPU and GPU contexts. More specifically, with the proposed QLAD acoustic model, we achieve a 11.2 times speedup on CPU and 3.3 times on GPU with respect to the LSTM based model. Based on this evidence, we can conclude that QLAD is a high quality and effective replacement

for the LSTM based linguistic–acoustic decoder, even surpassing the performance and speed of a non-recurrent model such as SALAD.

In the literature review WaveNet was presented as an effective deep generative model that synthesizes highly natural waveforms (van den Oord et al., 2016b). Moreover, it can be conditioned on linguistic features to build a TTS system, although it is not efficient due to its autoregressive structure (Kalchbrenner et al., 2018). In the next chapter we explore the possibilities of generative adversarial networks, presented in section 3.2.2.1, to build an end-to-end system that is efficient and exportable to transform speech. The latter means it is as parallelizable as possible, built upon convolutional structures, and takes one inference step to operate in its task, contrarily to autoregressive models that take T steps to predict T samples. The task we choose is speech enhancement, where we can build a one-shot inference that compresses an input speech signal into a hidden representation and converts it into a speech signal of another domain. In this case, the target domain will be the one of clean signals. To that end, we can take advantage of learnable decimations with an encoder block, and interpolations with a decoder block. This ensures our model will operate faster than real-time thanks to reducing time-resolution for the convolution operations, effectively transforming a full utterance of noisy speech into clean speech in parallel. This task in turn also facilitates the first explorations we do on making an end-to-end model based on an encoder-decoder structure, where the input and output share the low level structure of speech, as well as the time resolution.

Chapter 5

Speech Enhancement Generative Adversarial Network

As discussed in the literature review, most of the current speech enhancement systems are based on the short-time Fourier analysis/synthesis framework, where only the spectral magnitude is treated to remove contaminating artifacts (Loizou, 2013). Recovering the signal is, in that case, a matter of recombining the cleaned-up magnitude with the input phase. This approach is common practice, as it is often claimed that short-time phase is not important for speech enhancement (Wang and Lim, 1982). Nonetheless, other studies show that significant improvements of speech quality are possible, particularly when a clean phase spectrum is known (Paliwal et al., 2011).

Generative adversarial networks (GANs; Goodfellow et al., 2014) are state-of-the-art generative models within the deep learning framework, as explained in the deep learning review (see section 3.2.2.1). In this chapter, we present a GAN for speech enhancement that works with the raw audio signal and aims at more generalized speech enhancement tasks (Pascual et al., 2017; Pascual et al., 2019b). The main advantages of the proposed speech enhancement GAN (SEGAN) are:

- It works end-to-end, with the raw audio. Therefore, no hand-crafted features are extracted and, with that, no explicit assumptions about the raw data are taken.
- It provides a quick enhancement process. No causality is required and, hence, there is no autoregressive operation like in RNNs or the WaveNet. This makes it an efficient alternative as an end-to-end and speech-to-speech system, compared to other deep generative models that could be applied.
- It learns from different speakers and noise types, and incorporates them together into the same shared parameterization. This makes the system simple and generalizable in those dimensions.

.

In section 5.1 we first describe our speech enhancement generative adversarial network (SEGAN) applied to denoising, together with an extensive exploration of variations that leads to an increase in performance and efficiency in section 5.3.1. We find that two key important variations that improve the SEGAN model are the introduction of learnable skip connections and the reduction of the architecture size from its base design by means of larger convolutional strides, which in turn increases the adversarial training stability. We then compare our approach to classic speech enhancement algorithms, such as Wiener filtering and statistical-based methods (Loizou, 2013), and to other deep neural networks working in the frequency

domain. The model transferability to other noises and languages is studied in section 5.4, together with its adaptive capacity to these new conditions unseen during training. Additionally, two novel applications of SEGAN are also presented that go beyond simple denoising and time-wise sample correspondence as a result of several changes in the GAN loss functions. We first substitute temporal regularization with spectral regularization. Second, we enforce content preservation with the addition of an extra adversarial signal. In section 5.5, these augmentations allow us to recover missing components of speech reconstructed with a motion capture device (Fagan et al., 2008), where excitation information is missing, hence sounding unnatural and whispered. We then use our modified model to convert this whispered speech into a more natural and voiced signal, so we call this modification a whispered-to-voiced conversion, applicable to the assistance of people lacking vocal folds, potentially after total laryngectomy surgery (Gonzalez et al., 2017a; Gonzalez et al., 2017b). Finally, in section 5.6 we extend SEGAN towards a more generalized speech enhancement task such that we recover cleaner speech out of severely distorted utterances (Pascual et al., 2019c). This generalization towards different speech distortions is interesting and directly applicable to modern communication technologies, where connections are interrupted (i.e. losing voice packets), and voice processing pipelines can distort our signals through amplifiers, compression of transmission encoders, etc. We emulate these effects by introducing four applied distortions, with different severity levels per distortion. We then propose the use of a new regression component on top of the discriminator that serves as an additional self-supervised task, boosting the generated recovered speech quality. This new task is specially effective when we use the second modification, a two-stage adversarial training schedule as a warm up and fine-tunning sequence.

5.1 Speech Enhancement GAN

In the enhancement problem, we have an input noisy signal $\tilde{\mathbf{x}}$, and we want to clean it, thus obtaining the enhanced signal $\hat{\mathbf{x}}$. Concretely, we first tackle the concrete problem of denoising, where the noisy signal is perturbed with additive background noise that has to be removed. In our configuration we have, for every noisy signal, its clean reference $\mathbf{x}$ during training. In this section we first describe the model we propose to perform the enhancement task. In section 5.1.2 we describe the procedure we follow to train the proposed model. Finally, in section 5.1.3 we describe the procedure to use the proposed model to perform the denoising.

5.1.1 Model

The proposed model follows the conditioned generative adversarial approach described in section 3.2.2.1. We call our model speech enhancement GAN, or SEGAN for short. The Generator Network G is structured as a deep convolutional autoencoder (Fig. 5.1) that first compresses the input waveform in time with the encoder and then reconstructs a plausible clean version of it with the decoder. Compression is done to discourage learning the identity function in the reconstruction of $\tilde{\mathbf{x}}$. In fact, our first attempts with non-decimating architectures yielded bad results, in terms of not eliminating the noise. Additionally, this autoencoder design also accelerates the convolution operations in the decimated parts of the structure (shorter sequence lengths involve faster processing) and can reduce the memory footprint if we use a small number of feature maps in the deepest part of the encoder.

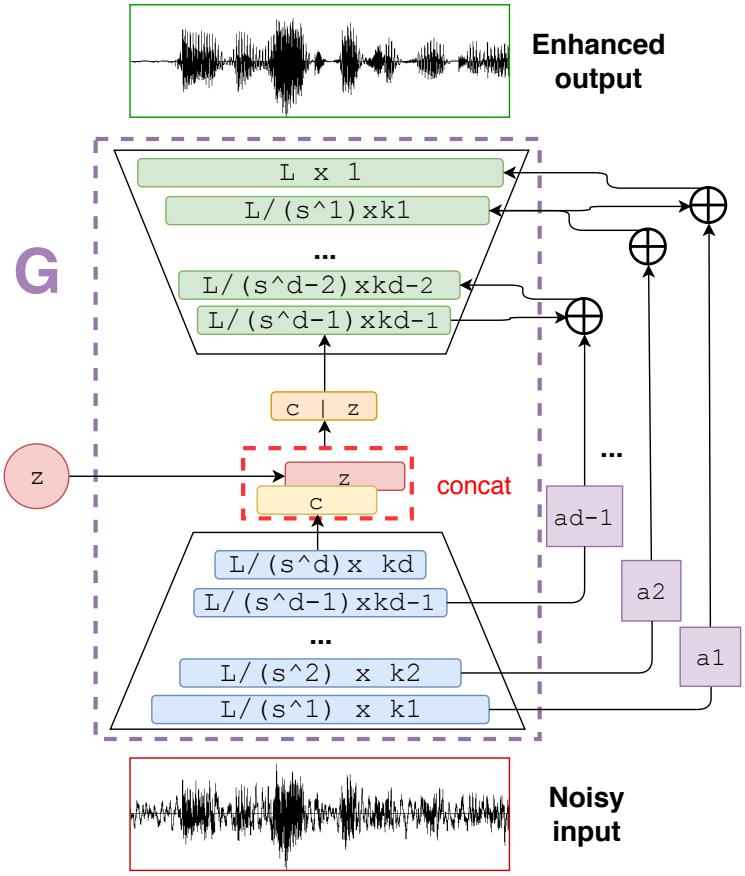

FIGURE 5.1: Autoencoder architecture for speech enhancement (G network). Feature maps are depicted in blue and green. The decimation/interpolation factor s^d depends on the stride s and layer depth index d. The input waveform length is designated L, and the number of kernels/channels at each layer is k_d. The right-side arrows denote skip connections, which have a multiplicative scalar factor a_d.

The generator input is the noisy speech signal $\tilde{\mathbf{x}}$, which is projected into an intermediate representation (see Fig. 5.1). Its output is the enhanced version $\hat{\mathbf{x}} = G(\mathbf{z}, \tilde{\mathbf{x}})$. As the design of G is exclusively convolutional, there are no fully connected layers nor autoregressive connections. This condition encourages the network to focus on temporally close correlations in the input signal and throughout the whole forward process across layers, whilst allowing it to process any input signal duration (unlike fully connected layers). Additionally, we note that it is a fast way to perform forward operations, as we process the full signal with one forward operation through the whole G. This approach contrasts with that of autoregressive or RNN models as mentioned earlier, which cannot be parallelized when computing each time step.

In the generation stage, the input signal $\tilde{\mathbf{x}}$ is decimated and expanded feature-wise through a number of strided convolutional layers, followed by multiparametric rectified linear units (PReLUs; He et al., 2015), i.e., activation with a learnable slope per feature channel for negative values. The aim of using this activation is to make everything as learnable as possible to be optimized depending on the task. In fact, we observe a linearized activation (large slope for negative values) for those layers that are close to the waveform domain (i.e., input and output), whereas the middle parts of the auto-encoder behave more like ReLUs (i.e., close to zero slope). Other works also show this pattern for PReLUs in deep convolutional auto-encoder structures (Paszke et al., 2016). We choose strided convolutions as they are more stable

than other pooling approaches for well-known GAN configurations (Radford et al., 2016). Decimation is implemented until we obtain a condensed representation of a few time samples (in the form of vectors of features), commonly called the thought vector $\mathbf{c}$. This result is concatenated with the generative noise component $\mathbf{z}$, which adds stochastic behavior to the generator predictions $\hat{\mathbf{x}}$ (we use isotropic Gaussian noise for $\mathbf{z}$). The encoding process is reversed in the decoding stage by means of transposed convolutions (sometimes called deconvolutions), followed again by PReLUs. The only exception is the last layer, which has a tanh activation to ensure that the output range is between -1 and 1. This step is introduced for stability purposes as it avoids exploding gradients, owing to its activation saturating regions, as in deep convolutional GANs (Radford et al., 2016).

The generator G also features skip connections, linking each encoding convolutional layer output to its homologous decoding layer and bypassing the compression performed in the middle of the model (Fig. 5.1). Note that the first connection is added after at least one decimation level, hence still avoiding a copy of the original signal itself in a trivial solution. We do this because the input and output signals of the model share the same underlying structure of natural speech. We hypothesize that low-level details to reconstruct the speech waveform properly could be lost if we force all information to flow through the compression bottleneck. Skip connections can help in this scenario, directly passing the fine-grained information of the waveform to the decoding stage. Moreover, we observed that skip connections offer a better training behavior, as the gradients can flow deeper through the whole structure (He et al., 2016). Our skip connections contain a multiplicative scalar factor $a_{l,k}$ per signal channel k and layer l. Therefore, if we have $k = 16$ channels, after the first encoder layer ($l = 1$) we will have a vector $\mathbf{a}_1 \in \mathbb{R}^{16}$ of amplifying or attenuating factors. These $\mathbf{a}_l$ vectors are learned together with the whole convolutional structure. In this way, the scaling of every feature map is optimized for the end task. At the j-th decoder layer input, we merge the (scaled) l-th encoder layer with the $(j-1)$-th decoder layer responses, following either a summation,

$$\mathbf{h}_j = \mathbf{h}_{j-1} + \mathbf{a}_l \odot \mathbf{h}_l, \tag{5.1}$$

or a concatenation,

$$\mathbf{h}_j = [\mathbf{h}_{j-1}; \mathbf{a}_l \odot \mathbf{h}_l], \tag{5.2}$$

where $\mathbf{h}_j$ is the output of the j-th layer and $\odot$ is an element-wise product along channels. Concatenation gives us slightly better results, but summation can also be competitive and compelling to make the system work with computationally restricted resources, as it requires fewer feature maps than the other option (see section 5.3).

To complete the GAN structure, we have the discriminator network, which follows the same one-dimensional convolutional structure as the G encoder, hence matching the conventional topology of a convolutional classification network. However, there are a few differences from the G encoder: (1) the discriminator network provides two input channels, (2) it can use some form of batch normalization (Ioffe and Szegedy, 2015) before LeakyReLU nonlinearities of $\alpha = 0.3$, and (3) in the last activation layer, there is a one-dimensional convolution layer with a single filter of width 1 and stride 1. The latter (3) reduces the amount of parameters required for the final classification neuron, which is fully connected to all hidden activations with a linear behavior (no activation function in between). This aspect reduces the amount of required parameters in the last activation from $T \times 1024$ to T with a learnable weighting.

5.1.2 Training

With the generator G and the discriminator D, we then build the adversarial setup following the least squares GAN (LSGAN) formulation with binary coding introduced in section 3.2.2.1. This means that D leaks information to G during backpropagation of what is real and what is synthetic. This way, G can slightly correct its output waveform towards the realistic distribution, discarding the noisy signals as those are signaled to be synthetic. In this sense, D can be understood as learning a loss function for the G output to look real, so that the enhancement must remain faithful to the speech signal and eliminate all the surrounding noise as much as possible. However, in preliminary experiments, we found it convenient to add a secondary regression component to the loss of G to minimize the distance between its generations and the clean examples. This way, the adversarial component can add more fine-grained and realistic results to the regression component. Both losses together were more stable than the separated case.

We chose the L_1 norm to be our regularizer, as it has been proven to be effective in the image manipulation domain (Isola et al., 2017; Pathak et al., 2016). Therefore, taking G's adversarial equation 3.24 with binary coding (i.e. with $a = 0$), the G loss becomes

$$\begin{aligned} \min_G V(G) = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[(D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-1)^2] \\ + \lambda \left\| G(\mathbf{z},\tilde{\mathbf{x}}) - \mathbf{x} \right\|_1, \end{aligned} \tag{5.3}$$

where λ is a hyperparameter that controls the magnitude of the regression component. We set λ to 100 after observing a better minimization trend correlated with signal quality. If λ is set to a smaller value, the L_1 term oscillates erratically. If it is set to a larger value, G behaves as a simple regressor. At approximately 100, this regularization term helps stabilize the training and yields favorable results (which is expected on a purely signal denoising task). However, having an L_1 regularization term can be a limitation when we have misalignment in input/output pairs, as it forces every sample of the output to match with the corresponding sample of the input. We did not encounter this problem when removing additive noise, but we had to replace this term when dealing with speech reconstruction (see section 5.5).

In terms of input data for D, and contrasting to typical adversarial training configurations, our configuration does not check whether a chunk is real or synthetic. Instead, training works with pairs of chunks that make real or synthetic targets as follows: a real pair is composed by a clean and a noisy signal $(\mathbf{x},\tilde{\mathbf{x}})$ and a synthetic pair is composed by an enhanced and a noisy signal $(\hat{\mathbf{x}},\tilde{\mathbf{x}})$. This is why D needs a *stereo* input: it classifies the comparison between both chunks as being real or synthetic. This training process is depicted in Fig. 5.2.

The training data were obtained by sliding a window of 16,384 samples (approximately 1 s at 16 kHz) every 500 ms from every training waveform. To study the variations on our architecture, models train for 100 epochs with a batch size of 300. A validation set of another 300 segments with maximum variability (different speakers than those in training, different noises, and different SNRs) is used to find reasonable plateaus in COVL and SSNR metrics (section 5.2.3).

5.1.3 Generation

As G is a fully convolutional structure, we can forward any chunk size L at test or generation time, with the only restriction being that the size be a multiple of the decimation factor Δ of the encoder. This means that we have to pad sequences with

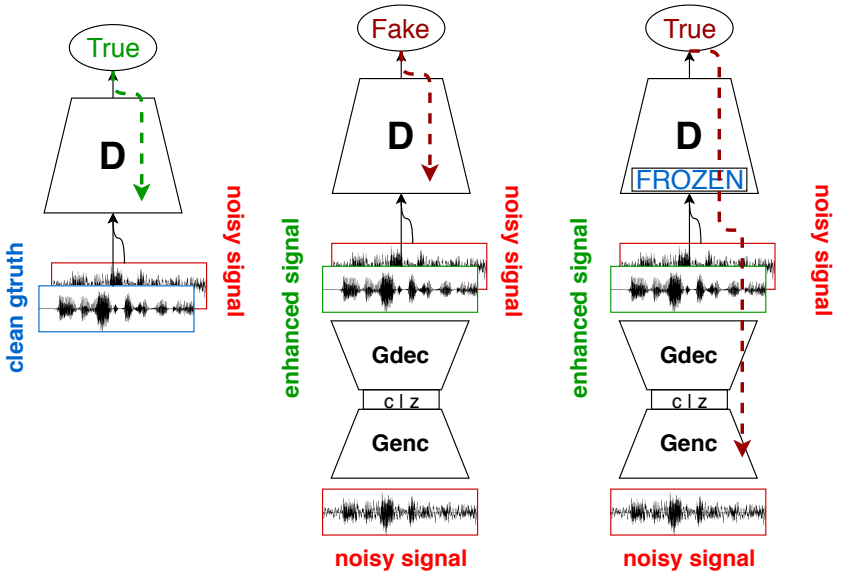

FIGURE 5.2: SEGAN training process. First, D backpropagates with the real pair $(\mathbf{x}, \tilde{\mathbf{x}})$ classifying it as real. Then, D backpropagates a fake pair $(\hat{\mathbf{x}}, \tilde{\mathbf{x}})$. Finally, D parameters are frozen and G backpropagates to make D miss-classify the pair $(\hat{\mathbf{x}}, \tilde{\mathbf{x}})$ as real. Gtruth: ground-truth clean signal. Genc: Generator encoder structure. Gdec: Generator decoder structure.

P zeros in some cases to fulfill $N = \frac{L+P}{\Delta} \in \mathbb{Z}$, so that we recover $L + P$ samples in the decoder output, to finally remove the leftover P values and retain our original L samples in our region of interest. When it comes to training, however, D has a fully connected output classification layer, which requires us to use fixed-size chunks.

At test/generation time, the difference between concatenating individually processed chunks of 1 second and processing any length T through G was objectively negligible. Hence, for long signals where intermediate network activations do not fit in memory (neither GPU's nor RAM), we can chunk without overlap, and by sliding G with the same $\mathbf{z}$ through the chunks, we can reconstruct with a concatenation.

5.2 Experimental Configuration

5.2.1 Data

To evaluate the effectiveness of our approach, we employ the clean speech in the VCTK Corpus (Veaux et al., 2017) and the noises from the Demand dataset (Thiemann et al., 2013), together with some extra synthesized noises following the structure and scripts of Valentini-Botinhao et al. (2016). We choose to generate these data ourselves, based on publicly available datasets with a massive amount of speakers because, this way, we can increase the pool of available speaker variability following the same SNR and noise variation structure as in Valentini-Botinhao et al. (2016). We have 109 available speakers in VCTK, out of which we split into 80 for training, 14 for validation, and 15 for testing. We force different speakers per split to study the generalization of the enhancement algorithm to unseen speakers.

To further lessen the intersection between splits, we run preprocessing to look for the least possible intersection in terms of textual contents between them. We find a total of 44,085 text files in the corpus, but with simple preprocessing, we obtain approximately 14,000 unique ones at the sentence level. We process the text files by lowercasing and eliminating any carriage return characters, punctuation signs,

and repeated spaces. This way, we obtain strings that can be compared literally among speakers (even though some strings can still have same spoken contents but slightly different text files, with some spontaneous missing determinants such as "the"). Based on this simple rule, we select the 15 speakers that have minimum text intersection with others for testing. The validation set is made of 14 speakers within the remaining pool of 94 available ones after test selection. We also retain gender balance to remain consistent in each subcorpus.

The noise conditions imposed with the abovementioned structure and scripts are 10 different noises with 4 SNR levels each for training and 5 different noises with other 4 SNR levels each for testing. The SNR conditions were $\{0, 5, 10, 15\}$ dB for training and $\{2.5, 7.5, 12.5, 17.5\}$ dB for testing. The noises used were (1) synthetic babble: many speakers in background; (2) real cafeteria: a busy office cafeteria; (3) real car: in a private passenger vehicle; (4) real kitchen: inside a kitchen preparing food; (5) real meeting; (6) real metro: a subway; (7) real restaurant; (8) synthetic ssn: white noise low-pass filtered; (9) real station; and (10) real traffic: a busy traffic intersection; for training/validation, and (1) real bus; (2) real cafe: the terrace of a cafe at a public square; (3) real living: inside a living room; (4) real office; and (5) real square: a public town square with tourists; for testing (noise types were randomly selected).

5.2.2 Baselines

We compare our model with two sets of baselines: (1) classic methods that do not require training parameters and (2) two deep learning methods that work in the spectral domain. Regarding the classic methods, we used a Wiener filter together with a statistical model based on the LogMMSE estimator. Both are taken from Loizou (2013). The deep learning methods are based on discriminative learnable nonlinear mappings. First we have models mapping noisy spectrum frames to clean spectrum frames. Based on Xu et al. (2015) and the modifications of the deep neural network baseline by Fu et al. (2017), we first build a deep neural network (DNN) with fully connected units, where we inject C input log-power spectral frames and obtain a single clean one. Consequently, we have a context window for which we clean up the central frame (Fig. 5.3). The structure of the network is a stack of 4 hidden layers of 1024 units each and an output layer that projects to the proper dimensionality to match the number of frequency bins F of our signal. We refer to this model as log-power spectrum DNN (LPS-DNN). Every hidden layer is a stack of an affine transformation, followed by a multiparametric PReLU activation and batch normalization. The FFT resolution remains for all experiments at 512, so that we address $F = 257$ bins. We then consider L_1 and L_2 losses and variations of C to check the regime of values in which we have a competitive baseline.

We also implement a bidirectional LSTM network for its known good modelling capacity for sequential problems like this (Maas et al., 2012; Weninger et al., 2015; Weninger et al., 2014; Erdogan et al., 2015). Our BLSTM has 650 cells, followed by a multi-layer perceptron to perform a final transformation and dimension adaptation. This last module's parameters are shared at all time-steps, and the hidden layer is of size 1024, following the LPS-DNN output structure. This network is designed to be comparable in terms of parameters to those of the LPS-DNN with $C = 1$, where only the sequential processing structure is changed. We also perform the L_1 and L_2 loss variations. We find that the LPS-BLSTM models require more careful tuning than the LPS-DNN given their tendency to stop learning because of gradient propagation issues if activations and gradients do not have the proper magnitudes (Hochreiter and

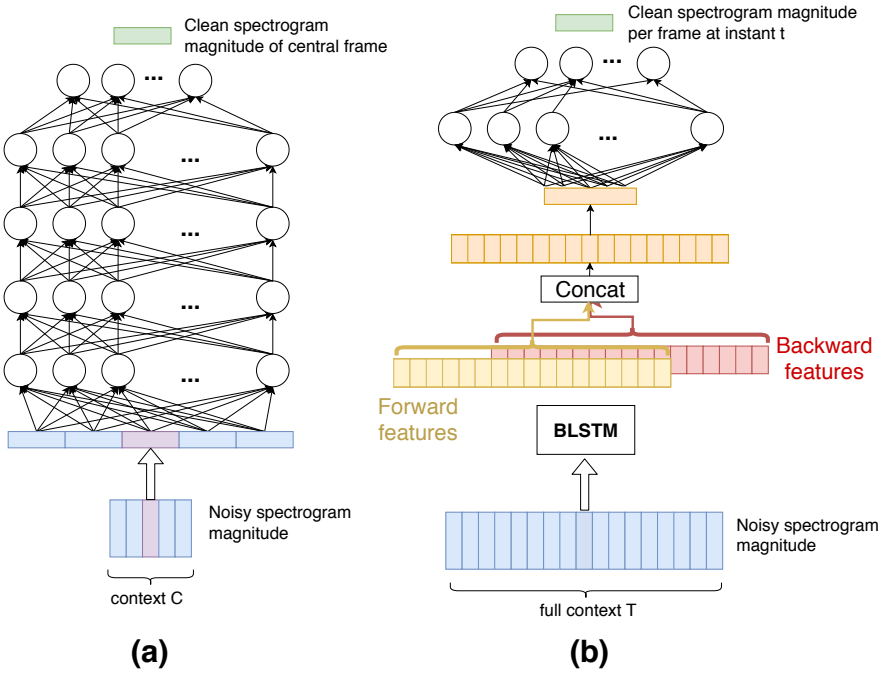

FIGURE 5.3: (a) LPS-DNN: Deep neural network baseline mapping of a context window of C log-power spectral frames to the central clean frame. (b) LPS-BLSTM: Bidirectional long-short term memory recurrent neural network that maps the full input noisy sequence to the clean one. The output of the BLSTM (forward and backward extracted features) is fed to an additional multi-layer perceptron to fullfil a final transformation and dimension adaptation sharing weights through time.

Schmidhuber, 1997). For this reason, we normalize the first two statistical moments of the inputs and apply gradient clipping to stabilize and promote the proper learning during back-propagation through time (Pascanu et al., 2013), which we observe to be beneficial in this case.

Finally, we also make a fully convolutional auto-encoder structure trained as a plain L_1 regression, hence decoupling the adversarial component from the system presented in section 5.1. The specific configuration of this model is the best one resulting from the ablation study performed in section 5.3.1, so that only the adversarial component is removed. This is an interesting way to assess the effect of the adversarial component for the considered tasks.

All models were trained with the Adam optimizer (Kingma and Ba, 2015) with default parameters as in `PyTorch` version 0.4.1 (Paszke et al., 2017). They are trained with all noise types, SNR conditions, and speakers (section 5.2.1). Note that these approaches do not use any supervision such as speaker identity or noise type. This way, we expect them to generalize to the different kinds of noises and speakers, which we also do with our model. These baselines are competitive counterparts to our waveform-based model. Specially the LPS ones as they work with more condensed information, with a prelocation step we perform that focuses on the spectral magnitude, where additive noise can be detected and removed easily.

5.2.3 Objective Metrics

We evaluate the quality of the enhanced speech with a set of well-known objective metrics, which serve as tools to obtain an estimation on how well the models work.

All of them compare the enhanced signal with the clean reference of 4,432 test set files. They have been computed with our Python reimplementation of the algorithms in Loizou (2013), which were available at the publisher website. The metrics, their meaning, and their range of values are as follows:

- PESQ: Perceptual evaluation of speech quality using the wide-band version recommended in ITU-T P.862.2 (ITU-T, 2007).
- CSIG: Mean opinion score (MOS) prediction of the signal distortion attending only to the speech signal (Hu and Loizou, 2008).
- CBAK: MOS prediction of the intrusiveness of noise (Hu and Loizou, 2008).
- COVL: MOS prediction of the overall effect (Hu and Loizou, 2008).
- SSNR: Segmental SNR (Scalart and Filho, 1996).
- STOI: Short-time objective intelligibility (Taal et al., 2010; Taal et al., 2011).

The PESQ metric ranges between -0.5 and 4.5. MOS regression metrics (CSIG, CBAK, and COVL) take values between 1 and 5. SSNR, in dB, is in the range $[-10, 35]$, as we trim it following the abovementioned implementation. STOI takes values in the range of 0 to 1. For all metrics, the higher the score is, the better the speech quality and the intelligibility.

5.3 Results

In the following, we make two blocks of analyses. In the first block, we conduct an ablation study of different SEGAN configurations and structures. The departing structure follows a simplistic design, where the generator includes conservative striding factors of 2 in each convolutional layer with kernel width 31. Moreover, its skip connections are non-learnable, hence behaving as identity functions. Additionally, the latent vector $\mathbf{z}$ is present unless the contrary is specified. The exact configuration of the convolutional filters is described down below for more detail. The discriminator features the same encoder structure (including filters, kernel width and strides) as G but with two input channels, as well as including batch normalization in each layer (Pascual et al., 2017). Note that D never has a $\mathbf{z}$ vector. The new configurations built upon this allow us to obtain an improved version of SEGAN, namely SEGAN+. In the second block of analyses, we make performance comparisons between SEGAN/SEGAN+ and baseline systems, which comprises both objective and subjective evaluations.

5.3.1 Model Variations

The variations introduced in the first block of experiments to improve SEGAN are the encoder/decoder stride size, encoder/decoder kernel size, optimizer, normalization schemes, enhancement with $\mathbf{z}$, and skip connections design. All variation results are shown in table 5.1, where model variants have an identifier that follows a tree development, from V1 (first SEGAN) toward the latest leaves in the V1.2.8.x level. We highlight the best node (SEGAN+) of the tree in bold. We follow these different node details in the following sections, where a set of takeaways emerge:

- It is important to have an aggressive decimation factor per encoder stage while maintaining a large kernel width. This approach allows for an efficient architecture that, as a result of the large receptive field, maintains satisfactory performance.
- Some normalization mechanism, either spectral normalization from Zhang et al. (2018a) (G and D) or batch normalization (D), is essential for the correct training of the system. Both increase performance in a similar way, allowing for better training gradient flows for a deep structure such as the one we consider.
- Learnable skip connections are a significant improvement on the G architecture. Using a scalar factor per hidden feature allows for importance filtering of feature maps from encoder to decoder. These are precisely the values $\mathbf{a}_l$ in equations 5.1 or 5.2, depending on whether we sum or concatenate feature maps. This approach provides better results with the same stability and similar gradient propagation as regular skip connections.
- The latent vector $\mathbf{z}$ is not clearly used as a generative component in noise removal, but we find that it helps as a regularization factor of G (without any additional requirements of dropout, batch normalization, or weight magnitude restrictions).

In the following subsections, we comment each variation in detail.

TABLE 5.1: Objective performance for different architecture variations. SEGAN is the first approach we developed in Pascual et al. (2017) but is evaluated over the new dataset (V1). SEGAN+ is the new best-performing model out of the different variations (V1.2.8). For both COVL and SSNR metrics, higher is better. Letters η and ω denote the learning rate and kernel width, respectively.

Model	Description	COVL	SSNR
V1 (SEGAN)	SEGAN first version with a stride of 2, a kernel width of 31, and batch norm in D.	2.77	5.15
V1.1	V1 made smaller and narrower, with a stride of 4 and a kernel width of 11.	2.58	4.38
V1.2	V1 made smaller with the same kernel sizes, with a stride of 4 and a kernel width of 31.	2.89	6.65
V1.2.1	V1.2 with spectral normalization in G and D and $\eta_G = 10^{-4}$, $\eta_D = 4 \cdot 10^{-4}$.	2.33	6.42
V1.2.2	V1.2 with spectral normalization in G and D and $\eta_G = 10^{-4}$, $\eta_D = 4 \cdot 10^{-4}$, and no batch norm in D.	2.72	6.56
V1.2.3	V1.2 with Adam and no batch norm in D.	2.10	3.35
V1.2.4	V1.2 with spectral normalization in G and D and $\eta_G = 10^{-4}$, $\eta_D = 4 \cdot 10^{-4}$, Adam, and no batch norm in D.	2.88	6.59
V1.2.5	V1.2 with no batch norm in D.	2.00	3.75
V1.2.6	V1.2 with Adam.	2.73	6.84
V1.2.7	V1.2 with convolutional skip connections.	2.73	3.30

V1.2.8 (SEGAN+)	**V1.2 with learnable scalar skip connections initialized at 1.**	**3.00**	**7.05**
V1.2.8.1	V1.2.8 with skip connections initialized at 0.	2.89	6.83
V1.2.8.2	V1.2.8 with summation merge of feature maps.	2.83	6.82
V1.2.8.3	V1.2.8 skipping post-activation feature maps.	2.90	6.04
V1.2.8.4	V1.2.8 without **z** vector.	2.20	4.19
V1.2.8.5	V1.2.8 without biases.	2.88	7.11
V1.2.8.6	V1.2.8 modifying kernel widths: $\omega_{\text{Genc}} = 31$, $\omega_{\text{Gdec}} = 4$, and $\omega_D = 31$.	2.61	6.37
V1.2.8.7	V1.2.8 modifying kernel widths: $\omega_{\text{Genc}} = 31$, $\omega_{\text{Gdec}} = 4$, $\omega_D = 31$, and no biases.	2.83	5.96

5.3.1.1 Encoder/decoder stride and kernel sizes

Section 5.1 introduces SEGAN as a flexible deep convolutional design. The first SEGAN proposal (V1) is composed of convolutions/deconvolutions of stride 2 and kernel width 31, as mentioned before (Pascual et al., 2017). The feature map configuration of the G network is as follows: 16384×1, 8192×16, 4096×32, 2048×32 , 1024×64, 512×64, 256×128, 128×128, 64×256, 32×256, 16×512, and 8×1024. This configuration is mirrored in the decoder to go back to 16384×1 resolution, with some possible doubled feature channels if we use concatenative skip connections (section 5.1).

One of the goals of our design is its speed, and decimation is a key factor to increase speed in a fully convolutional setup. An initial step we take is to reduce the size of the first model, hence increasing its computational efficiency, by means of increasing the stride factor from 2 to 4. After changing the stride to 4, we reduce the amount of layers from 22 to 10 such that we obtain the feature map structure 16384×1, 4096×64, 1024×128, 256×256, 64×512, and 16×1024 in the G encoder. Our first interest was reducing the model depth itself while maintaining the quality, but we found a quality increase, and we hypothesize that this factor might be related not only to less depth but also to the change in decimation, as we observe a change in high-frequency artifacts appearing in the G output with this structural change. Recently, the aliasing effect increasing with convolutional pooling was shown to degrade classification tasks with a waveform injected into the network (Gong and Poellabauer, 2018). Nonetheless, it remains unclear how this aliasing affects the quality of waveform generative models (Donahue et al., 2019), as it may be used to reconstruct missing frequency bands when upsampling from the latent space in GAN frameworks.

In this first level of experiments, we determine the effectiveness of increasing kernel stride in terms of both stability and performance, in addition to the degradations in performance associated with smaller kernel widths (V1.1 and V1.2, table 5.1). This condition also makes G more efficient, yielding a generation process that is 1.7 times faster than real time on a CPU and 17 times faster on a GPU.

5.3.1.2 Normalization schemes

After experimenting with different normalization schemes, we determine how important they are to stabilize the adversarial training. Hence, either batch normalization in D or spectral normalization in both networks (G and D) is required to obtain

training stability. Nonetheless, they should not be applied jointly because the performance would degrade. These observations are shown in results V1.2.1–6 (table 5.1), where we vary optimizers and normalization schemes.

Whenever we do not use any form of normalization (V1.2.3 and V1.2.5), we obtain more unstable results that lead to lower objective scores, particularly in terms of SSNR, as outputs are quite noisy. Hence, unless we use some form of normalization somewhere in the full GAN structure, either training diverges or the results are noisy and of poor quality. Moreover, we encountered no substantial difference between using virtual batch normalization as we did originally (Pascual et al., 2017) or plain batch normalization while reproducing SEGAN on the current data (V1). Thus, we use plain batch normalization for the sake of simplicity in all our current experiments. Spectral normalization, a promising technique for conditioned generator structures, is based on upper bounding gradient magnitudes (Zhang et al., 2018a). Nonetheless, we could not obtain a better result than the one we had with plain G and batch normalization in D.

5.3.1.3 Optimizer

We also find that both Adam and RMSprop optimizers (Tieleman and Hinton, 2012) are effective and yield a stable training across the different configurations with varying learning rates (V1.2.1–6, table 5.1). We depart from the optimizer configuration of V1, with small and balanced learning rates $\eta_G = \eta_D = 5 \cdot 10^{-5}$, and we also implement the recent two-timescale update rule (TTUR; Heusel et al., 2017). TTUR is a promising schedule to emulate a discriminator that is updated more often than the generator by simply applying a scaled ratio $\frac{\eta_D}{\eta_G} = 4$, thus ensuring $\eta_D = 4 \cdot 10^{-4}$ and $\eta_G = 10^{-4}$ (Heusel et al., 2017). Even though we achieve competitive results with TTUR (even better than those of V1), we discard them because the results are not better than that of V1.2. We therefore continue using RMSprop with $\eta_G = \eta_D = 5 \cdot 10^{-5}$.

5.3.1.4 Skip connections

We see that including skip connections with a simple learnable scalar boosts the performance of the system. Skip connections facilitate gradient flow, while learnable scalars are trained to determine which level of detail from the encoder layers is shuttled to the decoder layers. We also found that skip connections in G help to stabilize the training process (so much that if we try to train the system without them, it collapses). In V1, we have the simplest skip connections possible: they forward the feature map through an identity function to shuttle features, and gradients flow back and reach the deepest part of the structure. We decided to make them learnable such that the optimization process can weight their importance independently because they can act as pseudo-attention mechanisms of what levels of features are more important to be shuttled in the decoding process. Experiment V1.2.8 shows the effectiveness of this approach, surpassing the performance of V1.2.4 (table 5.1).

Following the finding that learnable skip connections can enable additional processing that helps the decoding stages, we also tried configuring them as convolutional layers of kernel width 11. This approach was expected to allow them to transform certain filter bands or shift subsignals temporally in the hidden layers, as much as the kernel width. However, the result of this scheme was not positive (V1.2.7). We hypothesize this is due to the possible introduction of noisy transformations when shuttling the data and to the fact that phase transformations are not

well indicated for the task of denoising the speech. Overall, we suggest that convolutional skip connections could be useful in future tasks if we have strong misalignments between input and output signals, so that these connections can operate with signal shifts from encoder to decoder.

In addition to determining that learnable scalar skip connections give us the best result so far, we experiment with two different initialization weights on them: 0 and 1. We reach the conclusion that 1 (V1.2.8) is better than 0 (V.1.2.8.1). This result is an intuitive one, given the issues in gradient and data flow provoked having no connections at the beginning. We also try the summation merge described in section 5.1 (V1.2.8.2), which is ultimately worse than both concatenation alternatives with different initialization schemes. Still, the use of this scheme might be of interest for running the system in environments where memory and/or computing power is restricted, as having fewer feature maps (and thus parameters) can be important. Finally, we also check what happens if we pick the feature activations after the PReLU (V.1.2.8.3) instead of prior to it (V1.2.8) in the encoder to shuttle them up. These are injected into the input of each mirrored decoder layer as before. Performance degrades in this case, thus indicating the superiority of the linear projection before the activation for the skip connection.

5.3.1.5 Latent z

We find it beneficial to have a latent vector $\mathbf{z}$ at the core of the G structure, which yields better enhancement performance due to a possible regularization effect in the denoising task. In the GAN context, $\mathbf{z}$ serves as a stochastic element to make novel samples at each inference, thus providing generative characteristic to G. Our first intention to place it in G is because the enhanced signal is a regeneration of the noisy one. However, the preliminary hearing results of $\mathbf{z}$ suggest that it only minimally affects any hearable structure in the output (with the same input noisy signal sounding similar to different $\mathbf{z}$), which was already noted in the research of conditional generators when incorporating GANs (Isola et al., 2017). Nonetheless, when we remove $\mathbf{z}$, we systematically find a worse performance in all objective metrics (V1.2.8.4, table 5.1). We hypothesize that this effect is related to some form of overfitting when we lack this noise. Further training checkpoints yielded similar performances, including the model checkpoint of parameters at the epoch with minimum validation error in COVL and SSNR.

Although $\mathbf{z}$ has a reduced relevance as a generative component in the speech denoising application, it can become a key piece in a speech restoration task. In section 5.5, we apply SEGAN+ on a more difficult signal regeneration task, specifically, to construct pitch contours from damaged, silent speech. In this new task, we empirically observed the generation of different but plausible pitch contours with the same input. An example of this approach is shown in Fig 5.8.

5.3.1.6 Biases

After experimenting with the introduction and absence of biases in the full convolutional structures of G, we choose to maintain them as they give a higher peak performance in perceptual objective results. Nevertheless, it is worth discussing the importance of not having biases and considering this feature for future implementations. The intuition behind the absence of bias is that if we have pure silence in the input, we should have pure silence in the output. This condition arises only if we have a multiplicative interaction within the network with a zeroed-out signal,

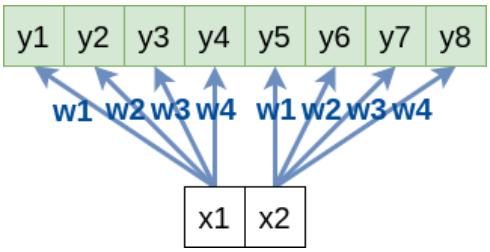

FIGURE 5.4: One-dimensional transposed convolution ('deconvolution') with both kernel width and stride equal to 4. The same kernel with weights $\mathbf{w}_i \in \mathbb{R}$ with $i \in [1, 4]$ (in a one-channel input case) does not overlap itself with neighboring samples.

which is guaranteed if we do not have bias terms in our convolutions. This variation (V1.2.8.5) shows a slightly better SSNR than V1.2.8, but COVL shows a degradation (table 5.1).

5.3.1.7 Transposed Convolutions

We also note the importance of having large kernel widths even in the decoder of the generator. We tried to reduce them to avoid overlapping in the transposed convolution operations, owing to high-frequency artifacts that appear in the output waveform. These could be related to the checkerboard artifacts appearing in image generation transposed convolutions (Odena et al., 2016), something already observed in other GAN-based speech generation systems (Donahue et al., 2019). A plausible mechanism to remove them is working with non-overlapping interpolated segments.

We thus try to reduce the kernel width in the decoder layers in an attempt to make non-overlapping interpolations. Fig. 5.4 shows a schematic of the transposed convolution concept with a non-overlapping form. If we have a kernel width larger than the stride, information between inputs would be mixed in the output of the layer (in the example figure samples, y_i with $i > 4$ would also depend on x_1). Intuitively, for a learnable interpolation in the decoder, a non-overlapping upsampling could suffice, provided that the encoder has sufficient capacity to extract a representation with a large receptive field over the input signal. Nevertheless, we find empirical evidence of a worse performance, particularly in terms of SSNR (compare V1.2.8 with V1.2.8.6–7).

5.3.2 Comparative Results

We now report the results of the aforementioned second block of experiments, comparing SEGAN (V1) and SEGAN+ (V1.2.8) with the baselines (section 5.2.2). We first report objective performance, assessed on a held-out split (section 5.2.1). Next, we report the results of a subjective preference test, based on the mean opinion scores (MOSs) of 42 subjects.

5.3.2.1 Objective Performance

Table 5.2 shows the comparison between SEGAN, SEGAN+ and the considered baselines. First, we observe that all deep learning baselines are over the classic baselines, specially for CBAK and COVL. Nonetheless, the SSNR of the LogMMSE is much better than the one of the spectral deep learning baselines: the SSNR of the

TABLE 5.2: Objective evaluation results with the considered baselines. The L_1/L_2 prefix of DNNs and LSTMs describes the regression loss used, and the C value describes the amount of context frames. For all metrics, higher is better.

Model	CSIG	CBAK	COVL	PESQ	SSNR	STOI
Noisy	3.28	2.28	2.56	1.92	0.03	0.74
Wiener	2.91	2.43	2.43	2.13	3.32	0.73
LogMMSE	3.16	2.67	2.64	2.27	5.00	0.72
L_2-DNN-C1	3.82	2.70	3.02	2.26	2.98	0.70
L_2-DNN-C7	**3.98**	2.83	**3.22**	**2.47**	3.22	0.73
L_1-DNN-C7	3.95	2.81	3.17	2.42	3.38	0.72
L_2-BLSTM	3.75	2.65	2.96	2.21	2.59	0.71
L_1-BLSTM	3.82	2.69	3.01	2.24	2.84	0.70
SEGAN	3.52	2.69	2.77	2.10	5.15	0.73
SEAE+	3.66	2.84	3.00	2.42	5.00	0.73
SEGAN+	3.66	**2.97**	3.00	2.37	**7.05**	**0.75**

LPS-DNN and LPS-BLSTM systems is actually at the level of the Wiener filter or below. The LPS-BLSTM approaches achieve better perceptual scores, but do not reach the level of the LPS-DNN systems in this setup. In terms of PESQ and MOS-like metrics, LPS-DNN and LPS-BLSTM systems generally perform better than other systems. For LPS-DNN, increasing the context C actually helps the DNN, as expected. We also observe that there is a slight difference between using L_2 and L_1 losses in spectral deep learning baselines, but both behave comparably.

It is notorious that the speech enhancement auto-encoder (SEAE+) is objectively comparable to SEGAN+ across perceptual metrics. Nonetheless, CBAK and SSNR show some benefit on using the adversarial component to reduce the intrusiveness of background noise. This result is reasonable for a purely denoising task, where it suffices to remove noisy components as in a regression problem, as stated by Donahue et al. (2018). However, it does not suffice for other applications that require better generative characteristics as in reconstruction (see section 5.5).

In terms of STOI, we observe comparable values between all approaches, with SEGAN+ presenting the best average, suggesting a better resulting intelligibility over all presented models. We also observe that SEGAN+ is superior to all other systems in terms of SSNR, which is related to removing more noise, although it has slightly worse PESQ and MOS-like metrics than the DNN-C7 baselines. We suspect that this result is related to the generative capability of the system: the network regenerates a signal that sounds plausible, but such signal still differs from the original one used for evaluation. Another possible source of trouble is high-frequency artifacts that we identify listening to some samples during model development. Such artifacts can introduce accentuated distortions that lower model scores.

5.3.2.2 Subjective Performance

Objective evaluations such as the ones in the previous section are useful indicators, comparing spectral distortions and noise against clean speech levels. However, such evaluations are not completely fair when we face the generation of new data that can

TABLE 5.3: Subjective evaluation results comparing the considered systems. BCK stands for background noise removal and SPE for speech distortion introduced by the system (see section 5.3.2.2). For both values, higher is better. Each cell shows the mean of each system (standard deviation in parentheses).

Model	BCK	SPE
Noisy	2.84 (1.10)	**4.59 (0.81)**
Wiener	3.19 (1.12)	4.47 (0.87)
LogMMSE	3.43 (1.08)	4.44 (0.93)
L_1-DNN-C7	4.26 (1.08)	4.12 (1.22)
SEGAN	4.24 (1.01)	4.11 (1.21)
SEGAN+	**4.27 (1.09)**	4.21 (1.12)

include audible artifacts noticeable to humans but not accounted for by the metric. In addition, if the regenerated signal differs from the ground truth in terms of amplitude, phase, or other properties that make it intelligible and natural but not an exact fit, objective scores identifying exact matches between low-level properties can lead to misleading results. For these reasons, a subjective test was conducted to assess, with an averaged opinion among many people, how the system performs with regard to the regeneration of plausible speech that resembles a clean reality. For this test, we select the subset of best objectively performant systems per category to compare against SEGAN+. We take the LPS-DNN and LogMMSE as best representatives of the deep learning (spectral) and classic groups, and also maintain the baseline SEGAN and Wiener used in Pascual et al. (2017) as incremental references.

The test was taken by 42 subjects. Each subject was presented with 8 utterances, with each utterance being enhanced by 6 systems. Thus, each user had to rate $8 \times 6 = 48$ audio samples. For each audio, we asked participants to give a MOS rate regarding (a) how intrusive was the background noise (BCK: 5–Not noticeable, 4–Slightly noticeable, 3–Noticeable but not intrusive, 2–Somewhat intrusive, and 1–Very intrusive) and (b) how much speech was distorted (SPE: 5–Not distorted, 4–Slightly distorted, 3–Somewhat distorted, 2–Fairly distorted, and 1–Very distorted). Table 5.3 shows the results for these metrics.

We first focus on the background noise being removed (BCK). We can confirm the incremental gap from Noisy to Wiener and from Wiener to LogMMSE. After LogMMSE, we have the three deep learning systems falling within a comparable range of values, with SEGAN+ achieving a marginally better BCK. In terms of the amount of speech distortion (SPE), we observe a detrimental gap in performance from Noisy to Wiener and LogMMSE, and then the three deep learning systems. Overall, we can understand this result as a trade-off between how much noise they remove and how much speech they destroy. Notably, SEGAN+ remains better than the other two deep learning options for this score, although its performance lies under the classic baselines, as expected, because it clears more intrusive noise as shown in the BCK metric. A conclusion from this result is that with the current subjective results, SEGAN+ seems more selective destroying intrusive signals and that improvements on SEGAN+ make it perform better than the original SEGAN both objectively and subjectively.

5.4 Language and Noise Transfer in SEGAN

Speech enhancement deep learning systems usually require large amounts of training data to operate in broad conditions or real applications. This makes the adaptability of those systems into new, low resource environments an important topic. In this section, we want to quantify how SEGANs perform on languages and noises for which they have not been trained, and assess the amount of new data required to transfer what they have learnt to the new setting. Some of the main research questions we aim to answer in this section are: Does the system perform well for a language it has not been trained for? And for noises? If not, which is the amount of data necessary to adapt the system to the new language/noise? Is it worth to retrain from scratch or is it better to reuse a pre-trained GAN? In both cases, how critical is the selection of training data for adapting the system to the new task? To answer these transfer learning questions in the context of speech enhancement and GANs, we resort to the first SEGAN architecture presented in section 5.3, although these results are exportable to the enhanced SEGAN+ version.

We investigate transfer learning from English to Catalan and from English to Korean[1], as well as the requirements for noise generalization with two experiments for each target language. Experiment 1 trains SEGAN over different speech durations for two baselines: one pre-trained with English (preeng), and the other one based on random parameters (scratch). With this experiment we want to show the amount of speech data required to saturate performance in the transfer learning process. Experiment 2 trains SEGAN over a number of different noise types for both English and random initializations in order to see how generalization to unseen test noises varies with training noise types. Both experiments 1 and 2 consider unseen speakers, sentences, and noise types in the test sets (further details on experiments below).

The English data set consists of a total of 30 speakers (15 male, 15 female), with each speaker recording 400 sentences from the Voice Bank Corpus (Veaux et al., 2013). To train the English baseline, 28 speakers (14 male, 14 female) were chosen, mixed with 40 noise conditions (10 noise types, 4 SNR: 15, 10, 5, and 0 dB). Therefore, around 10 sentences have each noise condition (Valentini-Botinhao et al., 2016). The test set contains 2 speakers contaminated by 20 different conditions from 5 types of noise with 4 SNR each (17.5, 12.5, 7.5, and 2.5 dB).

The Korean data set consists of around 20 minutes of speech for 12 speakers (6 male, 6 female; Chungnam National University students). Recordings were conducted in a quiet room with the speaker and recorder together, with sentences chosen from the Korean web portal NAVER Open Podium and NAVER Encyclopedia. The Catalan data set consists of recordings from 12 speakers (6 male and 6 female) as part of the FestCat project (Bonafonte et al., 2008). Each speaker recorded at least 1h of short paragraphs, which were selected from a set of novels to achieve phonetic and prosodic coverage. The recordings took place in a recording studio. We selected 20 minutes/speaker to balance with Korean data for experiments.

To obtain the training data set for Experiment 1, for each language, we added noise in the same way as the English baseline. The training data set for Experiment 2 was made by adding different amounts of noise types (gradually from 1 to 10). The test data set, for each language, considered 20 noise conditions of 5 different noise types and 4 SNRs from the noise conditions of the training data set for 2 speakers.

Experiment 1: Based on English and random initializations, we train SEGAN over a range of speech durations: 24 s, 1, 2, 4, 10, 20, 50, 100, and 200 minutes. We

[1]The Korean experimentation was a product of a collaboration with Prof. K. Ahn and M. Park from Chungnam National University in Korea.

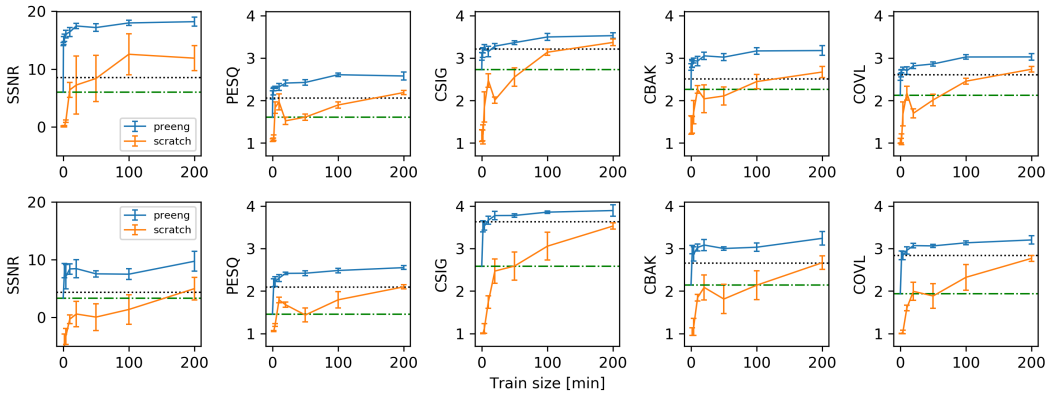

FIGURE 5.5: Objective metrics for Catalan (top row) and Korean (bottom row). Blue line (preeng): Pre-trained with English. Orange line (scratch): trained from scratch. Green dashed line: SEGAN level without fine tunning. Black dash-dotted line: Noisy level.

train the models several times with different utterance selection as follows: ten times for the 24 s, 1, and 2 minute durations, and five times for the 4, 10, 20, 50, 100, and 200 minutes durations. In this way, we can observe the performance differences between the two initialization schemes, with varying speech durations in order to explain transfer learning and data requirements.

Experiment 2: Verification of performance with unseen noises is conducted by training over different noise types. After conducting Experiment 1, we found that 20 minutes duration of training data is enough to saturate performance, so we use this amount. We perform five trainings with each randomly selected noise, increasing one by one the amount of noises until they are ten. Consequently, training is performed 50 times in total. At this time, as in Experiment 1 different utterances are selected for every training, until the training data duration reaches 20 minutes.This allows us to observe the performance differences between the number of training noise types, to include with transfer learning behavior.

The English model is trained for 86 epochs with RMSprop (Tieleman and Hinton, 2012) optimizer, with same hyper-parameters as in the first SEGAN design. All Catalan and Korean models are trained with batches of 100 samples for 30 epochs. The amount of epochs was reduced to a third of the original setting for faster experimentation given the large amount of models. Architecture details remain as in Pascual et al. (2017). To evaluate the quality of the enhanced speech, we compute the same objective measures presented in section 5.2.3.

5.4.1 Language and Noise Transfer Results

The results of the previous metrics as a function of training data time for both Catalan and Korean are shown in Fig. 5.5. Interestingly, despite the important differences between Catalan and Korean, the results for the two languages show very similar trends. We now elaborate on them.

First of all, we note that the pre-trained English system alone does not perform well with the new languages. We can see it with the green dash-dotted lines, corresponding to the non-adapted SEGAN performance, which is always below the dotted black line, corresponding to the noisy test data without processing. Next, and more importantly, we observe that only a few minutes of new training data are needed to drastically improve performance. In fact, even the minimal training time considered in our methodology (24 s) significantly improves test performance.

To put this result into context, we can compare with the results in Xu et al. (2014), where speech enhancement of a resource-limited language was achieved through cross-language transfer learning of 1 min, so we observe a similar trend in our case.

Apart from the aforementioned performance increase, we also see that all the evaluation metrics present a knee at about 10 min, a threshold where more training data shows diminishing returns, with comparable results to those of SEGAN in English from section 5.3.2. This is also amazingly small compared to the training time for English, which was 9.4 h and for 86 epochs, thus three times more epochs than the adaptation experiments. These results also indicate that transfer learning can overcome a lack of training epochs with random initialization.

When deploying a system into the real world it is important to consider the final conditions in which the system will work. In speech enhancement, one may be worried about the final noise conditions being quite different from the training ones, and whether the different types of noise in the training data are sufficient to generalize to such unseen noise. To quantitatively assess this situation we conduct the aforementioned Experiment 2, whose results are depicted in Fig. 5.6. This is the result of training a model 5 times per amount of noises, sampling the noises randomly from the pool of possibilities at every realization of every tick. Once the 5 models are trained, the mean and variance (vertical bar) are shown. Strikingly, the objective test metrics do not present a dependence on the number of types of training noise. Whether the training is with English pre-training or from scratch, performance to unseen noises is not affected by the amount of training noises. The only difference between both versions seems to be in the variance of the results, with the scratch version presenting a much larger variance (vertical bars in the orange curves). This implies that, in the case of training from scratch, one should be careful in which types of noise are considered for training. A possible explanation for the invariability on average, however, might be related to the model separating what is speech from what is non-speech. And this decision might be less volatile if the system is trained with more noise types as in the preeng case.

Finally, we turn our attention to test noise types (Fig. 5.7). Here we also observe consistent behaviors between languages and metrics, with office and bus noise types performing best and street noise, living room, and cafe noises (in this order) performing worse. As a further, more anecdotal note, we see that office and bus noise seem to 'cluster' in some metrics (e.g., COVL) in the upper side of the plots, while street noise, living room, and cafe noises also 'cluster' in lower metrics.

5.5 Whispered SEGAN

In this section we begin exploring the enhancement capabilities of SEGAN+ beyond the specific task of denoising (i.e., eliminating additive noises of many kinds). An important set of enhancement applications are those that directly affect the speech, allowing for the recovery of a more natural spoken utterance out of a damaged one. As a first step in this direction, we begin exploring the application of whispered-to-voiced speech conversions. We refer to this conversion as dewhispering or voicing of the speech signal.

Whispered speech can be uttered on purpose, but it is also expressed by people suffering from disease or trauma that manifests as aphonia (e.g., patients after a total laryngectomy). The dewhispering process is typically performed in the spectral domain or with vocoder features, similarly to the denoising case. Once these features are obtained, some statistical models such as Gaussian mixture models (Toda et al.,

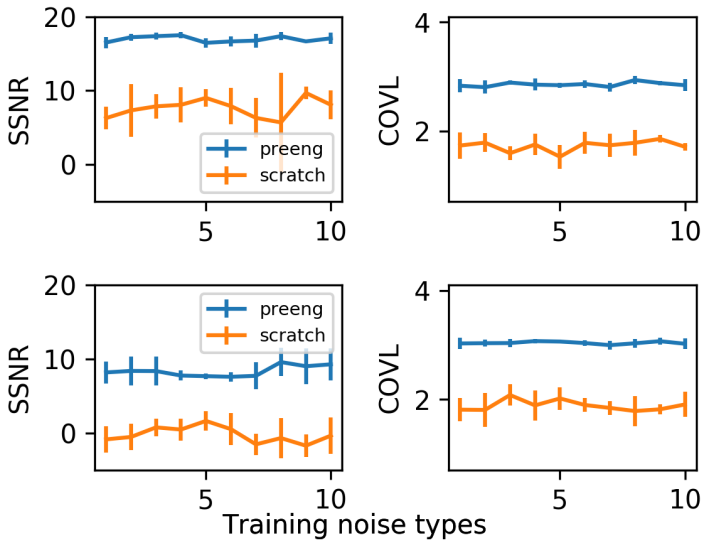

FIGURE 5.6: Noise experiment results for Catalan (top row) and Korean (bottom row). Blue lines (preeng): Pre-trained with English. Orange lines (scratch): trained from scratch. Each line shows the resulting objective evaluation of a certain metric depending on the amount of training noises used to test the curve. The amount of noises ranges from 1 to 10, thus the total 10 available in the training set. Qualitatively similar plots were obtained for the PESQ, CSIG, and CBAK metrics.

2008; Nakamura et al., 2011; Nakamura et al., 2012) or DNNs (Gonzalez et al., 2017a) are applied to reconstruct corrupted/missing components, and then, features are reverted to the time domain.

This work is done in collaboration with J.A. González from Sheffield University. The whispered utterances are obtained with an articulator motion capture device (Fagan et al., 2008) that monitors the movement of the lips and tongue, tracking the magnetic field generated by small attached magnets. Then, an existing synthesis module generates speech from articulatory data by means of an RNN model trained on parallel articulatory-to-speech samples (Gonzalez et al., 2017a; Gonzalez et al., 2017b). The speech produced by this system has a reasonable quality but sounds monotonous and robotic, owing to limitations when estimating the pitch (i.e., the capturing device does not have access to any information about the glottal excitation). Hence, we can use the RNN to generate a reconstructed whispered speech out of articulatory data while discarding the predicted pitch and then apply SEGAN+ as an enhancement over it. SEGAN+ recovers a more natural sounding speech with a whispered input such that it must implicitly generate pitch curves with proper intonations embedded in the waveform. Here, we compare our model against the existing system that employs the RNN-based architecture to regress pitch and performs a vocoder-based synthesis (Gonzalez et al., 2017b).

5.5.1 Experimental Setup

Importantly, because of the data acquisition and synthesis procedures, small temporal misalignments remain between the input and output records (i.e., whisper and natural speech differ in length and are not accurately parallel). Therefore, the original SEGAN version is not immediately effective when receiving the new data, particularly because the L_1 regularization loss is restricted to work when the output is fully aligned with the input. Similarly, we also expect an L_1 auto-encoder to not

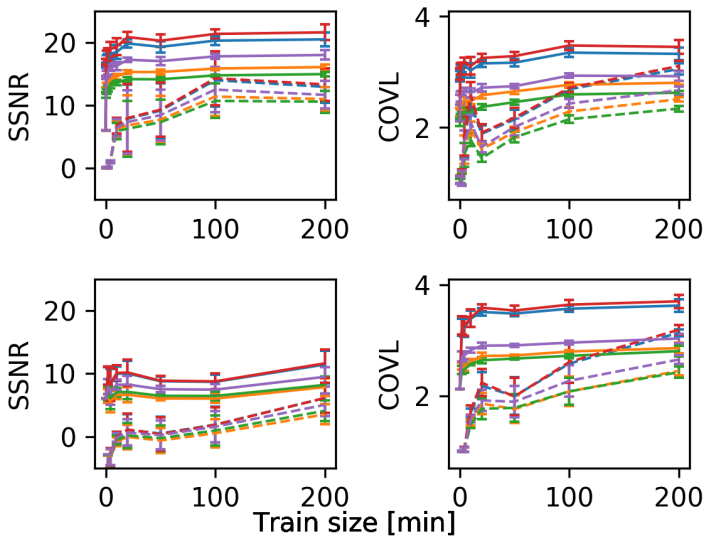

FIGURE 5.7: Performance on different test noise types. From top to bottom (first solid, then dashed lines), noise types correspond to: office (red), bus (blue), street (purple), living room (orange), and cafe (green) noises. Qualitatively similar plots were obtained for the PESQ, CSIG, and CBAK metrics.

be effective in the regeneration of missing components, as stated in section 5.3.2.1. Nonetheless, we consider the SEAE+ architecture in this setup too, as a standalone reconstruction system.

The amount of data we have to carry out this experimentation is 30 min of training utterances plus 3 min of test utterances[2] (it is important to note the large gap in terms of amount of data to train on, from SEGAN/SEGAN+ models, which handle 32 h, relative to this small set of 30 min). We noticed that this data shortage has two main effects: (1) it introduces artifacts at many frequencies, particularly the high ones, and (2) intelligibility is sometimes lost in the reconstruction phase. Hence, we had to make two reformulations to SEGAN+ to adapt it to the current task. We denote this reformulated version as WSEGAN to specify its applicability to dewhispering.

First, time-domain regularization is removed from the loss of G, and we use only the power loss as a regularizer (van den Oord et al., 2017). In this way, we try to mitigate the allocation of energy in non-speech-like frequency bins and thus reduce the aforementioned artifacts. We also add a denoising SEGAN+ processing system on top of WSEGAN to remove erratic artifacts and corrupted speech segments, which acts specifically on silence regions. Second, we introduce a new adversarial loss that enforces content preservation between the input and the output of G. More specifically, a synthetic signal (0 in the LSGAN binary coding, section 3.2.2.1) is triggered whenever we have the current clean reference signal $\mathbf{x}$ and another randomly chosen clean signal $\mathbf{x}_r$. Both signals are clean and look natural, and the only difference is the content mismatch; thus, D must learn that mismatched information is not realistic.

[2]The corpus contains a single English male speaker that recorded a random subset of the CMU Arctic corpus (Kominek and Black, 2004).

TABLE 5.4: Mel cepstral distortion results for the three considered systems: RNN baseline, SEAE+ (L1 auto-encoder), and WSEGAN.

	RNN	SEAE+	WSEGAN
MCD [dB]	8.01	17.19	12.81

With these two changes, the WSEGAN loss becomes

$$\begin{aligned}\min_D V(D) &= \frac{1}{3}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}[(D(\mathbf{x},\tilde{\mathbf{x}})-1)^2]+ \\ &+ \frac{1}{3}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})^2] \\ &+ \frac{1}{3}\mathbb{E}_{\mathbf{x},\mathbf{x}_r\sim p_{\text{data}}(\mathbf{x})}[D(\mathbf{x},\mathbf{x}_r)^2] \\ \min_G V(G) &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[(D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-1)^2]+ \\ &+ \alpha\,\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}},\mathbf{x}\sim p_{\text{data}}}[|\Phi(G(\mathbf{z},\tilde{\mathbf{x}}))-\Phi(\mathbf{x})|],\end{aligned} \tag{5.4}$$

where $\tilde{\mathbf{x}} \in \mathbb{R}^T$ is the whispered utterance; $\mathbf{x} \in \mathbb{R}^T$ is the natural speech; $\mathbf{x}_r \in \mathbb{R}^T$ is a randomly chosen natural chunk within the batch; $G(\mathbf{z},\tilde{\mathbf{x}}) \in \mathbb{R}^T$ is the enhanced speech; $D(\mathbf{x},\tilde{\mathbf{x}})$, $D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})$, and $D(\mathbf{x},\mathbf{x}_r)$ are the discriminator decisions for each input pair; and $\Phi(\mathbf{x})$ corresponds to the short-time Fourier transform magnitude in dBs (20 ms windows, 10 ms stride, and 2048 bins), with $\alpha = 10^{-3}$ corresponding to the weighting of this term.

5.5.2 Results

First of all, we perform an objective evaluation with mel cepstral distortion (MCD). MCD is an indicator of correct uttered content generation and speaker identity match in speech synthesis. We use the same formulation as in the TTS work from section 4.4.4. Table 5.4 shows the results for the baseline RNN, the SEAE+ and WSEGAN. Firstly, we can see that SEAE+ has the highest distortion rate, indicating its lack of reconstruction capacity from the whispered signal towards the clean one. Actually, qualitative listenings allow us to appreciate how it is not able to reconstruct voiced segments, and the best it does is a low pass reconstruction of the input whispered signal itself. The application of the adversarial component is thus more critical in this setup as our intuition from section 5.3.2.1 suggested. The qualitative listenings for WSEGAN reveal pitch reconstructions that match natural intonations and the expected modelled male identity (no low-pass effect is observed in this case). Regarding the RNN, it obtains the best score objectively, thus indicating possibly the best match to the clean signal. It is expectable to obtain such score as the model was directly optimized to minimize its quadratic error towards the clean spectral components. Nevertheless, low distortion scores are not always an indicator of a natural sounding voice in speech synthesis. In fact, a model with increased variance in its acoustic predictions (as in the case of WSEGAN), which in turn increments speech naturalness, can be an objectively inferior model (Henter et al., 2018). Hence, a subjective evaluation is normally the best procedure to assess the generated speech naturalness.

For the case of generated intonations of the two successful models, Fig. 5.8 lets us appreciate examples of generated pitch contours. We can first observe an increased

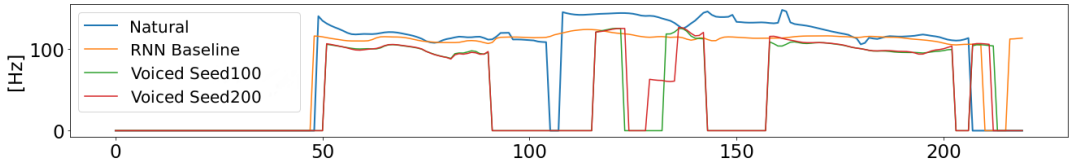

FIGURE 5.8: Natural pitch contour in blue and three example reconstructions. Orange is the RNN baseline contour, with a relatively flat behavior. Green and red are two different voiced versions from WSEGAN, produced with different latent codes $\mathbf{z}_1$ and $\mathbf{z}_2$ (i.e., different random seeds).

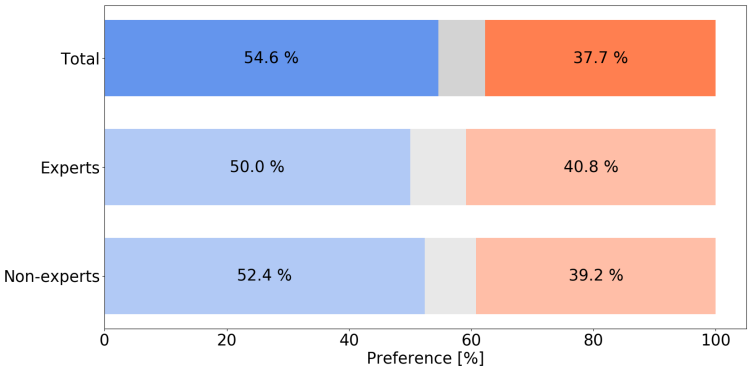

FIGURE 5.9: Subjective test preference results on naturalness rating between RNN regression baseline and WSEGAN. Blue denotes WSEGAN preference, orange denotes RNN preference and gray denotes that both are equally preferred.

variance in the pitch contours of the signal for WSEGAN as opposed to the RNN. The figure also shows different trajectories that match plausible intonation contours depending on a randomly selected latent description $\mathbf{z}_i$, which is enabled by the generative capacity of the model. We can also appreciate that the regenerated signal has different voiced/unvoiced regions (unvoiced regions are denoted by a 0 Hz signal). We hypothesize that these mismatches with the ground-truth signal may be corrected with the addition of more data, as the network could better estimate the right placement of pitch contours within the spoken contents of the damaged signal.

A subjective test was carried out to assess the improvement of WSEGAN with respect to the pitch regression RNN baseline system (online samples are available online[3] so that the reader can evaluate the differences qualitatively). A set of 25 subjects listened to and rated 10 randomly selected test utterances from a pool of 44, choosing whether they preferred the naturalness of one system, the other one, or both of them (the order of the two systems per utterance was shuffled). The results of this test are shown in Fig. 5.9, where WSEGAN (green) is preferred in 54.6% of the utterances, against the 37.7% of preference for the RNN system (red). Additionally, we observe no clear difference of preference between expert and nonexpert listeners (12 participants declared having expertise with speech signals and audio processing techniques). Participants noted that WSEGAN could sound more natural, implicitly producing proper intonations matching the sentences, but loses intelligibility in some utterances, potentially owing to the lack of data for such a large model. Interestingly, a native English listener even unveiled the geographic accent of the English speaker accent after WSEGAN recovery.

[3] `http://veu.talp.cat/whispersegan`

5.6 Generalized SEGAN

After demonstrating the effectiveness of SEGAN+ to recover the voicing speech components, we take a step further and make it generalize to more distortions. Hence in this section we extend the generalization from dewhispering and consider the problem of reconstructing speech that has been degraded by a number of (different) signal manipulations, each of them being potentially highly perceptually harmful. We propose to mix up such manipulations together with several speaker identities at training time, with the objective of being able to train generative algorithms to recover multiple (and diverse) speech components simultaneously. As a first approximation towards this generalized speech enhancement task, we consider the following signal manipulations:

- **Whispered speech:** This distortion is similar to the one introduced in the WSEGAN section 5.5. However, it differs in the fact that the work in section 5.5 used a magnetic sensor to acoustic contours conversion for alaryngeal speech conversion, where voicing is inherently missing as it is not sensed from the real speaker. Here, on the other hand, instead of using magnetic sensors, we synthesize whispering speech by encoding the clean speech with a vocoder, removing the log-F0 (i.e. making all frames unvoiced), and recovering the signal into a version that whispers, hence artificially removing voicing. The vocoder used is Ahocoder with the default parameters (Erro et al., 2011).

- **Bandwidth reduction:** We downsample the audio signals with different severity factors, ranging from $\times 2$ to $\times 8$, thus reducing the bandwidth. Then the generalized enhancement model must reconstruct entire frequency bands, according to the clean signals seen in training.

- **Chunk removal:** For the parts of the waveform that contain speech (detected automatically with a voice activity detector), a random number of chunks are subtracted by inserting silences. Thus, we cancel the signal at random temporal sections that contain speech. The length of a silence is sampled from one of two distributions $\mathcal{N}(0.05, 0.025)$, and $\mathcal{N}(0.1, 0.05)$ (numerical values in seconds).

- **Clipping:** The waveform is clipped globally by different severity factors relative to the maximum absolute peak of the whole utterance (e.g., 30%). The regenerated signal thus has to re-condition the signal inside a proper, non-distorted range of amplitudes.

To better deal with the added difficulty that these signal manipulations can introduce to the speech signal, we propose two modifications to the existing SEGAN+ pipeline: the addition of two acoustic losses and the introduction of a two-stage training schedule for applying such losses.

5.6.1 Acoustic Losses

In general, D can be understood as a learnable loss function, where the realistic features we want to generate are implicit in the back-propagation, and G gets better because of D's gradient flows (Goodfellow et al., 2014). This builds a need for D to be a competitive feature extractor, so that the better the features extracted by D, the better G can capture reality. Athough GANs were designed as an unsupervised learning strategy, they are proven to be more stable and effective when coupled with

labels that are either injected as input conditionings (Radford et al., 2016; Brock et al., 2019) or that help classify some traits about the data being generated (Odena et al., 2017; Lučić et al., 2019). Also the successful speech synthesis model parallel WaveNet (van den Oord et al., 2017) proved that a multi-task aggregation on top of the speech generative model is beneficial to improve speaker identity, prosodic, and content traits. Hence, the use of auxiliary classifying labels in D is generally helpful. Additionally, multi-task setups can boost generative modeling performance, where multiple factors of the signal are predicted at the output of the generative model to enforce modeling better perceptual qualities of the signal (van den Oord et al., 2017). This is applicable to the SEGAN setup where the discriminator only learns features that lead to a `real` or `fake` decision, but is not concerned about specific factors of the signal that are important to remark the reconstructed speaker identity, prosody, or contents. Note that we can also have some additional regularization loss in the output of G too, aggregated to the adversarial loss coming from D. For instance the first version of SEGAN used an L_1 regularization term that made G output a zero-centered signal (Pascual et al., 2017), and the posterior WSEGAN implementation substituted it by a less restrictive power loss (van den Oord et al., 2017; Ping et al., 2019), enforcing proper energy allocation in the frequency bands.

Following the aforementioned multi-task improvement criteria, we propose the use of an additional acoustic loss that opens a new branch in the end of D. This makes D necessarily aware of additional acoustic components that can inform G about important mistakes when mismatching identities, intonations, or contents in the reconstruction without necessarily injecting any additional conditioning in the input of G besides the distorted signal $\tilde{\mathbf{x}}$. This is advantageous in terms of avoiding restricting the generalization of G, as we do not inject any code limited by a set of possible classes to specify identities, distortion types, or any other kind of information. This new D output branch grows from a certain level in the convolutional structure, where the time decimation factor is 256, thus each time-step corresponds to 16 ms stride over the waveform (emulating a short-time Fourier transform sliding window). At each output frame, it predicts a concatenation of LPS bins, MFCC, and prosodic features. We design it using this collection of features as we hypothesize they properly convey different levels of acoustic cues, ranging from power allocation expressing identity, to content and intonation clues. This branch is just in addition to the binary output neuron that tells `real` and `fake`, and is only active when this binary activation is `real` with input $\mathbf{x}$ or $\tilde{\mathbf{x}}$ (when G tries to fake). The proposed acoustic loss in D and power regularization in G redefines the loss function as

$$
\begin{aligned}
\min_G V(G) = & \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[(D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-c)^2] \\
& + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}[(\delta(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-\Theta(\mathbf{x}))^2] \\
& + \alpha\,\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}[\Phi(G(\mathbf{z},\tilde{\mathbf{x}}))-\Phi(\mathbf{x})], \\
\max_D V(D) = & \frac{1}{4}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}[(D(\mathbf{x},\tilde{\mathbf{x}})-b)^2] \\
& + \frac{1}{4}\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}\sim p_{\text{data}}(\mathbf{x},\tilde{\mathbf{x}})}[(\delta(\mathbf{x},\tilde{\mathbf{x}})-\Theta(\mathbf{x}))^2] \\
& + \frac{1}{4}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z}),\tilde{\mathbf{x}}\sim p_{\text{data}}(\tilde{\mathbf{x}})}[(D(G(\mathbf{z},\tilde{\mathbf{x}}),\tilde{\mathbf{x}})-a)^2] \\
& + \frac{1}{4}\mathbb{E}_{\mathbf{x},\mathbf{x}^r\sim p_{\text{data}}(\mathbf{x})}[(D(\mathbf{x},\mathbf{x}^r)-a)^2],
\end{aligned}
\tag{5.5}
$$

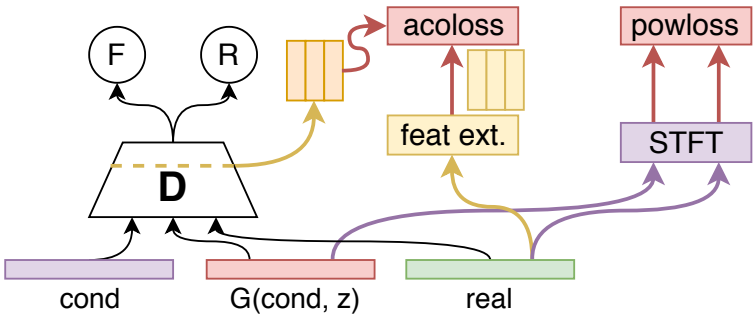

FIGURE 5.10: Setup of new multi-task framework with acoustic loss, power loss, and adversarial outcomes (F and R).

where $\mathbf{x}$, $\tilde{\mathbf{x}}$, $\mathbf{x}^r$ and $\mathbf{z}$ are the same as in equation 5.4, $a = -1$, $b = 1$, and $c = 0$ contrarily to the previous sections that used binary coding, $\Phi(\mathbf{x})$ corresponds to the short-time Fourier transform magnitude in dBs (20 ms windows, 10 ms stride, and 2048 bias), and $\alpha = 10^{-3}$ is a weighting term Pascual et al., 2018b. The LSGAN constants a, b, and c are changed in this section because after some preliminar listenings comparing both coding schemes this seemed to raise better results with the same amount of iterations for this task. The δ function is the discriminator output at some intermediate layer of our choice, and $\Theta(\mathbf{x})$ is the feature extractor of LPS, MFCC and prosodic features. Hereafter, we will refer to the joint power loss regularization and acoustic loss as acoustic losses.

5.6.2 Adversarial Pre-Training

Whereas SEGAN with only an adversarial loss is stable and learns in a steady equilibrium (equation 3.24), the addition of the acoustic losses induces a particular unbalance effect during training. The addition of these terms makes D learn quicker and converge faster in both losses, conversely making G converge slower. Importantly, both G and D should maintain an equilibrium learning from each other, so making one of them quickly better discourages the other to perform properly Goodfellow et al., 2014. We hypothesize that scheduling the learning of the discriminator as a two-stage process makes the addition of these acoustic losses more effective for it first gets high-level representations classifying, and then focuses on specific speech properties by doing regressions. Hence we first do an adversarial warm up in D with equation 3.24 formulation, and then add the acoustic losses.

5.6.3 Dataset

We again employ the VCTK Corpus for this experimentation (Veaux et al., 2017). We select 80 speakers for training and 14 for test. We trim large silence regions to 100 ms with the help of a voice activity detector. After this, we get roughly 20 h of training speech and 3 h of test speech. We incorporate the aforementioned distortions with an online process jointly working with training. When we construct a training minibatch, we (1) load the clean utterance, (2) get a random ≈1 s chunk (16384 samples) inside the utterance, and (3) apply a series of transforms that get activated independently under a probability $p = 0.4$. This way one or two distortions coincide often, and zero (thus purely autoencoder) or more than two coincide less. Table 5.5 shows this activation distribution with up to the 4 mentioned combinations. In addition to being active or not, each transform has a certain severity level or factor, as

TABLE 5.5: Frequency of number of training active transformations using $p = 0.4$.

Num. of transforms	0	1	2	3	4
Relative frequency	0.14	0.34	0.33	0.15	0.04

mentioned before (section 5.6). The only transform with just one level of severity is the whispering transform. Hence every time a transformation is activated in the pipeline, one of the possible factors is randomly selected. These possibilities are: clip factors of 30%, 40% and 50%, signal resampling factors of 2, 4, and 8 and, in the case of chunk removal, we allow the system to zero out up to 5 chunks from speech regions. Then, inside each region, the length of the chunk is sampled from one of the two mentioned Gaussian distributions.

For the test split, we have two different setups, one for an objective evaluation and another one for a subjective one. For the objective test set, we proceed as with the training set. For the subjective test, we generate a special split with two subsets in order to account for two different effects present in the system outcomes. The first subset is designed to focus on generated speaker identity (*SpkID* pool). To do so, we just use the most severe version of bandwidth reduction ($\times 8$) and the whispering manipulations, so the ones that affect the most this feature. Then, for each test speaker we degrade four utterances, two with bandwidth reduction and two with whispering. The second subset focuses on the generated speech naturalness (*Nat* pool). To evaluate this aspect, we pick 4 test speakers (two male and two female) and apply clipping, resampling, and whispering to 6 utterances in total (two utterances per distortion). Both clipping and resampling have severity factors 30% and $\times 8$, respectively.

5.6.4 Experiments

We experiment with three models applied over the aforementioned data. Firstly, we use SEGAN+ in a plain adversarial setup with the LSGAN loss as in equation 3.24, where the only signal comes from `real` or `fake` decisions. This is the baseline, which is trained for 400 epochs with two-timescale update rule (TTUR) learning rates (Heusel et al., 2017) $\eta_D = 4 \cdot 10^{-4}$ and $\eta_G = 1 \cdot 10^{-4}$. Secondly, we apply the acoustic losses presented in section 5.6.1. This system trains also for 400 epochs, but the learning rates are balanced equal $\eta_D = \eta_G = 5 \cdot 10^{-5}$, because the discriminator already has an advantage with the extra signals and TTUR involved a noisier result on G. We name this approach SEGAN-Aco. Finally, we include our improved proposal, for which we pre-train the system over 100 epochs the same way we do with the baseline, and then we activate the acoustic losses for the remaining 300 epochs, also lowering the learning rates to $\eta_D = \eta_G = 5 \cdot 10^{-5}$. We name this approach SEGAN-PTAco.

In terms of architecture for this application, the discriminator is the only diverging component with respect to WSEGAN. The change happens in the classification part of the network, i.e. after the convolutional stack. We substitute the previous average pooling with a single output neuron for a MLP with 16384 inputs (1024×16 feature map, unrolled), 256 hidden PReLU units, and the single output for `real` and `fake` predictions.

Secondly, we find the acoustic branch for SEGAN-Aco and SEGAN-PTAco models at the fourth convolutional layer, which outputs 512×64 feature maps for an input of approximately 1 s at 16 kHz sampling rate. These corresponds to 64 frames,

each one injected into the acoustic prediction MLP of 128 hidden PReLU units and 277 linear outputs. These outputs predict 257 log-power spectral bins, 16 MFCCs, 1 log-F0 value of the frame, 1 voiced/unvoiced frame flag, 1 frame energy coefficient, and 1 frame zero crossing rate. Mini-batches of 150 samples are used for all models. Additionally, *D* implements spectral normalization to avoid sudden exploding gradients leading to training collapse (Zhang et al., 2018a; Miyato et al., 2018) and phase shuffle of 5 to reduce high-frequency artifacts in the output of *G* (Donahue et al., 2019).

5.6.5 Evaluation

To assess each model result we perform two evaluations. First, we run a number of objective distortion metrics, often applied in speech synthesis problems: the mel cepstral distortion (MCD; in dB) as previously done with WSEGAN. We also compute the F0 root mean squared error (RMSE; in Hz), and the voiced/unvoiced frame prediction error (UV) Pascual and Bonafonte, 2016a. These errors give us a first clue on how close is each system to the clean original signal in terms of content, identity, and intonation.

Given the importance of perceptual scores, we also conduct a subjective evaluation with 26 listeners. This has two stages, with two tasks to be evaluated by the listeners: speaker identification and naturalness rating. For speaker identification, listeners are asked to determine how close a reconstructed signal is towards the original speaker identity. They are presented with 4 randomly-selected utterances out of the *SpkID* pool (section 5.2.1). For each utterance, the clean reference is shown, as well as the four systems to be rated: (1) the degraded input signal to *G*, (2) the SEGAN+ baseline, (3) SEGAN-Aco, and (4) SEGAN-PTAco. Due to the fact that we impose incremental changes over the SEGAN+ baseline, and all the models should also perform incrementally better than the distorted signals, the rating is as simple as ordering the systems by preference. Additionally, using a MOS score for this test could cluster different systems together with the distorted reference due to preferred artifacts by some listeners, thus obtaining confusing results. This way the comparison among systems is simplified and more discriminative. In this ranking, position 1 is for the system closest to the reference and position 4 is the furthest one. For naturalness rating, 6 utterances taken randomly out of the *Nat* pool (section 5.2.1) are shown to each listener. For each utterance, the four different aforementioned systems are shown and asked to be ranked from most natural (1) to least natural (4). There is no reference shown for this case as there is no similarity trait like a speaker identity, it is only comparing the perceptual quality of the synthesized utterances. In all subjective evaluations, systems are shown in random order at every utterance.

5.6.6 Results

Objective results are shown in table 5.6. We can see how, in all metrics, SEGAN-PTAco is the best system and the least noisy one. A lower MCD can be assumed to correlate with better content preservation, identity reconstruction, and naturalness generation (Pascual, 2016). The lower values obtained for F0-RMSE and UV error also typically denote better intonation schemes matching the input identity with prosodic contents. The distorted signals have the noisiest behaviors, with large variances in all error metrics. Also expectedly, we observed that increasing the number of signal degradations yielded worse objective metrics.

TABLE 5.6: Objective metrics for the different systems (standard deviation into parenthesis). For each metric, lower is better.

Model	MCD [dB]	RMSE [Hz]	UV [%]
Distorted	7.6 (5.6)	52.9 (75.9)	22.3 (25.5)
SEGAN+	8.2 (3.5)	37.2 (43.5)	9.1 (11.4)
SEGAN-Aco	7.4 (2.7)	37.5 (53.5)	20.0 (23.4)
SEGAN-PTAco	**6.5 (2.7)**	**22.7 (28.5)**	**5.8 (6.3)**

TABLE 5.7: Subjective mean ranking score (lower is better).

Model	Speaker ID	Naturalness
Distorted	2.87 (1.07)	3.00 (1.04)
SEGAN	2.85 (1.03)	2.69 (1.04)
SEGAN-Aco	2.88 (0.83)	2.78 (0.92)
SEGAN-PTAco	**1.41 (0.74)**	**1.52 (0.84)**

Subjective results are shown in table 5.7. In general, they also highlight the importance of both the acoustic losses and the two-stage adversarial training schedule. We can see that the baseline and SEGAN-Aco are clustered together with the distorted signals for the Speaker ID test. This implies that, for both bandwidth extension and dewhispering problems, neither systems reconstruct a proper speaker identity from the input signal. A qualitative listening in this case tells us that the baseline imposes an arbitrary identity to the reconstruction, and SEGAN-Aco sounds robotic and muffled. In contrast, SEGAN-PTAco is consistently ranked above the distorted signal, and we can clearly hear a competitive identity consistency with the conditioning. In the naturalness task, the baseline and SEGAN-Aco systems do better than the distorted signals. This was expected, as they involve an enhancement process that removes very noticeable artifacts and recover speech nuances. Nevertheless, SEGAN-PTAco is still the best system by a large margin. Some audio samples are available at `http://veu.talp.cat/gsegan/`.

5.7 Discussion and Conclusion

In this chapter, we proposed a speech enhancement method framed within the GAN methodology using raw audio. We explore some variations of it that make it more efficient and effective. The model is an encoder-decoder fully convolutional structure, which makes it adaptable to deal with sequences of any length. With the introduced variations, we unveil some possible future paths to further improve the architecture, specifically in terms of encoder structure to obtain a better decimation scheme. The current results suggest that our approach performs better than classic baselines such as Wiener or LogMMSE. They also show that the approach is competitive with custom-tuned deep learning models in the log-power spectral domain, trained with a regression on the magnitude, where a major amount of noise is detected and removed. Our approach, on the other hand, requires little preprocessing, working on the raw waveform, and is more flexible to work with other enhancement tasks. We also verify the effectiveness of the adversarial component over the fully convolutional regression system.

We also show that transfer learning is very efficient for inter-language speech enhancement by generative adversarial network. Pre-trained SEGAN with English achieves high performance even for short training time of Catalan and Korean (24 s)

with unseen speakers and noise, allowing us to adapt the system to low resource environments. We also find that the number of noise-type in training is not crucial factor to the performance of the speech enhancement. While training SEGAN is a difficult task, our results imply that one can detour the problem through transfer learning using a pre-trained network.

Finally, we propose the application of speech enhancement GANs in a more generalized signal recovery framework. First, we build a speaker dependent model to solve the dewhispering problem, improving the generated naturalness over that obtained with a baseline RNN system, as proven by objective and subjective results. Then, we scale the generalization to different adverse conditions by introducing four aggressive distortions applied to the speech signals. Next, we introduce a new acoustic regression component in the SEGAN+ discriminator and, in addition, we propose a two-stage adversarial training method to make the new acoustic regression component reach an appropriate performance. Objective and subjective results are provided, showing how both the regression losses and and the two-stage schedule outperform the regression losses alone and the purely adversarial approach.

This chapter is devoted to the construction of an end-to-end speech enhancement model, hence it makes no assumption on the input data or features. The feature extraction and signal regeneration is learned in the optimization loop itself with the end-task. We can observe, however, that inducing prior knowledge improves the model performance, boosting specially its generated quality for adverse conditions as shown in section 5.6. To achieve this, we make use of a self-supervised output branch (see section 2.3) that predicts a mixture of acoustic features which correlate with a perceptual description of the signal. In the next chapter we will mix a raw waveform based speech encoder and the self-supervised framework. With this, we are merging the best of end-to-end processing (no assumptions about the input data and feature pre-computation) with a prior knowledge based design.

Chapter 6

Learning Problem-Agnostic Speech Representations from Multiple Self-supervised Tasks

Despite the recent progress detailed in chapter 2, applying self-supervised learning to speech remains challenging. Speech signals are not only high-dimensional (i.e. long) and variable-length sequences, but also entail a complex hierarchical structure that is difficult to infer without supervision (phonemes, syllables, words, etc.). It is thus hard to find a single self-supervised task that can learn general and meaningful representations able to capture this latent structure.

To address this issue, we propose to jointly tackle multiple self-supervised tasks using an ensemble of simple neural networks that cooperate to discover good intermediate speech representations. Importantly, the self-supervised tasks are designed upon prior knowledge about the structure of speech. Hence, we will induce the discovery of multiple levels of well-known and effective speech features like the MFCCs, the LPS, prosodic features, and contextual relations among speech frames in terms of pseudo-stationarity or sequentiality. The intuition behind this self-supervision in a multi-task scenario is that each task may bring a different view or soft constraint on the learned representation. Even though not all the tasks may help for a final supervised problem of interest (e.g. speaker recognition), there is likely a subset of them that could be useful. Moreover, even when some tasks may not be useful, they may not be harmful either to any other supervised task. An important implication of this proposal is that our approach requires consensus across tasks, hence naturally imposing constraints into the learned representations.

This way, our approach is likely to learn general, robust, and transferable features, and less likely to focus on superficial features of the signal which may be sufficient for the given training data but are insufficient when considering broader types of data. To highlight the latter property, we call our proposed architecture the problem-agnostic speech encoder (PASE). PASE encodes the raw speech waveform into a representation that is fed to multiple regressors and discriminators. The regressors deal with standard features computed from the input waveform, resembling a decomposition of the signal at many levels. On the other hand, the discriminators deal with either positive or negative samples and are trained to separate them by minimizing binary cross-entropy as in Ravanelli and Bengio (2019). Both regressors and discriminators (hereinafter called *workers*) contribute to add prior knowledge into the encoder, which turns out to be crucial to derive meaningful and robust representations.

Importantly, a number of design choices make the encoder efficient and easily

exportable, facilitating its direct usage or adaptation to different problems. We begin testing PASE representations on 3 supervised tasks: speaker recognition, emotion recognition and speech recognition. The latter is tested both in clean and noisy conditions (i.e. distant speech recognition). We also propose the use of the resulting problem-agnostic speech embeddings in a multi-speaker acoustic model for TTS based on the SampleRNN generative model. This way, we feed the acoustic model with speaker acoustically-dependent representations that enrich the waveform generation more than discrete embeddings unrelated to these factors. Finally, we also close the gap between this new learnable representation and the previously proposed end-to-end system for generalized speech enhancement with some preliminary results.

We first describe the proposed PASE framework in section 6.1. The details about the designed encoder are described in section 6.1.1. Then, a detailed description of the designed self-supervised tasks to train the encoder is given in section 6.1.2. What follows is a description of the self-supervised training strategy we perform upon the proposed encoder in section 6.1.3. A brief description in how PASE can be used after self-supervision is given in section 6.1.4. Then, the different corpora to carry both training stages are described in section 6.2, as well as the details about each supervised task to assess the final quality of PASE representations. The results on the aforementioned classification tasks are discussed in section 6.3. Importantly, the use of such classifiers allows us to first evaluate how better or worse PASE becomes with the subtraction of each worker in the self-supervised loop, hence an ablation study. Secondly, we can compare PASE features with other classic ones such as MFCCs or FBANKs, known to be robust and widely used for recognition tasks. Finally, the transferability of PASE features to new acoustic conditions not seen during training, such as noisy ones, can also be studied. Then, once we have assessed the quality of these representations to convey information like the speaker identity, we can use PASE features to condition a speech generation module such as a TTS acoustic model. This application is explored in section 6.4.

6.1 Problem-agnostic Speech Encoder

The PASE architecture, depicted in Fig. 6.1, is built as a fully-convolutional neural network. This is followed by seven MLP workers, which cooperatively solve different self-supervised tasks. We now describe these modules.

6.1.1 Encoder

The first layer of the encoder is based on the recently-proposed SincNet model (Ravanelli and Bengio, 2018b). SincNet performs the convolution of the raw input waveform with a set of parameterized sinc functions that implement rectangular band-pass filters. While CNNs learn all elements of each filter (see section 3.1.2), SincNet learns their low and high cutoff frequencies only, leading to a very compact model (Ravanelli and Bengio, 2018a). Therefore, an interesting property of SincNet is that the number of parameters does not increase with the kernel size. Similarly to Ravanelli and Bengio (2018b) and Ravanelli and Bengio (2018a), we use a large kernel width $W = 251$ to implement $F = 64$ filters with a stride $S = 1$. The subsequent composition is a stack of 7 convolutional blocks (Fig. 6.1). Each block employs a one-dimensional convolution, followed by batch normalization (BN) (Ioffe and Szegedy, 2015), and a PReLU activation (He et al., 2015). For the 7 blocks we use

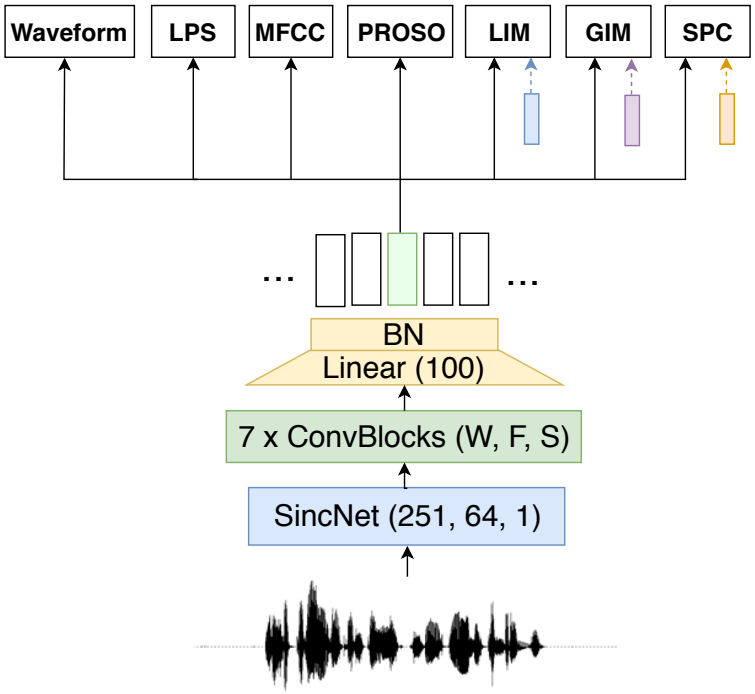

FIGURE 6.1: The PASE architecture, with the considered workers.

kernel widths $W = \{20, 11, 11, 11, 11, 11, 11\}$, $F = \{64, 128, 128, 256, 256, 512, 512\}$ filters, and strides $S = \{10, 2, 1, 2, 1, 2, 2\}$. An additional layer performs a fully connected projection, mapping the 512 features per time-step of the last convolutional layer to embeddings of dimension 100. The final PASE representation is produced by a non-affine BN layer that normalizes by the mean and variance of each dimension.

Note that, similarly to common speech feature extractors based on the short-time Fourier transform, we emulate an overlapping sliding window using a set of convolutions. The convolution, in fact, employs a sliding kernel over the signal that extracts localized patterns at different time shifts. In our case, we use stride factors $S > 1$ for most of the convolutional blocks, such that the input signal is decimated in time by a factor of 160. Therefore, given an input waveform of T samples, the amount of output feature vectors (frames) is $N = \frac{T}{160}$. At 16 kHz, this is equivalent to a 10 ms stride, similar to common speech processing pipelines. As such, PASE could easily replace previous feature extractors. The receptive field of the encoder is about 150 ms, making each output frame aware of a relatively large temporal context. The key motivation behind having a problem-agnostic speech encoder is to have a representative structure that can be used across tasks that require speech as input. We will see in the rest of this chapter how we can leverage the representation that this encoder yields out of waveforms.

6.1.2 Workers

Workers are fed by the encoded representation and solve seven self-supervised tasks, defined as regression or binary discrimination tasks (Fig. 6.1). In all cases, workers are based on very small feed-forward networks, composed of a single hidden layer of 256 units with PReLU activation (the only exception is the waveform worker, see below). Notice that we here employ simple networks on purpose. This way, we encourage the encoder, and not the workers, to discover high-level features that can be successfully exploited even by classifiers with limited capacity.

We first consider the use of regression workers, which break down the signal components at many levels in an increasing order of abstraction. These workers are trained to minimize the mean squared error (MSE) between the target features and the network predictions (again the waveform worker is an exception, see below). Features are extracted with `Librosa` (McFee et al., 2019) and `Pysptk` (Yamamoto et al., 2019) using default parameters, if not stated otherwise. As regression workers we consider:

- **Waveform:** we predict the input waveform in an auto-encoder fashion. This is the only worker that is not a single MLP, as we have a decimated latent representation that must be interpolated back to time domain. This is why three deconvolutional blocks (see section 3.1.3) are used with strides 4, 4, and 10, thus up-sampling by a factor 160. After these, an MLP of 256 PReLU units is used with a single output unit per time-step. This worker then learns to reconstruct waveforms by means of mean absolute error (L_1) minimization. The choice of L_1 is driven by robustness, as the speech distribution is very peaky and zero-centered with prominent outliers. This is the same type of regression used in section 5.1.

- **Log power spectrum (LPS):** as with the next features, we compute it using a Hamming window of 25 ms and a step size of 10 ms, with 1025 frequency bins per time step. These features are measured in dBs.

- **Mel-frequency cepstral coefficients (MFCC):** we extract 20 coefficients from 40 mel filter banks (FBANKs).

- **Prosody:** we also predict four basic prosodic features per frame, namely the interpolated logarithm of the fundamental frequency (log $F0$), voiced/unvoiced probability, zero-crossing rate, and energy. These features are called "Prosody", inheriting a terminology often used in emotion recognition (Neumann and Vu, 2017; A. Paeschke, 1999). Importantly, these features are correlated with intonation, expressiveness, and voicing.

All these features are normalized by their first two statistical moments, as explained later in section 6.1.3.

Next, we also consider three binary discrimination tasks, learning a higher level of abstraction than that of signal features. These tasks rely on a pre-defined sampling strategy that draws an anchor $\mathbf{x}_a$, a positive $\mathbf{x}_p$, and a negative $\mathbf{x}_n$ sample from the pool of PASE-encoded representations available in the training set. The reference anchor $\mathbf{x}_a$ is an encoded frame extracted from a random sentence, while $\mathbf{x}_n$ and $\mathbf{x}_p$ are encodings drawn using the different sampling strategies described below. An MLP then minimizes the following formulation of the binary cross-entropy:

$$L = \mathbb{E}_{X_p}[\log(g(\mathbf{x}_a, \mathbf{x}_p))] + \mathbb{E}_{X_n}[\log(1 - g(\mathbf{x}_a, \mathbf{x}_n))],$$

where g is the discriminator function, and $\mathbb{E}_{X_p}$ and $\mathbb{E}_{X_n}$ denote the expectation over positive and negative samples, respectively. Intuitively, by minimizing L, the model learns a speech embedding such that positive examples end up closer to their anchors than the corresponding negatives. Notice that the encoder and the discriminators are not adversarial here as in the case of GANs (Goodfellow, 2016), but must cooperate to derive good representations. In this work, we explore the following approaches to sample positive and negative examples:

- **Local info max (LIM):** as proposed in Ravanelli and Bengio (2019), we draw the positive sample from the same sentence of the anchor and a negative sample from another random utterance, which likely belongs to a different speaker. Since the speaker identity is a reliable constant factor within random features of the same utterance, this worker can learn a representation that embeds this kind of information. Hence, in this case $\mathbf{x}_a$ and $\mathbf{x}_p$ will be two random PASE frames from the current utterance, whereas $\mathbf{x}_n$ will be a random PASE frame from another random utterance.

- **Global info max (GIM):** in this and the subsequent worker, we compare global representations rather than local ones. The anchor representation is obtained by averaging all the PASE-encoded frames of a random utterance within a long random chunk of 1 s. The positive sample is similarly derived from another random chunk within the same sentence, while the negative one is obtained from another sentence. This way, we encourage the encoder to learn representations containing high-level information on the input sequence, that are hopefully complementary to those learned by LIM. GIM is also related to Deep InfoMax (Hjelm et al., 2019), which recently proposed to exploit local and global samples to learn image representations.

- **Sequence predicting coding (SPC):** in this case, the anchor is a single frame, while positive and negative samples are randomly extracted from its future and past elements. In particular, x_p contains 5 consecutive future frames, while x_n gathers 5 consecutive past ones. To make the task less trivial, we avoid sampling inside the current-frame receptive field (150 ms). On the other hand, to avoid making this task too complex or even unfeasible, we sample up to 500 ms away from the anchor. We expect this worker to capture information about the sequential order of the frames and the signal causality, encouraging PASE to embed a longer time contextual information. This approach is similar to the sampling strategy used in the contrastive predicting coding work (van den Oord and Vinyals, 2017). The main difference is that our negative sample is extracted from the past of the same sentence, rather than coming from a different one.

6.1.3 Self-supervised Training

Encoder and workers are jointly trained with backpropagation by optimizing a total loss that is computed as the average of each worker cost. Within the encoder, the gradients coming from the workers are thus averaged as well, and the optimization step will update its parameters pointing to a direction that is a compromise among all the worker losses (Serrà et al., 2018). To balance the contribution of each regression loss, we standardize all worker outputs using their mean and variance train set statistics, before computing the regression loss (either MSE or L_1). The encoder and the workers are optimized with Adam (Kingma and Ba, 2015), using an initial learning rate of $5 \cdot 10^{-4}$ which is halved every 30 epochs. We use mini-batches of 32 waveform chunks, each with 16 k samples corresponding to 1 s at a 16 kHz sampling rate. The system is trained for 150 epochs (i.e., until the validation losses reach a plateau for all the workers).

6.1.4 Usage in Supervised Classification Problems

The representations discovered by the encoder can be later used for supervised classification in different ways. One possibility is to keep the encoder frozen while training the classifier (PASE-Frozen). The encoder is thus used as a standard feature extractor and the features do not dynamically change during training. A better way consists of fine-tuning both the encoder and classifier during supervised training (PASE-FineTuned). This way, the extracted features are further optimized to better adapt themselves to the application of interest. For comparison, our results also include the case where PASE is trained on the supervised task from scratch, with random initialization (PASE-Supervised).

6.2 Corpora and Tasks

The self-supervised training of PASE is performed with the portion of the LibriSpeech dataset (Panayotov et al., 2015) used in Ravanelli and Bengio (2019). Speech sentences have been randomly selected to exploit about 15 s of training material for each of the 2484 speakers.

To assess the quality of the learned representations, we consider three supervised problems: (1) speaker identification (Speaker-ID), (2) speech emotion classification (Emotion), and (3) automatic speech recognition (ASR)[1]. For speaker identification, we use the VCTK dataset (Veaux et al., 2017), which contains 109 speakers with different English accents. To make this task more challenging and realistic regarding a possible need for exportability of PASE to low resource environments, we consider a subset of it that only contains 11 s of training for each speaker. For emotion recognition, we use the English utterances of the INTERFACE dataset (Hozjan et al., 2002). This corresponds to approximately 3 h for training, 40 min for validation, and 30 min for test. In this dataset we have two speakers (male and female) and eight emotions to be recognized: *anger*, *disgust*, *fear*, *neutral fast loud*, *neutral*, *joy*, *surprise*, and *sadness*.

For speaker and emotion recognition, the predicted posterior probabilities are averaged over all the time frames and we take the class with the highest score. To evaluate the capability of PASE to learn phoneme representations, a first set of ASR experiments is performed with the standard TIMIT dataset (Garofolo et al., 1993). Next, to assess our approach in more challenging noisy and reverberant conditions, in Section 6.3.3 we use the DIRHA dataset (Ravanelli et al., 2015). Training and validation sets are based on the original WSJ-5k corpus (consisting of 7138 sentences uttered by 83 speakers) that is contaminated with a set of impulse responses measured in a real apartment. The test set is composed of 409 WSJ sentences uttered by six American speakers and is based on real recordings in a domestic environment with a reverberation time of 0.7 s and an average signal-to-noise ratio of about 10 dB. ASR experiments are performed with the `PyTorch-Kaldi` toolkit (Ravanelli et al., 2019) and are based on the DNN-HMM framework. The DNN is trained to predict context-dependent phones and an HMM decoder is later employed to retrieve the sequence of phonemes for TIMIT or words for DIRHA (using the language models of the Kaldi recipes (Povey et al., 2011)). The DNN labels were derived by performing a forced alignment procedure using Kaldi (Povey et al., 2011).

[1]This work is a product of the collaboration with Mirco Ravanelli and Yoshua Bengio. Mirco Ravanelli conducted the speech recognition experiments.

6.3 Results

In this section three types of results are presented. First, an ablation study that helps us assess the importance of each worker, so that we can ensure that all of them contribute to the learning strategy. Secondly, the performance of PASE features in the different conditions in which it can be used, described in section 6.1.4. Finally, we evaluate PASE capacity to transfer its representation to a distant speech recognition environment, hence working in noisy acoustic conditions that were not seen during self-supervision. Throughout the different results, classification accuracies for speaker and emotion recognition are at utterance level, whereas speech recognition results report phoneme level accuracy.

6.3.1 Worker Ablation

First of all, we study whether all considered workers contribute to the final accuracy of PASE, and assess their impact on different target problems. To do so, we retrain the encoder discarding one of the workers at a time. We then extract PASE features (using the frozen encoder described in section 6.1.4), and we use them to feed MLP classifiers that solve the considered supervised problems. The experiments in this section are conducted with simple MLP classifiers based on a single layer, except for ASR, where we use three layers.

The classification accuracies of Table 6.1 show that no worker is dispensable. The best results are achieved with all workers, and we never observe performance improvements when discarding any of them. Nevertheless, while some workers are helpful for all the speech tasks, the benefits of some others turn out to be more application-dependent. For instance, Waveform, LPS, and MFCC regressors are generally helpful for all the applications, since they force the encoded representation to retain low-level information of the speech signal itself. The MFCC worker, in particular, is the most crucial one since it injects valuable prior knowledge on the most important frequency bands of the speech sequence. The prosody worker, instead, has a remarkable and expectable impact on emotion recognition only (+131% in relative error). This is due to the fact that our prosody features are correlated with intonation, expressiveness, and voicing, which are crucial clues for detecting emotion. LIM and GIM seem to be more helpful for Speaker-ID and Emotion rather than for ASR. These workers are designed to extract high-level information of speech that can be better exploited by higher-level classification tasks. A similar trend is observed for the SPC worker. This tends to extract longer contextual information, which turns out to be helpful for speaker and emotion recognition (+16% and in +13% relative error, respectively). The adopted receptive field of 150 ms, instead, embeds a context large enough for a DNN-HMM ASR system, as observed in Ravanelli and Omologo (2018).

6.3.2 Comparison with Standard Features

We now compare our PASE representations with more standard features such as MFCCs and FBANKs (McFee et al., 2019). Despite being proposed more than 40 years ago in Davis and Mermelstein (1980), these coefficients are still the most common speech features, and it is not easy to find alternatives that consistently outperform them. To provide a more fair comparison, MFCCs and FBANK are gathered in context windows that embed contextual information of about 150 ms (similar to the

TABLE 6.1: Accuracies using PASE and an MLP as classifier. Rows below the "all workers" model report absolute accuracy loss when discarding each worker for self-supervised training.

Model	Classification accuracy [%]		
	Speaker-ID *(VCTK)*	Emotion *(INTERFACE)*	ASR *(TIMIT)*
PASE (All workers)	97.5	88.3	81.1
– Waveform	−1.3	−3.9	−0.3
– LPS	−1.5	−5.3	−0.5
– MFCC	−2.4	−3.2	−0.7
– Prosody	−0.5	−5.3	−0.1
– LIM	−0.8	−1.3	−0.0
– GIM	−0.6	−0.5	−0.3
– SPC	−0.4	−1.6	−0.0

TABLE 6.2: Accuracy comparison on the considered classification tasks using MLPs and RNNs as classifiers.

Model	Classification accuracy [%]					
	Speaker-ID *(VCTK)*		Emotion *(INTERFACE)*		ASR *(TIMIT)*	
	MLP	RNN	MLP	RNN	MLP	RNN
MFCC	96.9	72.3	90.8	91.1	81.1	84.8
FBANK	98.4	75.1	94.1	92.8	80.9	85.1
PASE-Supervised	97.0	80.5	93.8	92.8	82.1	84.7
PASE-Frozen	97.3	82.5	91.5	92.8	81.4	84.7
PASE-FineTuned	**99.3**	**97.2**	**97.7**	**97.0**	**82.9**	**85.3**

receptive field of the encoder). MFCCs are also augmented with their first and second derivatives. As mentioned, we also compare with the purely supervised version of PASE, trained from scratch on the target task.

Table 6.2 shows the classification accuracies obtained with both MLP and RNN classifiers based on GRU. The hyper-parameters of all classifiers (number of hidden layers and neurons, learning rate, batch sizes, dropout rates, etc.) are independently tuned on the validation set and for each problem. PASE features provide most of the times a performance better than MFCCs and FBANKs, even when freezing the encoder (PASE-Frozen). The performance improvement becomes more evident when pre-training the encoder and fine-tuning it with the supervised task of interest (PASE-FineTuned). This approach consistently provides the best performance over all the tasks and classifiers considered here, also outperforming the PASE-Supervised baseline. Our best Speaker-ID result compares favorably with some recent works on the same dataset, such as Wang et al. (2019) and Chang et al. (2017). The phoneme accuracy of 85.3% on the TIMIT dataset (an error rate of 14.7%) is a competitive performance as well, especially when compared to state-of-the-art results that do not use complex techniques as system combination, speaker adaptation, or multiple steps of lattice rescoring and decoding (Povey et al., 2011; Ravanelli et al., 2019; Ravanelli et al., 2018; Michálek and Vanek, 2018).

TABLE 6.3: Word error rate (WER) obtained on the DIRHA corpus.

	WER [%]
MFCC	35.8
FBANK	34.0
PASE-Supervised	33.5
PASE-Frozen	32.5
PASE-FineTuned	**29.8**

6.3.3 Transferability

Finally, we study the exportability of PASE to acoustic conditions that are very different from the clean one used to train it. Table 6.3 reports the results obtained with the DIRHA dataset, which contains speech signals characterized by considerable noise and reverberation. We here employ the same version of PASE encoder used so far (trained on clean LibriSpeech data) coupled with a GRU classifier. Interestingly, PASE clearly outperforms the other systems. Even the frozen version of PASE overtakes FBANKs, MFCCs, and the supervised training baseline. PASE-FineTuned also outperforms previous results obtained with the standard SincNet model (Ravanelli and Bengio, 2018a). This result suggests the ability of PASE to effectively transfer its representation abstractions to different acoustic scenarios.

6.4 PASE Embeddings for Text-to-Speech

In TTS, an important dimension of study apart from naturalness is the acoustic mapping adaptability to generate voices from new speakers (i.e. unseen during training). In this section we study the use of PASE embeddings to build both a multi-speaker acoustic model for TTS based on SampleRNN, as well as a speaker adaptation mechanism[2]. We feed the acoustic model with speaker acoustically-dependent representations that enrich the waveform generation more than embeddings unrelated to these factors, like typical one-hot codes (Álvarez et al., 2019).

6.4.1 Multi-Speaker SampleRNN Acoustic Model

SampleRNN, introduced as a state of the art neural vocoder in chapter 2, was designed as an autoregressive likelihood based model (see section 3.2.1) by Mehri et al. (2016). In our work we use it as a multi-speaker acoustic model that directly predicts the waveform samples out of linguistic, prosodic and identity features. As depicted in Fig. 6.2, SampleRNN propagates data from upper to bottom layers and from previous to next time steps. In each layer, different scales of previous samples are used to condition the lower layer until it predicts the next sample in the last layer. The lowest layer is called sample-level layer and the others are called frame-level layers. We want to control both the content of the speech and the speaker identity, so we inject these two conditionings apart from previous samples to the system.

In order to inject identity information, we first experimented injecting the conditioning frames at the top-level layer (as in existing vocoding applications of SampleRNN (Barbany et al., 2018)). However this solution had difficulties to generate

[2]This work is the product of a collaboration with the master student David Álvarez and he conducted the TTS experiments and SampleRNN implementation.

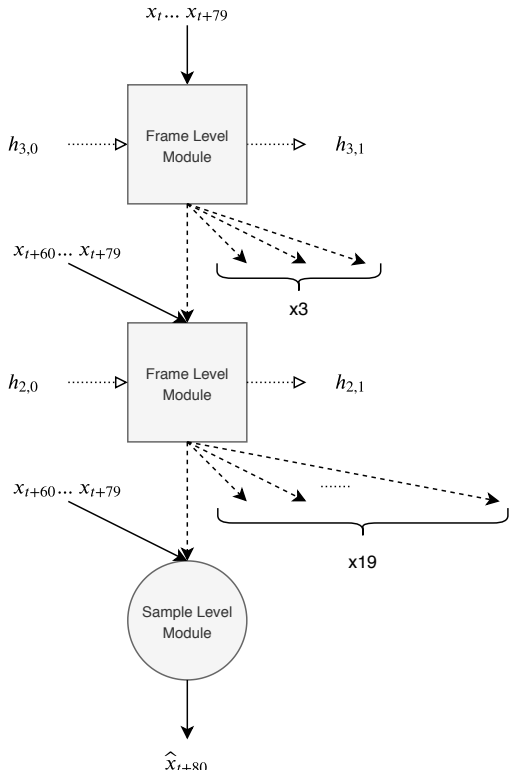

FIGURE 6.2: Original SampleRNN architecture with 2 frame-level layers and upsampling ratios {4, 20}

intelligible speech in our acoustic model for multiple identities, or at least to converge fast enough. To solve that, we use a similar strategy as in Zhou et al. (2018), combining both the original inputs with a global feature vector composed of speaker and linguistic features. These new feature vectors are used as inputs in the frame and sample level layers and are then concatenated with the other inputs before getting into the recurrent units.

Given a speaker i with its feature vector $\boldsymbol{e}_i \in \mathbb{R}^E$ and the linguistic feature vector of a certain time interval $\boldsymbol{l}_{\Delta t} \in \mathbb{R}^V$, we use a linear layer to obtain a global conditioning vector $\boldsymbol{c}_{i,\Delta t} \in \mathbb{R}^C$. Fig. 6.3 gives an overall scheme of this conditioning methodology along with other inputs of frame-level layers. Besides, the initial states of the frame-level layers $h_{z,0}$ are learned via backpropagation. For the case of the sample-level module, it follows a similar structure but without time-step conditioning.

6.4.2 Acoustic Seed

The speaker identity vectors can be introduced either with embedded one-hot codes as in previous approaches (van den Oord et al., 2016b; Pascual, 2016), or with some feature extractor out of speech signals that serve as seeds (Chen et al., 2019; Jia et al., 2018), like PASE. Fig. 6.4[3] shows the result of projecting the average PASE features per utterance, colorized per speaker label in two different datasets (i.e. averaging the frames in time for a full utterance). These are t-SNE projections, configured with a perplexity of 30 and converging during 1 k iterations. We can see how each speaker

[3]The PASE version used in these plots follows a slightly different configuration as that of section 6.1 in terms of architecture and some additional regression task, but equally proves our point of clustered identities.

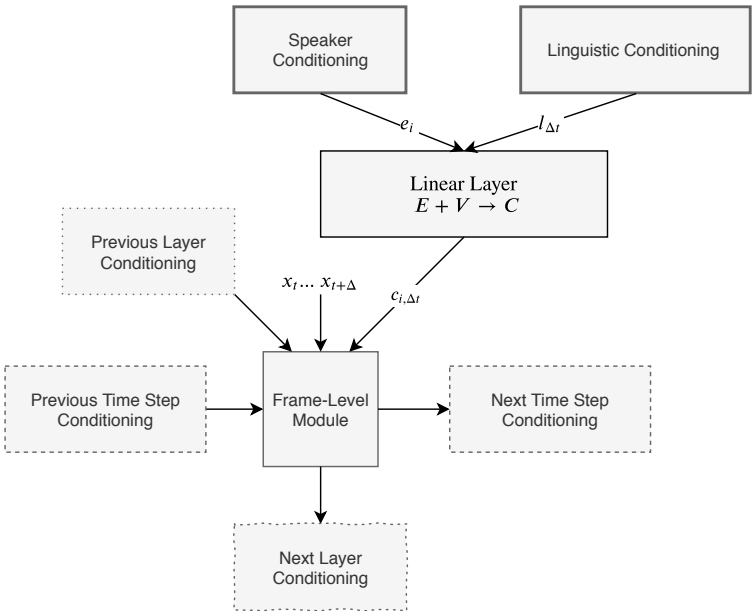

FIGURE 6.3: Combination of speaker and linguistic features along with other SampleRNN model-specific conditioning inputs

is clustered separately with homogeneous regions, which are prone to be local in the PASE hyperspace given the nature of the t-SNE projection (Maaten and Hinton, 2008).

Due to these observations, it is reasonable to use average PASE embeddings to describe speaker identities out of a frozen pre-trained PASE. We call these averaged embeddings acoustic seeds. We then experiment with training a TTS acoustic model per conditioning type: one-hot codes and PASE codes. We also explore the capability of these acoustic seeds to encode out-of-corpus speaker identities to do speaker adaptation without retraining. Our goal is that using PASE embeddings as speaker descriptors, the system is able to generalize even to unseen identities.

6.4.3 Experimental Setup

To train the acoustic models, we use utterances from two different datasets: VCTK (Veaux et al., 2017) and CMU Arctic (Kominek and Black, 2004). We decide to use speakers from both datasets because CMU Arctic contains more recorded speech per speaker, but it does not contain enough speakers to include a high variability factor for our experiments. Additionally, to avoid excessively modelling of silences, we trim them to a maximum length of 100 ms with the help of a voice activity detector. To train the base models, we select the 20 speakers of each gender with most speech duration, allocating a total amount of 40 speakers for this purpose. Then, for each of those speakers, we select a total of 45 s of speech for the validation split and another 45 s to perform objective tests. The remaining data of each speaker is then allocated to the training split, obtaining an unbalanced training dataset where the less representative speaker (VCTK speaker) has about 10 minutes of speech and the most representative one (CMU Arctic speaker) reaches the 64 min. In total, the sum of the individual contributions of the 40 speakers reaches the 12 h of training data and 30 min. of validation and test data.

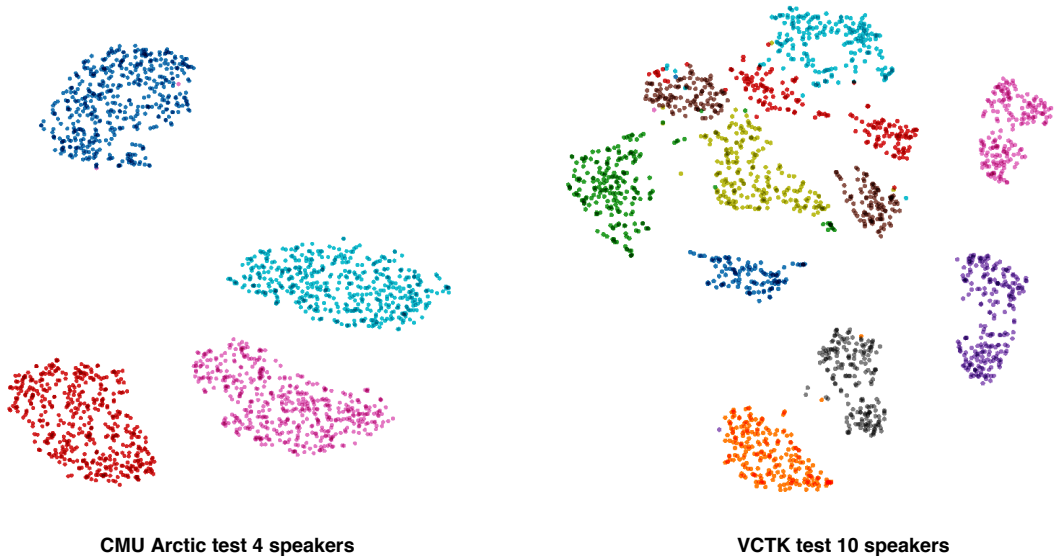

FIGURE 6.4: t-SNE projections of 4 CMU test speakers (left) and 10 VCTK test speakers (right) in clean conditions.

For the adaptation phase, we select 5 different random speakers per gender (10 speakers) and allocate enough data in the training split to experiment with seed signal lengths T up to 120 s. Note that we do not adjust the model weights, but we still require a training set of utterances to have enough T samples to build the acoustic seed. We then assess the adapted speaker similarity towards the target identity using a different test split with 180 s of data per speaker and a set of objective metrics.

We objectively evaluate the performance of our experiments by comparing the original utterances with the ones generated by our systems in two terms. First we have direct likelihood metrics as SampleRNN is a classifier computing the probability of the next waveform sample given the previous ones. This means we get a score that correlates with perceptual quality in the likelihood validation and test curves. Secondly, we have spectral distortion measures that depict how good is the waveform generation embedding speaker, prosodic and content characteristics in the speech (without an explicit modeling of spectral features themselves). Hence, we measure the MCD (in dB) and the RMSE of F0 (in Hz), as usually done in acoustic models assessment for TTS. Their formulations follow the same ones introduced in chapter 4 section 4.4.4.

6.4.4 Results

Fig. 6.6 illustrates the likelihood convergence in validation and the acoustic distortion metrics in test for four different model variations: .

- Linguistic features + one-hot codes
- Linguistic features + log $F0$ contour + one-hot codes
- Linguistic features + PASE codes
- Linguistic features + log $F0$ contour + PASE codes

We chose to train each model for 50 epochs for comparability purposes, but the different models can converge further as shown in these results. Nonetheless, some

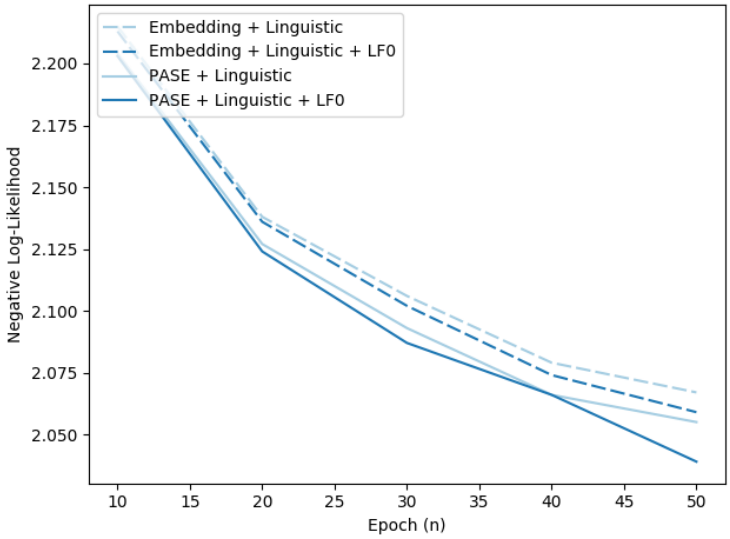

FIGURE 6.5: Validation negative log-likelihood loss. Lower loss indicates higher likelihood of the generated waveforms to be like the ground-truth ones, hence more similar to the actual speech.

evidences appear at this point worth commenting. First, in Fig.6.5 we can clearly see that the two experiments using our acoustic embeddings converge quicker than with the standard approach overall terms of validation likelihood. The similar trend is shown for the acoustic distortions shown in Fig. 6.6. The peaks we observe at some points in the spectral distortions are probably a reflect of under-optimized models, but we can still observe a clear convergence acceleration trend in the validation loss for models involving PASE embeddings that make these TTS variations comparable.

Moreover, the distortions of the models with log $F0$ contours as inputs are lower than those that lack them. This is expected because these contours supply long-term information that is useful for SampleRNN to retain far-past information better than it might in its internal states, as observed first in van den Oord et al. (2016b). Nevertheless, the experiment with PASE embeddings that lacks log $F0$ generally obtains better results than those of the one-hot embeddings with the log $F0$ contours. We hypothesize that this happens due to PASE embeddings capturing some bias in each speaker prosodic traits, but we still have an averaged representation in this case so it does perform worse than when we use PASE embeddings with the prosodic contours where dynamic long-term information is still fully present. These observations allow us to make a preliminar study of a speech generation task where PASE can be applied, apart from the previous classification ones presented in section 6.2. However, further development of this research direction will require also a subjective evaluation, as well as models with further convergence, where they reach a level of naturalness to be competitive to the state of the art.

Regarding the speaker adaptation experiment, we only take the PASE embeddings (without log $F0$ contours) model to proceed. This way we decouple any interference in prosodic or acoustic modeling coming from the log $F0$ contour. For each new speaker, we randomly sample chunks of lengths $T = \{1, 10, 60, 120\}$ seconds to build their time-average acoustic PASE embedding. Then, the 10 different speaker distortion curves (one per adaptation speaker) for both RMSE and MCD are averaged and depicted in Fig. 6.7. There we observe that the more seed signal we have, the less distortion we obtain across speakers, especially for MCD. The decreasing

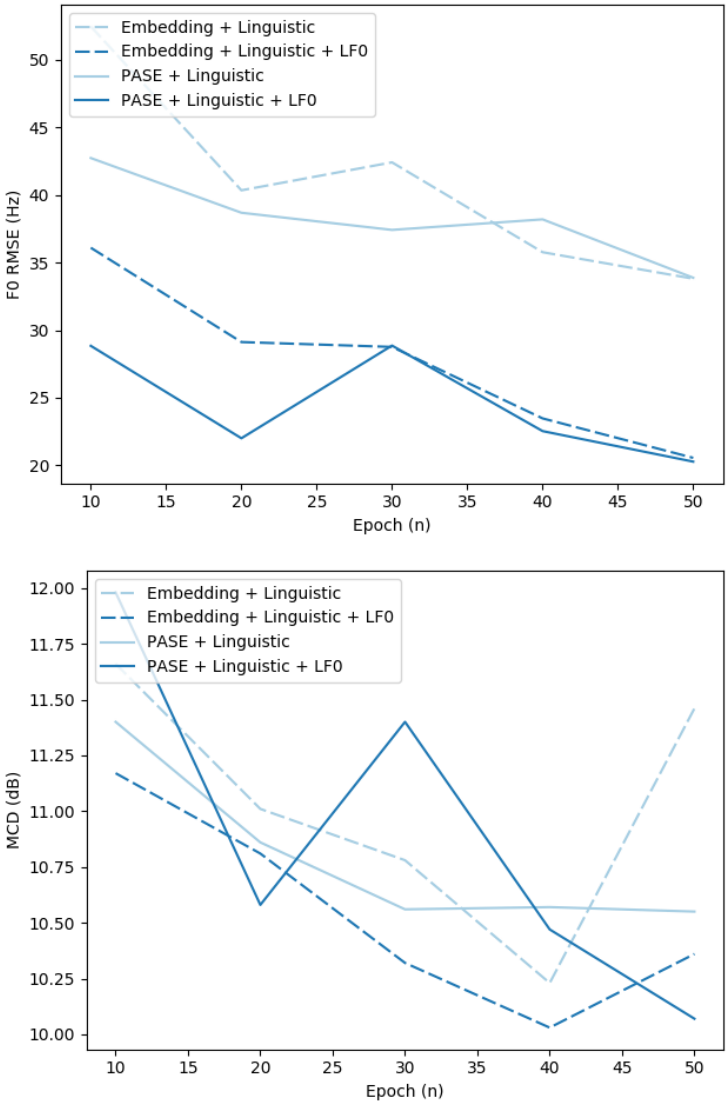

FIGURE 6.6: Top: RMSE of *F*0 for new speakers with respect to the length of speech used for the embedding generation. Bottom: MCD for new speakers with respect to the length of speech used for the embedding generation.

distortion trend depending on the amount of seed speech shows so far a promising direction towards generalizing speaker identities without fine-tuning TTS models. However, better optimized models are a potential need as shown in these results as models could converge further, hence biasing the adaptation curves to lower distortion rates with the same seed lengths. Qualitative results are available online in an audio samples webpage[4]. Informal listenings by the authors and different colleagues suggest that PASE embedding improve slightly the quality upon one-hot codes, and that systems are better if F0 is included as input in prosodic terms for both cases. Nonetheless, after a further development of more tunned acoustic models a formal subjective evaluation should be performed to establish this comparison firmly. The system proposed allows to generate speech with the identity of unseen speakers providing just the PASE embedding. As a general rule, 10 s are required to capture a similar identity to the target one. On the other hand, for the quality, the

[4] `http://veu.talp.cat/samplernn_pase`

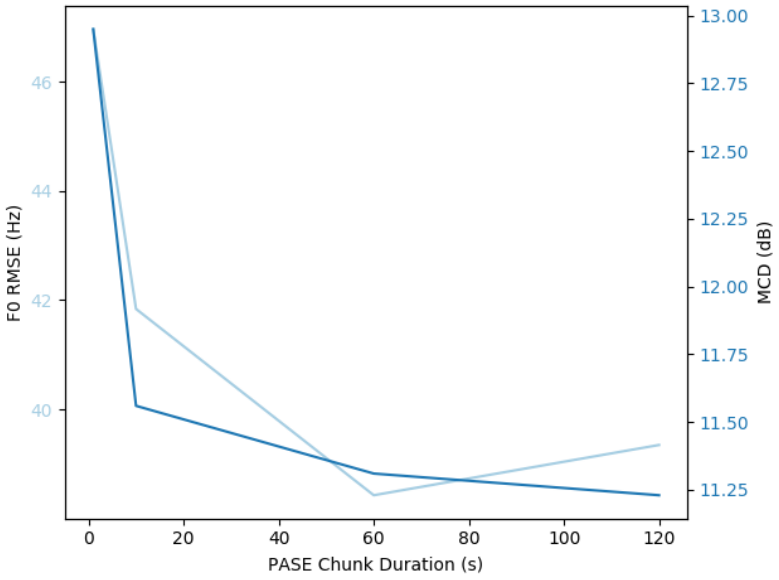

FIGURE 6.7: MCD and RMSE of *F0* for new speakers with respect to the length of speech used for the embedding generation

acoustic model does benefit from average embeddings with longer segments (10-60 s), potentially to avoid sampling only major silence regions which would include to identity and makes it sound more distorted.

6.5 Conclusion

The proposal in this chapter was twofold. On the one hand, we propose a multi-task self-supervised approach to learn speech representations. On the other hand, we provide an effective and exportable speech encoder that conveys waveforms into a sequence of latent embeddings. As evidenced by the considered problems, the discovered embeddings turn out to carry important information of the speech signal, related to, at least, speaker-identity, phonemes, and emotional cues. Learnt embeddings also showed their potential for of transferability to different datasets, tasks, and acoustic conditions. PASE is easily extendable as a semi-supervised framework and can embed in the future many other self-supervised tasks. After checking the information stored in the PASE embeddings with the supervised classification tasks, we also try PASE to extract speaker identifiers out of some seed voice from an arbitrary speaker, injecting them into a speech synthesis system to imitate the seed voice. The distortion results obtained in the first preliminary experiments, built upon a SampleRNN acoustic model, suggest that PASE allows for faster convergence than usual discrete one-hot codes. Additionally, this mechanism also allows us to do speaker adaptation by creating new seed PASE identifiers out of example utterances from target speakers. This requires no retraining of the SampleRNN nor the PASE itself, and proofs to be more effective specially when large utterances are available for the target speaker to minimize the spectral distortion towards the target speaker.

Chapter 7

Summary and Future Perspectives

In this thesis several questions have been addressed to study the possibilities of deep learning in the new context of speech processing and speech generation technologies: Can we make speech generation systems more efficient and yet maintain the same synthesized speech quality level? Can we also build an end-to-end deep generative model that works efficiently to do speech enhancement, agnostically to the speaker identity and the noise conditions? Does this model transfer easily its performance to new languages and noises? Can we go beyond the speech denoising task and truly unveil the generative capabilities of the model, hence reconstructing clean speech out of distorted signals? Finally, can be build compact and informative speech representations with unsupervised learning techniques? Can these embeddings be used simultaneously to work on several tasks that require speech as input? These questions, that cover a broad set of research directions within speech processing, motivated the investigations presented in this thesis (section 1.1). In the light of the results presented in this thesis, we can repeatedly answer: *yes*.

7.1 Summary of Contributions

The first problem tackled in this thesis is presented in chapter 4, in the context of text-to-speech (TTS). There we answer the question on whether we can increase efficiency of the recurrent structures that give state of the art results in two-stage parametric speech synthesis systems, and yet reach competitive results. The solution is found in the pseudo-recurrent structures, where quasi recurrent neural networks (QRNNs) seem to be most effective as an alternative to RNNs.

Secondly, we took the research direction of deep generative models, proposing an end-to-end speech enhancement model based on GANs. In this case we proposed a fully convolutional design that consumes noisy waveforms and generates clean waveforms in its output. The model, with its fully convolutional design, as well as with the GAN setup it conforms, is much more efficient than its autoregressive counterparts, reaching performance quotas beyond real-time (see section 5.3.1). We also explore the capabilities of this model to adapt its operability to new languages once it is trained in English. We demonstrate a quick adaptation capacity to new language conditions under a minimum amount of training data (in the range of seconds). Additionally, the model seems to be invariant to being adapted to new noises towards the test phase. This may happen due to a decision boundary between speech and non-speech intrinsic to the model regeneration process. Hence, adapting to the new language speech patterns seems more critical to reduce the final distortion, as measured in section 5.4. Furthermore, the use of a deep generative model can be potentially extended beyond a denoising condition, and as such we conduct the experimentation of applying the proposed SEGAN to a generalized set

of distortion conditions that directly affect the speech signal, and hence its missing and corrupted components must be regenerated.

Finally, unsupervised learning for speech representations has been explored in chapter 6. More concretely, a multi-task self-supervised framework has been designed to train a fully convolutional speech encoder that consumes waveforms and returns compact frames conveying multiple levels of features. This design mixes both types of constructions: the raw signal consumption of end-to-end models with the prior knowledge based design of appropriate self-supervised tasks (section 6.1). This model is named problem-agnostic speech encoder (PASE), as it yields frames useful for different tasks that require speech as input, but it is trained without any information about the end-tasks it can be applied to. Precisely, the self-supervised training methodology ensures no superficial features are learned, as well as their exportability. First, we exemplified the exportability of PASE by plugging classifiers trained on top of the feature extractor to make speaker recognition, emotion recognition and speech recognition. Secondly, we framed the encoder in speech generation. To that end, we applied PASE as an identity descriptor for speaker identities in a multi-speaker TTS system. The TTS results suggest that PASE contains relevant acoustic information that allow for a faster convergence in the TTS acoustic model when compared to discrete one-hot identifiers.

To sum up, this thesis contributes to the study of deep learning architectures applicable to speech synthesis, enhancement and processing in terms of efficiency, end-to-end designs and self-supervised learning applied to the construction of speech representations. More specifically:

- It provides a comprehensive overview of the available literature on speech generation for both text-to-speech and speech-to-speech applications, as well as unsupervised learning to create speech representations. This includes a review from classic concatenative techniques until the most recent deep generative models that shift these problems towards end-to-end approaches.

- It provides a study of pseudo-recurrent alternatives to do speech synthesis. These alternatives include the QRNN and the self-attention model, both widely used and successful in natural language applications. This study yields a much more efficient linguistic–acoustic mapping system using QRNNs, with which we achieve a 11.2 times speedup on CPU and 3.3 times on GPU with respect to the LSTM based model.

- It proposes, to the best of the author's knowledge, the first approach to tackle a speech processing task with generative adversarial networks (GANs). More specifically, it provides an end-to-end speech enhancement GAN (SEGAN), designed as a fully convolutional network to be fully parallelizable (hence efficient).

- It provides an extensive study on the improvements to be made in an auto-encoder architecture like SEGAN's. The adaptability of the proposed system to new languages and noises is also studied.

- It proposes a new trend with a long-term perspective to tackle a more generalized speech enhancement framework, where different distortions of the signal have to be palliated in addition to the usual denoising and dereverberation tasks. This comes jointly with the recent application of deep generative models to speech-to-speech transformations, as the one shown in this thesis.

- It provides a multi-task setup to adapt the SEGAN system to obtain more perceptually relevant results in terms of palliating the proposed distortions within the generalized speech enhancement problem.
- It proposes, to the best of the author's knowledge, the first self-supervised multi-task approach to learn abstract speech representations.
- It proposes, to the best of the author's knowledge, the first attempt to create a problem-agnostic speech encoder whose design allows for exportability and efficiency in the feature extraction process. The proposed encoder reflects the aim of the author to make an encoder ready to be used by speech researchers of different fields. The encoder is made to gather different levels of speech features, exploiting the huge statistical averaging benefits of deep learning to build robust representations absorbing variabilities in the data to abstract high-level representations.

The outcomes of the research presented in this thesis have been published in the form of several papers in international conferences, journals, blog posts and open source projects. A list of online demo pages with audible samples from the obtained results on speech generation is available in an annex to this thesis (Appendix A). The full list of the authors' publications is also provided in an annex to this thesis (Appendix B).

7.2 Some Future Perspectives

This thesis begins tackling two different problems, namely speech synthesis and speech enhancement, which can be related by the fact that both generate a clean, well structured speech signal. In the course of developing both systems, deep neural networks of different kinds have been used. Moreover, both end-to-end and non end-to-end approaches have been taken. In the process of developing these different systems, a convergence between speech enhancement and speech synthesis (usually known for text-to-speech applications) emerged. This potentially points towards a direction where a speech generation module, under the light of current deep generative models presented in section 3.2, may be plugged on top of a proper front-end that can extract features that represent a certain source signal to be converted. This source signal may either come from text as in TTS, or may come from another speech signal as in STS applications. Recent works prove the possibility to build systems that can do voice conversion and TTS simultaneously (Zhang et al., 2019), by sharing an acoustic model decoder by two specialized encoders. Similarly, our experiments shown in section 6.4 show that the problem-agnostic speech embeddings can also be used to tackle TTS. Hence, it is potentially true that there is a future convergence of the different STS and TTS applications to happen, where the different tasks will be tackled by the same model depending on what inputs do we feed.

A first possible point of convergence will be that of voice conversion and speech enhancement. This may come from a similar model to the one proposed in this thesis, the SEGAN (see chapter 5), where the decoder will contain an explicit speaker identifier to decode a changed identity, apart from being able to decode a clean representation of the distorted input. The identifier may be fed as current GAN conditionings through conditional normalization layers (Huang and Belongie, 2017; Karras et al., 2019), or as we recently proposed with a hyper-conditioning scheme as in Blow (Serrà et al., 2019). On the other hand, the identifiers themselves could be PASE

embeddings, as we have seen their effectiveness in retaining the speaker identity in the utterance average representation (see section 6.4). Also, owing to the effectiveness of PASE to represent the content as well as the speaker identity, a pre-trained PASE could extract both the local features in a frame by frame basis (content) and the global features (identity). Then, we could operate algebraically in that space to subtract the global features of the source, and then we would add the target speaker global representation. A decoder model would learn the correspondence between PASE frames and acoustic frames, and a neural vocoder would convert the frames to waveforms. This methodology would require training only a small part of the model, the one consuming PASE features and yielding acoustic features. Hence, less data would be required to optimize such a model with respect to an end-to-end alternative.

Another research direction made possible due to the PASE self-supervised design is the one of language discovery focused on low resource text-to-speech. This is aligned with the recently proposed zero-speech challenge, also named "TTS without T" (Dunbar et al., 2019). The goal of this task is to discover subword units in an unsupervised way by using a unit discovery dataset, and then align these units to the voice recording in a way that works best for the purpose of synthesizing novel utterances from novel speakers (Muthukumar and Black, 2014; Scharenborg et al., 2018). Hence, PASE representations should be discretized, probably in a process similar to the one of the vector quantized variational auto-encoder (van den Oord and Vinyals, 2017). The potential usefulness of discretizing PASE representations for this task is an open issue to be tackled, but potentially effective due to its proven factorization of the high-level features from the speech waveforms. Actually, as stated in the zero-speech challenge, an effective representation of the linguistic units can then be fed into a generative model that reconstructs a speech waveform conditioned on certain speaker identity, similar to voice conversion. Nonetheless, an important matter to consider in this case is the exploration of low bit-rate encodings of the speech, hence enforcing a high compression in the latent space, something not explored with PASE so far.

Furthermore, the use of acoustic losses in an end-to-end model like the SEGAN has proven to be effective (see section 5.6.1). Nevertheless, those acoustic losses are still using low-level spectral features such as log-power spectrum frames or prosodic contours like logF0. A convergence point between SEGAN and PASE can also be the use of pre-trained PASE features as targets for a self-supervised SEGAN training, with the use of PASE in a distillation framework (Hinton et al., 2015). In this case a frozen pre-trained PASE may be plugged on top of the generator network to enrich its learning process, potentially making it converge faster and to a more perceptually relevant solution than with the vanilla adversarial setup or with the plain use of low-level acoustic features in the discriminator. This setup could also be a viable option for a robust generalized speech enhancement model, designed with the mixture of end-to-end and prior knowledge based models.

Appendix A

Demo Pages: Audio Samples

Efficient Neural Acoustic Modeling in Text-to-Speech
http://veu.talp.cat/efftts

Speech Enhancement Generative Adversarial Network
http://veu.talp.cat/seganp

Whispered SEGAN
http://veu.talp.cat/whispersegan

Towards Generalized SEGAN
http://veu.talp.cat/gsegan

PASE for Multi-speaker Text-to-Speech and Speaker Adaptation
http://veu.talp.cat/samplernn_pase

Appendix B

Publications by the Author

B.1 Chapter 4

Self-Attention Linguistic-Acoustic Decoder
Santiago Pascual, Antonio Bonafonte, Joan Serrà. *IberSPEECH 2018*.

Exploring Efficient Neural Architectures for Linguistic–Acoustic Mapping in Text-To-Speech
Santiago Pascual, Joan Serrà, Antonio Bonafonte.
MDPI Applied Sciences. Special Issue IberSPEECH 2018: Speech and Language Technologies for Iberian Languages.

B.2 Chapter 5

Time-domain Speech Enhancement Using Generative Adversarial Networks
Santiago Pascual, Joan Serrà, Antonio Bonafonte. *Elsevier Speech Communication 2019.*

SEGAN: Speech Enhancement Generative Adversarial Network
Santiago Pascual, Antonio Bonafonte, Joan Serrà. *INTERSPEECH 2017*.

Language and Noise Transfer in Speech Enhancement Generative Adversarial Network
Santiago Pascual, Maruchan Park, Joan Serrà, Antonio Bonafonte, Kang-Hun Ahn.
ICASSP 2018.

Whispered-to-voiced Alaryngeal Speech Conversion with Generative Adversarial Networks
Santiago Pascual, Antonio Bonafonte, Joan Serrà, Jose A. González.
IberSPEECH 2018.

Towards Generalized Speech Enhancement with Generative Adversarial Networks
Santiago Pascual, Joan Serrà, Antonio Bonafonte. *INTERSPEECH 2019.*

B.3 Chapter 6

Learning Problem-Agnostic Speech Representations from Multiple Self-Supervised Tasks
Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, Yoshua Bengio.

INTERSPEECH 2019. Nominated for the best student paper award.

Problem-Agnostic Speech Embeddings for Multi-Speaker Text-to-Speech with SampleRNN
David Álvarez, Santiago Pascual, Antonio Bonafonte.
ISCA 10-th Speech Synthesis Workshop.

Multi-task self-supervised learning for Robust Speech Recognition
Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, Yoshua Bengio. *Submitted to ICASSP 2020.*

B.4 Byproducts

Sample Drop Detection for Distant-Speech Recognition with Asynchronous Devices Distributed in Space
Tina Raissi, Santiago Pascual, Maurizio Omologo. *Submitted to ICASSP 2020.*

Blow: a Single-Scale Hyperconditioned Flow for Non-Parallel Raw-Audio Voice Conversion
Joan Serrà, Santiago Pascual, Carlos Segura. *NeurIPS 2019*.

Wav2Pix: Speech-Conditioned Face Generation Using Generative Adversarial Networks
Amanda Duarte, Francisco Roldan, Miquel Tubau, Janna Escur, Santiago Pascual, Amaia Salvador, Eva Mohedano, Kevin McGuinness, Jordi Torres, Xavier Giro-i-Nieto. *ICASSP 2019*.

Spanish Statistical Parametric Speech Synthesis Using a Neural Vocoder
Antonio Bonafonte, Santiago Pascual, Georgina Dorca. *INTERSPEECH 2018*.

Multi-Speaker Neural Vocoder
Oriol Barbany, Antonio Bonafonte, Santiago Pascual. *IberSPEECH 2018*.

Real Non-Volume Preserving Voice Conversion
Santiago Pascual, Joan Serrà, Antonio Bonafonte. *NeurIPS LXAI Workshop 2018*.
`http://veu.talp.cat/santi_slides/rnvpvc.pdf`.

Towards a Universal Neural Network Encoder for Time Series
Joan Serrà, Santiago Pascual, Alexandros Karatzoglou. *CCIA 2018*.

B.5 Open Source Projects

B.5.0.1 Chapter 4

MUSA TTS (`PyTorch >= 0.4.1`**)**: `https://github.com/santi-pdp/musa_tts`.
9 stars & 4 forks on Nov 2019.

B.5.0.2 Chapter 5

SEGAN (`TensorFlow <= 0.12.1`**)**: `https://github.com/santi-pdp/segan`.
518 stars & 213 forks on Nov 2019.

SEGAN (`PyTorch >= 0.4.1`**)**: `https://github.com/santi-pdp/segan_pytorch`.
141 stars & 40 forks on Nov 2019.

B.5.0.3 Chapter 6

PASE (`PyTorch >= 1.0.1`**)**: `https://github.com/santi-pdp/pase`.
126 stars & 25 forks on Nov 2019.

B.6 Blog post Tutorials on Convolutional Neural Networks

Receptive Fields in Convolutional Neural Networks
Santiago Pascual. *Medium blogpost 2018*.
`https://tinyurl.com/santty128-rfconvs`.

How PyTorch Transposed Convs1D Work
Santiago Pascual. *Medium blogpost 2018*.
`https://tinyurl.com/santty128-deconv`.

Bibliography

A. Paeschke M. Kienast, W. Sendlmeier (1999). "F0-contours in emotional speech". In: *Proc. of Int. Con. of Phonetic Sciences (ICPhS)*, pp. 929–932.

Agiomyrgiannakis, Yannis (2015). "Vocaine the Vocoder and Applications in Speech Synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*.

Álvarez, David, Pascual, Santiago, and Bonafonte, Antonio (2019). "Problem-Agnostic Speech Embeddings for Multi-Speaker Text-to-Speech with SampleRNN". In: *Proc. of 10th ISCA Speech Synthesis Workshop (SSW10)*, pp. 35–39. DOI: 10.21437/SSW.2019-7. URL: http://dx.doi.org/10.21437/SSW.2019-7.

Arandjelović, Relja and Zisserman, Andrew (2018). "Objects that Sound". In: *Proc. of the European Conf. on Computer Vision (ECCV)*, pp. 451–466.

Arik, Sercan Ö et al. (2017). "Deep Voice: Real-time Neural Text-to-Speech". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. Vol. 70, pp. 195–204. URL: http://proceedings.mlr.press/v70/arik17a.html.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon (2017). "Wasserstein Generative Adversarial Networks". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 214–223. URL: http://proceedings.mlr.press/v70/arjovsky17a.html.

Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E (2016). "Layer normalization". In: *ArXiv: 1607.06450*.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: http://arxiv.org/abs/1409.0473.

Barbany, Oriol, Bonafonte, Antonio, and Pascual, Santiago (2018). "Multi-Speaker Neural Vocoder". In: *Proc. of the Int. Conf. IberSPEECH*, pp. 30–34. DOI: 10.21437/IberSPEECH.2018-7. URL: https://doi.org/10.21437/IberSPEECH.2018-7.

Barker, Jon et al. (2018). "The Fifth 'CHiME' Speech Separation and Recognition Challenge: Dataset, Task and Baselines". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1561–1565. URL: http://dx.doi.org/10.21437/Interspeech.2018-1768.

Bengio, Yoshua (2011). "Deep Learning of Representations for Unsupervised and Transfer Learning". In: *Proc. of the Int. Conf. on Unsupervised and Transfer Learning Workshop*, pp. 17–37. URL: http://dl.acm.org/citation.cfm?id=3045796.3045800.

Bengio, Yoshua et al. (2006). "Greedy Layer-Wise Training of Deep Networks". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*.

Berouti, M., Schwartz, R., and Makhoul, J. (1979). "Enhancement of speech corrupted by acoustic noise". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 4, pp. 208–211.

Berthelot, David, Schumm, Thomas, and Metz, Luke (2017). "Began: Boundary Equilibrium Generative Adversarial Networks". In: *ArXiv:1703.10717*.

Biadsy, Fadi et al. (2019). "Parrotron: An End-to-End Speech-to-Speech Conversion Model and its Applications to Hearing-Impaired Speech and Speech Separation". In: *ArXiv:1904.04169*.

Bińkowski, Mikołaj et al. (2019). "High Fidelity Speech Synthesis with Adversarial Networks". In: *ArXiv:1909.11646*.

Bishop, Christopher M (1994). "Mixture Density Networks". In: *Technical Report*.

Black, Alan W., Zen, Heiga, and Tokuda, Keiichi (2007). "Statistical Parametric Speech Synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1229–1232. URL: https://doi.org/10.1109/ICASSP.2007.367298.

Bonafonte, Antonio et al. (2006). "TC-STAR: Specifications of language resources and evaluation for speech synthesis". In: *Proc. of the Int. Conf. on Language Resources and Evaluation (LREC)*, pp. 311–314.

Bonafonte, Antonio et al. (2008). "Corpus and Voices for Catalan Speech Synthesis". In: *Proc. of the Int. Conf. on Language Resources and Evaluation (LREC)*, pp. 3325–3329.

Bradbury, James et al. (2017). "Quasi-Recurrent Neural Networks". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=H1zJ-v5xl.

Brock, Andrew, Donahue, Jeff, and Simonyan, Karen (2019). "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=B1xsqj09Fm.

Bulut, Murtaza, Narayanan, Shrikanth S, and Syrdal, Ann K (2002). "Expressive Speech Synthesis Using a Concatenative Synthesizer". In: *Proc. on the Int. Conf. on Spoken Language Processing (ICSLP)*.

Cahn, Janet Elizabeth (1989). "Generating expression in synthesized speech". PhD thesis. Massachusetts Institute of Technology.

Chan, William et al. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4960–4964.

Chang, Shiyu et al. (2017). "Dilated Recurrent Neural Networks". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pp. 77–87. URL: http://papers.nips.cc/paper/6613-dilated-recurrent-neural-networks.pdf.

Chen, Jing et al. (2018). "The Effect of F0 Contour on the Intelligibility of Speech in the Presence of Interfering Sounds for Mandarin Chinese". In: *The Journal of the Acoustical Society of America* 143.2, pp. 864–877.

Chen, Sin-Horng, Hwang, Shaw-Hwa, and Wang, Yih-Ru (1998). "An RNN-based Prosodic Information Synthesizer for Mandarin Text-to-Speech". In: *IEEE Trans. on Speech and Audio Processing* 6.3, pp. 226–239.

Chen, Yutian et al. (2019). "Sample Efficient Adaptive Text-to-Speech". In: *Proc. on the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=rkzjUoAcFX.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. URL: https://www.aclweb.org/anthology/D14-1179.

Chorowski, Jan et al. (2019). "Unsupervised Speech Representation Learning Using WaveNet Autoencoders". In: *IEEE Trans. on Audio, Speech, and Language Processing* abs/1901.08810.12, pp. 2041–2053.

Chung, Junyoung et al. (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *Proc. on the Deep Learning and Representation Learning Workshop*.

Chung, Yu-An et al. (2019). "An Unsupervised Autoregressive Model for Speech Representation Learning". In: *Proc. of the Conf. of the Int. Speech Communication*

Association (INTERSPEECH), pp. 146–150. URL: http://dx.doi.org/10.21437/Interspeech.2019-1473.

Clark, Leigh et al. (2018). "The State of Speech in HCI: Trends, Themes and Challenges". In: *arXiv preprint arXiv:1810.06828*.

Clark, Robert AJ, Richmond, Korin, and King, Simon (2007). "Multisyn: Open-domain unit selection for the Festival speech synthesis system". In: *Speech Communication* 49.4, pp. 317–330.

Dahl, George E. et al. (2012). "Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition". In: *IEEE Trans. on Audio, Speech, and Language Processing* 20.1, pp. 30–42.

Davis, Steven B. and Mermelstein, Paul (1980). "Comparison of Parametric Representation for Monosyllabic Word Recognition in Continuously Spoken Sentences". In: *IEEE Trans. on Acoustics, Speech and Signal Processing* 28.4, pp. 357–366.

Deco, Gustavo and Brauer, Wilfried (1995). "Higher Order Statistical Decorrelation Without Information Loss". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. Vol. 7, pp. 247–254. URL: http://papers.nips.cc/paper/901-higher-order-statistical-decorrelation-without-information-loss.pdf.

Dehak, Najim et al. (2010). "Front-end Factor Analysis for Speaker Verification". In: *IEEE Trans. on Audio, Speech, and Language Processing* 19.4, pp. 788–798.

Dendrinos, Markos, Bakamidis, Stelios, and Carayannis, George (1991). "Speech Enhancement From Noise: A Regenerative Approach". In: *Speech Communication* 10.1, pp. 45–57.

Devlin, Jacob et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proc. of the Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423.

Ding, Ni et al. (2012). "Speaker Variability in Emotion Recognition - an Adaptation Based Approach". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5101–5104.

Dinh, Laurent, Sohl-Dickstein, Jascha, and Bengio, Samy (2017). "Density Estimation using Real NVP". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=HkpbnH9lx.

Doersch, Carl and Zisserman, Andrew (2017). "Multi-task Self-Supervised Visual Learning". In: *Proc. of Int. Conf. on Computer Vision (ICCV)*, pp. 2070–2079.

Donahue, Chris, Li, Bo, and Prabhavalkar, Rohit (2018). "Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5024–5028.

Donahue, Chris, McAuley, Julian J., and Puckette, Miller S. (2019). "Adversarial Audio Synthesis". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=ByMVTsR5KQ.

Duarte, Amanda et al. (2019). "Wav2Pix: Speech-Conditioned Face Generation Using Generative Adversarial Networks". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8633–8637.

Dunbar, Ewan et al. (2019). "The Zero Resource Speech Challenge 2019: TTS Without T". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1088–1092. URL: http://dx.doi.org/10.21437/Interspeech.2019-2904.

El Ayadi, Moataz, Kamel, Mohamed S, and Karray, Fakhri (2011). "Survey on speech emotion recognition: Features, classification schemes, and databases". In: *Pattern Recognition* 44.3, pp. 572–587.

Ephraim, Y. (1992). "Statistical-model-based speech enhancement systems". In: *Proc. of the IEEE* 80.10, pp. 1526–1555.

Ephraim, Yariv and Van Trees, Harry L (1995). "A signal subspace approach for speech enhancement". In: *IEEE Trans. on Speech and Audio Processing* 3.4, pp. 251–266.

Erdogan, Hakan et al. (2015). "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 708–712.

Erhan, Dumitru et al. (2009). "Visualizing higher-layer features of a deep network". In: *Technical Report, University of Montreal*.

Erro, Daniel et al. (2011). "Improved HNM-based vocoder for statistical synthesizers." In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1809–1812.

Fagan, Michael J et al. (2008). "Development of a (silent) speech recognition system for patients following laryngectomy". In: *Medical Engineering & Physics* 30.4, pp. 419–425.

Fan, Yuchen et al. (2015). "Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis". In: *Proc of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4475–4479.

Fernandez, Raul et al. (2014). "Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks." In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 2268–2272. URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2014/i14_2268.pdf.

Frey, Brendan J, Hinton, Geoffrey E, and Dayan, Peter (1996). "Does the Wake-sleep Algorithm Produce Good Density Estimators?" In: *Advances in Neural Information Processing Systems 8*. Ed. by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo. MIT Press, pp. 661–667. URL: http://papers.nips.cc/paper/1153-does-the-wake-sleep-algorithm-produce-good-density-estimators.pdf.

Fu, Szu-Wei et al. (2017). "Raw Waveform-based Speech Enhancement by Fully Convolutional Networks". In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017*, pp. 006–012.

Fu, Szu-Wei et al. (2018). "End-to-End Waveform Utterance Enhancement for Direct Evaluation Metrics Optimization by Fully Convolutional Neural Networks". In: *IEEE/ACM Trans. on Audio, Speech and Language Processing* 26.9, pp. 1570–1584.

Garcia-Garcia, Alberto et al. (2017). "A Review on Deep Learning Techniques Applied to Semantic Segmentation". In: *ArXiv:1704.06857*.

Garofolo, J. S. et al. (1993). *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM*.

Gibiansky, Andrew et al. (2017). "Deep Voice 2: Multi-Speaker Neural Text-to-Speech". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pp. 2962–2970. URL: http://papers.nips.cc/paper/6889-deep-voice-2-multi-speaker-neural-text-to-speech.pdf.

Gidaris, Spyros, Singh, Praveer, and Komodakis, Nikos (2018). "Unsupervised Representation Learning by Predicting Image Rotations". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=S1v4N2l0-.

Gong, Yuan and Poellabauer, Christian (2018). "Impact of Aliasing on Deep CNN-Based End-to-End Acoustic Models". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 2698–2702.

Gonzalez, Jose A et al. (2017a). "Direct Speech Reconstruction From Articulatory Sensor Data by Machine Learning". In: *IEEE/ACM Trans. on Audio, Speech, and Language Processing* 25.12, pp. 2362–2374.

Gonzalez, Jose A et al. (2017b). "Evaluation of a silent speech interface based on magnetic sensing and deep learning for a phonetically rich vocabulary". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 3986–3990.

Goodfellow, Ian (2016). "NIPS 2016 tutorial: Generative adversarial networks". In: *ArXiv:1701.00160*.

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pp. 2672–2680. URL: `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron (2016). *Deep learning*. MIT press.

Graves, Alex and Jaitly, Navdeep (2014). "Towards End-To-End Speech Recognition with Recurrent Neural Networks". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 1764–1772. URL: `http://proceedings.mlr.press/v32/graves14.html`.

Gulrajani, Ishaan et al. (2017). "Improved Training of Wasserstein GANs". In: *Proc. in Advances in Neural Information Processing Systems (NIPS)*, pp. 5767–5777. URL: `http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf`.

Hardcastle, William J and Hewlett, Nigel (2006). *Coarticulation: Theory, data and techniques*. Cambridge University Press.

He, Kaiming et al. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 1026–1034.

— (2016). "Deep Residual Learning for Image Recognition". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Henter, Gustav Eje et al. (2018). "Analysing Shortcomings of Statistical Parametric Speech Synthesis". In: *ArXiv:1807.10941*.

Heusel, Martin et al. (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Proc. in the Advances in Neural Information Processing Systems (NIPS)*, pp. 6626–6637. URL: `http://papers.nips.cc/paper/7240-gans-trained-by-a-two-time-scale-update-rule-converge-to-a-local-nash-equilibrium.pdf`.

Higuchi, Takuya et al. (2017). "Adversarial training for data-driven speech enhancement without parallel corpus". In: *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 40–47.

Hinterleitner, Florian, Weiss, Benjamin, and Möller, Sebastian (2016). "Influence of corpus size and content on the perceptual quality of a unit selection MaryTTS voice". In: *Proc. of the IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 680–685.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition". In: *IEEE Signal processing magazine* 29.

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff (2015). "Distilling the Knowledge in a Neural Network". In: *ArXiv:1503.02531*.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R (2006). "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786, pp. 504–507.

Hinton, Geoffrey E., Osindero, Simon, and Teh, Yee-Whye (2006). "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18, pp. 1527–1554.

Hjelm, R. Devon et al. (2019). "Learning deep representations by mutual information estimation and maximization". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=Bklr3j0cKX.

Hochreiter, Sepp and Schmidhuber, Jürgen (1997). "Long Short-Term Memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hochreiter, Sepp et al. (2001). "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. IEEE Press.

Hozjan, Vladimir et al. (2002). "Interface Databases: Design and Collection of a Multilingual Emotional Speech Database." In: *Proc. of the Int. Conf. on Language Resources and Evaluation (LREC)*, p. 174.

Hu, Qiong et al. (2015). "Fusion of Multiple Parameterisations for DNN-Based Sinusoidal Speech Synthesis with Multi-Task Learning". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 854–858. URL: https://www.isca-speech.org/archive/interspeech_2015/papers/i15_0854.pdf.

Hu, Y. and Loizou, P. C. (2008). "Evaluation of Objective Quality Measures for Speech Enhancement". In: *IEEE Trans. on Audio, Speech, and Language Processing* 16.1, pp. 229–238.

Huang, Gao et al. (2018). "CondenseNet: An Efficient DenseNet using Learned Group Convolutions". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2752–2761.

Huang, Xuedong et al. (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR.

Huang, Xun and Belongie, Serge (2017). "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization". In: *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 1501–1510.

Hunt, Andrew J and Black, Alan W (1996). "Unit selection in a concatenative speech synthesis system using a large speech database". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*. Vol. 1. IEEE, pp. 373–376.

India, Miquel, Safari, Pooyan, and Hernando, Javier (2019). "Self Multi-Head Attention for Speaker Recognition". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 4305–4309. URL: http://dx.doi.org/10.21437/Interspeech.2019-2616.

Ioffe, Sergey and Szegedy, Christian (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

Irie, Kazuki et al. (2019). "Language Modeling with Deep Transformers". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 3905–3909. URL: http://dx.doi.org/10.21437/Interspeech.2019-2225.

Isola, Phillip et al. (2017). "Image-to-Image Translation with Conditional Adversarial Networks". In: *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 5967–5976.

ITU-T (2007). *P.862.2: Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs*.

Jansen, Aren et al. (2018). "Unsupervised Learning of Semantic Audio Representations". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 126–130.

Jarrett, Kevin et al. (2009). "What is the best multi-stage architecture for object recognition?" In: *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*. IEEE, pp. 2146–2153.

Jia, Ye et al. (2018). "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis". In: *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*. URL: http://papers.nips.cc/paper/7700-transfer-learning-from-speaker-verification-to-multispeaker-text-to-speech-synthesis.pdf.

Jin, Zeyu et al. (2016). "Cute: A concatenative method for voice conversion using exemplar-based unit selection". In: *Proc of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5660–5664.

Jin, Zeyu et al. (2018). "FFTNet: A real-time speaker-dependent neural vocoder". In: *Proc of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 2251–2255.

Kalchbrenner, Nal et al. (2018). "Efficient Neural Audio Synthesis". In: *ArXiv:1802.08435*.

Kameoka, Hirokazu et al. (2018). "StarGAN-VC: Non-parallel many-to-many voice conversion using star generative adversarial networks". In: *Proc. of the IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 266–273.

Kang, Shiyin, Qian, Xiaojun, and Meng, Helen (2013). "Multi-distribution deep belief network for speech synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 8012–8016.

Karafiát, Martin et al. (2011). "iVector-based discriminative adaptation for automatic speech recognition". In: *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU*. IEEE, pp. 152–157.

Karras, Tero, Laine, Samuli, and Aila, Timo (2019). "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410.

Kawahara, Hideki (2006). "STRAIGHT, exploitation of the other aspect of VOCODER: Perceptually isomorphic decomposition of speech sounds". In: *Acoustical science and technology* 27.6, pp. 349–353.

Kenter, Tom et al. (2019). "CHiVE: Varying Prosody in Speech Synthesis with a Linguistically Driven Dynamic Hierarchical Conditional Variational Network". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 3331–3340. URL: http://proceedings.mlr.press/v97/kenter19a.html.

Kim, Jae-Bok, Park, Jeong-Sik, and Oh, Yung-Hwan (2011). "On-line speaker adaptation based emotion recognition using incremental emotional information". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4948–4951.

King, Simon (2011). "An introduction to statistical parametric speech synthesis". In: *Sadhana* 36.5, pp. 837–852.

Kingma, Diederik P and Ba, Jimmy (2015). "Adam: A Method for Stochastic Optimization". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: http://arxiv.org/abs/1412.6980.

Kingma, Diederik P. and Dhariwal, Prafulla (2018). "Glow: generative flow with invertible 1x1 convolutions". In: *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31, pp. 10215–10224. URL: http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.pdf.

Kingma, Diederik P. and Welling, Max (2014). "Auto-Encoding Variational Bayes". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*.

Kobayashi, Kazuhiro et al. (2017). "Statistical Voice Conversion with WaveNet-Based Waveform Generation". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1138–1142. URL: http://dx.doi.org/10.21437/Interspeech.2017-986.

Kominek, John and Black, Alan W. (2004). "The CMU Arctic speech databases". In: *Proc. of the 5th ISCA Speech Synthesis Workshop (SSW)*, pp. 223–224. URL: https://www.isca-speech.org/archive_open/archive_papers/ssw5/ssw5_223.pdf.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in neural information processing systems*, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kubichek, R (1993). "Mel-cepstral distance measure for objective speech quality assessment". In: *Proc. of the IEEE Pacific Rim Conf. on Communications Computers and Signal Processing (PACRIM)*. Vol. 1. IEEE, pp. 125–128.

Kumar, Kundan et al. (2019). "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis". In: *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*. URL: http://papers.nips.cc/paper/9629-melgan-generative-adversarial-networks-for-conditional-waveform-synthesis.pdf.

LeCun, Yann et al. (1989). "Generalization and network design strategies". In: *Connectionism in perspective*. Vol. 19. Citeseer.

Li, Naihan et al. (2019). "Neural Speech Synthesis with Transformer Network". In: Proc. of the Advancement of Artificial Intelligence (AAAI). URL: https://aaai.org/ojs/index.php/AAAI/article/view/4642/4520.

Lim, Jae and Oppenheim, A. (1978). "All-pole modeling of degraded speech". In: *IEEE Trans. on Acoustics, Speech, and Signal Processing* 26.3, pp. 197–210.

Lim, Jae Hyun and Ye, Jong Chul (2017). "Geometric GAN". In: *ArXiv:1705.02894*.

Ling, Zhen-Hua et al. (2015). "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends". In: *IEEE Signal Processing Magazine* 32.3, pp. 35–52.

Loizou, Philipos C. (2013). *Speech Enhancement: Theory and Practice*. 2nd. Boca Raton, FL, USA: CRC Press, Inc. ISBN: 1466504218, 9781466504219.

Lu, Heng, King, Simon, and Watts, Oliver (2013a). "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis". In: *Proc. of the 8th ISCA Speech Synthesis Workshop (SSW)*, pp. 281–285. URL: https://www.isca-speech.org/archive/ssw8/papers/ssw8_261.pdf.

Lu, Jiasen et al. (2017). "Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 375–383.

Lu, Xugang et al. (2013b). "Speech Enhancement Based on Deep Denoising Autoencoder". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 436–440. URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2013/i13_3259.pdf.

Lučić, Mario et al. (2018). "Are GANs Created Equal? A Large-Scale Study". In: *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 700–709. URL: http://papers.nips.cc/paper/7350-are-gans-created-equal-a-large-scale-study.pdf.

Lučić, Mario et al. (2019). "High-Fidelity Image Generation With Fewer Labels". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. Vol. 97, pp. 4183–4192. URL: http://proceedings.mlr.press/v97/lucic19a.html.

Maas, Andrew L et al. (2012). "Recurrent Neural Networks for Noise Reduction in Robust ASR". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 22–25. URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2012/i12_0022.pdf.

Maaten, Laurens van der and Hinton, Geoffrey (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Mao, Xudong et al. (2017). "Least Squares Generative Adversarial Networks". In: *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*. IEEE, pp. 2813–2821. URL: http://openaccess.thecvf.com/content_ICCV_2017/papers/Mao_Least_Squares_Generative_ICCV_2017_paper.pdf.

McFee, Brian et al. (2019). *librosa/librosa: 0.6.3*. URL: https://doi.org/10.5281/zenodo.2564164.

Mehri, Soroush et al. (2016). "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=SkxKPDv5xl.

Meng, Zhong, Li, Jinyu, Gong, Yifan, et al. (2018). "Cycle-Consistent Speech Enhancement". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1165–1169. URL: http://dx.doi.org/10.21437/Interspeech.2018-2409.

Merity, Stephen, Keskar, Nitish Shirish, and Socher, Richard (2018). "An Analysis of Neural Language Modeling at Multiple Scales". In: *ArXiv:1803.08240*.

Michálek, J. and Vanek, J. (2018). "A Survey of Recent DNN Architectures on the TIMIT Phone Recognition Task". In: *TSD*. Vol. 11107. Lecture Notes in Computer Science. Springer, pp. 436–444.

Misra, Ishan, Zitnick, C. Lawrence, and Hebert, Martial (2016). "Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification". In: *Proc. of the European Conf. on Computer Vision (ECCV)*, pp. 527–544.

Miyato, Takeru et al. (2018). "Spectral Normalization for Generative Adversarial Networks". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=B1QRgziT-.

Morise, Masanori, Yokomori, Fumiya, and Ozawa, Kenji (2016). "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications". In: *IEICE Trans. on Information and Systems* 99.7, pp. 1877–1884.

Muthukumar, Prasanna Kumar and Black, Alan W (2014). "Automatic discovery of a phonetic inventory for unwritten languages for statistical speech synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 2594–2598.

Nakamura, Keigo et al. (2011). "Estimation of fundamental frequency from surface electromyographic data: EMG-to-F 0". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 573–576.

Nakamura, Keigo et al. (2012). "Speaking-aid systems using GMM-based voice conversion for electrolaryngeal speech". In: *Speech Communication* 54.1, pp. 134–146.

Narayanan, Arun and Wang, DeLiang (2013). "Ideal ratio mask estimation using deep neural networks for robust speech recognition". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7092–7096.

Neal, Radford M (1992). "Connectionist learning of belief networks". In: *Artificial intelligence* 56.1, pp. 71–113.

Neumann, M. and Vu, N. T. (2017). "Attentive Convolutional Neural Network Based Speech Emotion Recognition: A Study on the Impact of Input Features, Signal Length, and Acted Speech". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1263–1267. URL: http://dx.doi.org/10.21437/Interspeech.2017-917.

Niwa, Jumpei et al. (2018). "Statistical voice conversion based on WaveNet". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5289–5293.

Odena, Augustus, Dumoulin, Vincent, and Olah, Chris (2016). "Deconvolution and checkerboard artifacts". In: *Distill* 1.10, e3.

Odena, Augustus, Olah, Christopher, and Shlens, Jonathon (2017). "Conditional Image Synthesis with Auxiliary Classifier GANs". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 2642–2651. URL: http://proceedings.mlr.press/v70/odena17a.html.

Oh, Tae-Hyun et al. (2019). "Speech2Face: Learning the Face Behind a Voice". In: *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 7539–7548.

Ortega-Garcia, J. and Gonzalez-Rodriguez, J. (1996). "Overview of speech enhancement techniques for automatic speaker recognition". In: *Proc. of the Int. Conf. on Spoken Language Processing (ICSLP)*. Vol. 2, pp. 929–932.

Owens, A. et al. (2018). "Learning Sight from Sound: Ambient Sound Provides Supervision for Visual Learning". In: *International Journal of Computer Vision* 126.10, pp. 1120–1137.

Paliwal, Kuldip, Wójcicki, Kamil, and Shannon, Benjamin (2011). "The importance of phase in speech enhancement". In: *Speech Communication* 53.4, pp. 465 –494.

Panayotov, V. et al. (2015). "Librispeech: An ASR corpus based on public domain audio books". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210.

Park, Se Rim and Lee, Jinwon (2017). "A fully convolutional neural network for speech enhancement". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*. URL: http://dx.doi.org/10.21437/Interspeech.2017-1465.

Parveen, S. and Green, P. (2004). "Speech enhancement with missing data techniques using recurrent neural networks". In: *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 733–736.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua (2013). "On the difficulty of training recurrent neural networks". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 1310–1318. URL: http://proceedings.mlr.press/v28/pascanu13.html.

Pascual, Santiago (2016). "Deep Learning Applied to Speech Synthesis". MA thesis. Universitat Politècnica de Catalunya. URL: https://upcommons.upc.edu/bitstream/handle/2117/100133/msc-thesis-final.pdf.

Pascual, Santiago and Bonafonte, Antonio (2016a). "Multi-Output RNN-LSTM for Multiple Speaker Speech Synthesis and Adaptation". In: *Proc. of the European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 2325–2329.

— (2016b). "Multi-output RNN-LSTM for multiple speaker speech synthesis with α-interpolation model". In: *Proc. ISCA SSW9*. IEEE, pp. 112–117.

— (2016c). "Multi-output RNN-LSTM for multiple speaker speech synthesis with α-interpolation model". In: *9th ISCA Speech Synthesis Workshop*, pp. 119–124.

Pascual, Santiago, Bonafonte, Antonio, and Serrà, Joan (2017). "SEGAN: Speech Enhancement Generative Adversarial Network". In: *Proc. of the Conf. of the Int. Speech*

Communication Association (INTERSPEECH), pp. 3642–3646. URL: http://dx.doi.org/10.21437/Interspeech.2017-1428.

— (2018a). "Self-Attention Linguistic-Acoustic Decoder". In: *Proc. of the Int. Conf. IberSPEECH*, pp. 152–156.

Pascual, Santiago et al. (2018b). "Whispered-to-voiced Alaryngeal Speech Conversion with Generative Adversarial Networks". In: *Proc. of the Int. Conf. IberSPEECH*, pp. 117–121. URL: http://dx.doi.org/10.21437/IberSPEECH.2018-25.

Pascual, Santiago, Serrà, Joan, and Bonafonte, Antonio (2019a). "Exploring Efficient Neural Architectures for Linguistic–Acoustic Mapping in Text-To-Speech". In: *Applied Sciences* 9.16, p. 3391. URL: https://www.mdpi.com/2076-3417/9/16/3391/pdf.

— (2019b). "Time-domain Speech Enhancement Using Generative Adversarial Networks". In: *Speech Communication* 114, pp. 10–21. ISSN: 0167-6393. URL: http://www.sciencedirect.com/science/article/pii/S0167639319301359.

— (2019c). "Towards Generalized Speech Enhancement with Generative Adversarial Networks". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1791–1795. URL: http://dx.doi.org/10.21437/Interspeech.2019-2688.

Paszke, A. et al. (2017). "Automatic differentiation in PyTorch". In: *NIPS Workshop on The Future of Gradient-based Machine Learning Software & Techniques (NIPS-Autodiff)*.

Paszke, Adam et al. (2016). "Enet: A deep neural network architecture for real-time semantic segmentation". In: *ArXiv:1606.02147*.

Pathak, Deepak et al. (2016). "Context encoders: Feature learning by inpainting". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *The Journal of Machine Learning Research* 12, pp. 2825–2830.

Ping, Wei et al. (2018). "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=HJtEm4p6Z.

Ping, Wei, Peng, Kainan, and Chen, Jitong (2019). "ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=HklY120cYm.

Popham, Sara et al. (2018). "Inharmonic speech reveals the role of harmonicity in the cocktail party problem". In: *Nature communications* 9.1, p. 2122.

Povey, Daniel et al. (2011). "The Kaldi Speech Recognition Toolkit". In: *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Prenger, Ryan, Valle, Rafael, and Catanzaro, Bryan (2019). "Waveglow: A flow-based generative network for speech synthesis". In: *Proc of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3617–3621.

Qian, Yao et al. (2014). "On the training aspects of deep neural network (DNN) for parametric TTS synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3829–3833.

Qin, Shan and Jiang, Ting (2018). "Improved Wasserstein conditional generative adversarial network speech enhancement". In: *EURASIP Journal on Wireless Communications and Networking* 2018.1, p. 181.

Radford, Alec, Metz, Luke, and Chintala, Soumith (2016). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. URL: http://arxiv.org/abs/1511.06434.

Ravanelli, M. and Omologo, M. (2018). "Automatic context window composition for distant speech recognition". In: *Speech Communication* 101, pp. 34–44.

Ravanelli, M. et al. (2018). "Light Gated Recurrent Units for Speech Recognition". In: *IEEE Trans. on Emerging Topics in Computational Intelligence* 2.2, pp. 92–102.

Ravanelli, M., Parcollet, T., and Bengio, Y. (2019). "The PyTorch-Kaldi Speech Recognition Toolkit". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*.

Ravanelli, Mirco and Bengio, Yoshua (2018a). "Interpretable Convolutional Filters with SincNet". In: *Proc. of the NIPS Workshop on Interpretability and Robustness for Audio, Speech and Language (IRASL)*.

— (2018b). "Speaker Recognition from raw waveform with SincNet". In: *Proc. of the IEEE Spoken Language Technology Workshop (SLT)*.

— (2019). "Learning Speaker Representations with Mutual Information". In: *Proc. Interspeech 2019*, pp. 1153–1157. URL: http://dx.doi.org/10.21437/Interspeech.2019-2380.

Ravanelli, Mirco et al. (2015). "The DIRHA-English corpus and related tasks for distant-speech recognition in domestic environments". In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pp. 275–282.

Rethage, Dario, Pons, Jordi, and Serra, Xavier (2018). "A Wavenet for speech denoising". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5069–5073.

Rezende, Danilo and Mohamed, Shakir (2015). "Variational Inference with Normalizing Flows". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. Vol. 37, pp. 1530–1538. URL: http://proceedings.mlr.press/v37/rezende15.html.

Rosenstein, M. T. et al. (2005). "To transfer or not to transfer". In: *NIPS Workshop in Inductive Transfer: 10 Years Later*.

Saon, George et al. (2013). "Speaker adaptation of neural network acoustic models using i-vectors". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, pp. 55–59.

Scalart, P. and Filho, J. V. (1996). "Speech enhancement based on a priori signal to noise estimation". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 2, 629–632 vol. 2.

Scharenborg, Odette et al. (2018). "Linguistic unit discovery from multi-modal inputs in unwritten languages: Summary of the "Speaking rosetta" JSALT 2017 workshop". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4979–4983.

Schuller, Björn et al. (2013). "Paralinguistics in speech and language—State-of-the-art and the challenge". In: *Computer Speech & Language* 27.1, pp. 4–39.

Sennrich, Rico, Haddow, Barry, and Birch, Alexandra (2016). "Edinburgh Neural Machine Translation Systems for WMT 16". In: *Proc. of the Conf. on Machine Translation (WMT)*, pp. 371–376. URL: https://www.aclweb.org/anthology/W16-2323.

Serrà, J., Pascual, S., and Karatzoglou, A. (2018). "Towards a universal neural network encoder for time series". In: *Artificial Intelligence Research and Development*. Vol. 308. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 120–129.

Serrà, Joan, Pascual, Santiago, and Segura, Carlos (2019). "Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion". In: URL: http://papers.nips.cc/paper/8904-blow-a-single-scale-hyperconditioned-flow-for-non-parallel-raw-audio-voice-conversion.pdf.

Sethu, Vidhyasaharan, Ambikairajah, Eliathamby, and Epps, Julien (2008). "Phonetic and speaker variations in automatic emotion classification". In: *Proc. of the Annual Conf. of the Int. Speech Communication Association*.

Shen, Jonathan et al. (2018). "Natural TTS Synthesis by Conditioning Wavenet on Mel Spectrogram Predictions". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4779–4783.

Shivakumar, Prashanth Gurunath and Georgiou, Panayiotis G (2016). "Perception Optimized Deep Denoising AutoEncoders for Speech Enhancement." In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 3743–3747. URL: http://dx.doi.org/10.21437/Interspeech.2016-1284.

Skerry-Ryan, RJ et al. (2018). "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron". In: *ArXiv:1803.09047*.

Snyder, David et al. (2018). "X-vectors: Robust dnn embeddings for speaker recognition". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5329–5333.

Sotelo, Jose et al. (2017). "Char2Wav: End-to-end speech synthesis". In: *International Conference on Learning Representations (ICLR): Workshop Track*. Toulon, France.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Sun, Lifa et al. (2015). "Voice conversion using deep bidirectional long short-term memory based recurrent neural networks". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4869–4873.

Sundermann, David et al. (2006). "Text-independent voice conversion based on unit selection". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 1. IEEE, pp. I–I.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V (2014). "Sequence to Sequence Learning with Neural Networks". In: *Proc. of the Advances in Neural Information Srocessing Systems (NIPS)*, pp. 3104–3112.

Swietojanski, Pawel and Renals, Steve (2014). "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models". In: *Proc. of the IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 171–176.

Szegedy, Christian et al. (2015). "Going Deeper with Convolutions". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9.

Taal, Cees H et al. (2010). "A short-time objective intelligibility measure for time-frequency weighted noisy speech". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4214–4217.

— (2011). "An algorithm for intelligibility prediction of time–frequency weighted noisy speech". In: *IEEE Trans. on Audio, Speech, and Language Processing* 19.7, pp. 2125–2136.

Tamamori, Akira et al. (2017). "Speaker-Dependent WaveNet Vocoder". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1118–1122. URL: http://dx.doi.org/10.21437/Interspeech.2017-314.

Tamura, S. and Waibel, A. (1988). "Noise reduction using connectionist models". In: *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 553–556.

Taylor, Paul (2009). *Text-to-speech synthesis*. Cambridge university press.

Thiemann, Joachim, Ito, Nobutaka, and Vincent, Emmanuel (2013). "The diverse environments multi-channel acoustic noise database: A database of multichannel environmental noise recordings". In: *Journal of the Acoustical Society of America* 133.5, pp. 3591–3591.

Tieleman, T. and Hinton, G. (2012). *Lecture 6.5-RMSprop: divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning 4, 2.

Toda, Tomoki, Black, Alan W, and Tokuda, Keiichi (2008). "Statistical mapping between articulatory movements and acoustic spectrum using a Gaussian mixture model". In: *Speech Communication* 50.3, pp. 215–227.

Tokuda, Keiichi et al. (2000). "Speech parameter generation algorithms for HMM-based speech synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 3. IEEE, pp. 1315–1318.

Trigeorgis, George et al. (2016). "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5200–5204.

Tur, Gokhan and De Mori, Renato (2011). *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Uria, Benigno, Murray, Iain, and Larochelle, Hugo (2013). "RNADE: The real-valued neural autoregressive density-estimator". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pp. 2175–2183. URL: http://papers.nips.cc/paper/5060-rnade-the-real-valued-neural-autoregressive-density-estimator.pdf.

Uria, Benigno et al. (2015). "Modelling acoustic feature dependencies with artificial neural networks: Trajectory-RNADE". In: *Proc of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4465–4469.

Valentini-Botinhao, Cassia, Wu, Zhizheng, and King, Simon (2015). "Towards minimum perceptual error training for DNN-based speech synthesis". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 869–873. URL: https://www.isca-speech.org/archive/interspeech_2015/papers/i15_0869.pdf.

Valentini-Botinhao, Cassia et al. (2016). "Investigating RNN-based speech enhancement methods for noise-robust Text-to-Speech". In: *Proc. of the 9th ISCA Speech Synthesis Workshop*, pp. 146–152.

Valin, Jean-Marc and Skoglund, Jan (2019). "LPCNet: Improving neural speech synthesis through linear prediction". In: *Proc of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5891–5895.

van den Oord, Aaron, Vinyals, Oriol, et al. (2017). "Neural Discrete Representation Learning". In: *Proc. in the Advances in Neural Information Processing Systems (NIPS)*, pp. 6306–6315. URL: http://papers.nips.cc/paper/7210-neural-discrete-representation-learning.pdf.

van den Oord, Aaron, Kalchbrenner, Nal, and Kavukcuoglu, Koray (2016a). "Pixel Recurrent Neural Networks". In: *ArXiv:1601.06759* 48, pp. 1747–1756. URL: http://proceedings.mlr.press/v48/oord16.html.

van den Oord, Aaron et al. (2016b). "Wavenet: A generative model for raw audio". In: *ArXiv:1609.03499*.

van den Oord, Aaron et al. (2017). "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *ArXiv:1711.10433*.

van den Oord, Aaron, Li, Yazhe, and Vinyals, Oriol (2018). "Representation Learning with Contrastive Predictive Coding". In: *Arxiv: 1807.03748*.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Veaux, Christophe, Yamagishi, Junichi, and King, Simon (2013). "The Voice Bank Corpus: Design, Collection and Data Analysis of a Large Regional Accent Speech

Database". In: *Proc. of the Conf. of the Oriental Int. Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (COCOSDA)*, pp. 1–4.

Veaux, Christophe, Yamagishi, Junichi, MacDonald, Kirsten, et al. (2017). "CSTR VCTK corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit". In: *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*. URL: http://dx.doi.org/10.7488/ds/1994.

Wang, Dequan and Lim, Jae S (1982). "The unimportance of phase in speech enhancement". In: *IEEE Trans. on Acoustics, Speech, and Signal Processing* 30.4, pp. 679–681.

Wang, Jixuan et al. (2019). "Centroid-based deep metric learning for speaker recognition". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*.

Wang, Yuxuan, Narayanan, Arun, and Wang, DeLiang (2014). "On training targets for supervised speech separation". In: *IEEE/ACM Trans. on Audio, Speech and Language Processing* 22.12, pp. 1849–1858.

Wang, Yuxuan et al. (2017). "Tacotron: Towards End-to-End Speech Synthesis". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 4006–4010. URL: http://dx.doi.org/10.21437/Interspeech.2017-1452.

Wang, Yuxuan et al. (2018). "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis". In: *ArXiv:1803.09017*.

Weninger, Felix et al. (2014). "Discriminatively trained recurrent neural networks for single-channel speech separation". In: *Proc. of the IEEE Global Conf. on Signal and Information Processing (GlobalSIP)*.

Weninger, Felix et al. (2015). "Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR". In: *Proc. of the Int. Conf. on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pp. 91–99.

Williamson, Donald S and Wang, DeLiang (2017). "Time-frequency masking in the complex domain for speech dereverberation and denoising". In: *IEEE/ACM Trans. on Audio, Speech, and Language Processing* 25.7, pp. 1492–1501.

Wu, Zhizheng and King, Simon (2016). "Investigating gated recurrent networks for speech synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5140–5144.

Wu, Zhizheng et al. (2015). "A Study of Speaker Adaptation for DNN-Based Speech Synthesis". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*. URL: https://www.isca-speech.org/archive/interspeech_2015/papers/i15_0879.pdf.

Xia, Bingyin and Bao, Changchun (2013). "Speech Enhancement with Weighted Denoising Auto-Encoder". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 3444–3448. URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2013/i13_3444.pdf.

Xie, Saining et al. (2017). "Aggregated residual transformations for deep neural networks". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1492–1500.

Xu, Yong et al. (2014). "Cross-language transfer learning for deep neural network based speech enhancement". In: *Proc. of the Int. Symp. on Chinese Spoken Language Processing (ISCSLP)*, pp. 336–340.

— (2015). "A regression approach to speech enhancement based on deep neural networks". In: *IEEE/ACM Trans. on Audio, Speech and Language Processing* 23.1, pp. 7–19.

Yamamoto, Ryuichi, Felipe, Joao, and Blaauw, Merlijn (2019). *r9y9/pysptk: 0.1.14*. URL: https://github.com/r9y9/pysptk.

Yang, Li-Ping and Fu, Qian-Jie (2005). "Spectral subtraction-based speech enhancement for cochlear implant patients in background noise". In: *Journal of the Acoustical Society of America* 117.3, pp. 1001–1004.

Yang, Zhilin et al. (2019). "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *arXiv preprint arXiv:1906.08237*.

Young, Tom et al. (2018). "Recent trends in deep learning based natural language processing". In: *ieee Computational intelligenCe magazine* 13.3, pp. 55–75.

Yu, Dong et al. (2008). "A minimum-mean-square-error noise reduction algorithm on mel-frequency cepstra for robust speech recognition". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4041–4044.

Yu, Fisher and Koltun, Vladlen (2015). "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122*.

Zen, Heiga (2006). "An example of context-dependent label format for HMM-based speech synthesis in English". In: *The HTS CMUARCTIC demo* 133.

Zen, Heiga and Sak, Hasim (2015). "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis". In: *Proc. on the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4470–4474.

Zen, Heiga and Senior, Andrew (2014). "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3844–3848.

Zen, Heiga, Tokuda, Keiichi, and Black, Alan W (2009). "Statistical parametric speech synthesis". In: *Speech Communication* 51.11, pp. 1039–1064.

Zen, Heiga, Senior, Andrew, and Schuster, Mike (2013). "Statistical parametric speech synthesis using deep neural networks". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7962–7966.

Zhang, Han et al. (2018a). "Self-Attention Generative Adversarial Networks". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. URL: http://proceedings.mlr.press/v97/zhang19d.html.

Zhang, Meng et al. (2008). "Improving HMM based speech synthesis by reducing over-smoothing problems". In: *Proc. of the Int. Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, pp. 1–4.

Zhang, Mingyang et al. (2019). "Joint Training Framework for Text-to-Speech and Voice Conversion Using Multi-Source Tacotron and WaveNet". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*. URL: http://dx.doi.org/10.21437/Interspeech.2019-1357.

Zhang, Xiangyu et al. (2018b). "Shufflenet: An extremely efficient convolutional neural network for mobile devices". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 6848–6856.

Zhao, Zhong-Qiu et al. (2019). "Object detection with deep learning: A review". In: *IEEE Trans. on neural networks and learning systems*.

Zhou, Cong et al. (2018). "Voice Conversion with Conditional SampleRNN". In: *Proc. of the Conf. of the Int. Speech Communication Association (INTERSPEECH)*, pp. 1973–1977. URL: http://dx.doi.org/10.21437/Interspeech.2018-1121.