Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Disentangled Representations in Speech Processing Applications

by

Matthew Baas

Dissertation presented for the degree of Doctor of Philosophy (Electronic Engineering) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. Herman Kamper

2024

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2 June 2024 ....

Acknowledgements

I would like to sincerely thank my family, friends, supervisor, colleagues, and funder, the Harry Crossley Foundation. This thesis is the distillation of a great many number of ideas, experiments, and algorithms attempted over the last few years, with only a small subset of the good ideas making it into the final document. To persevere through the many failed ideas and explorations, as well as help critique and steer me towards better ideas, I am truly grateful to all those that have aided me.

I would like to thank Prof. Herman Kamper for being a truly amazing supervisor. On the technical side his ideas and insights, as well as the paper reviews and comments helped me develop both my writing and machine learning skills. He also was invaluable on the administrative side, always being on top of conference deadlines and organization, funding and event reminders, university administration paperwork, and generally being on top of this. Without these, this work would likely have taken much longer.

A large thank you is also due to Kevin Eloff, Benjamin van Niekerk, and several of members of the MediaLab for their friendship, support, and technical insight. In particular, Kevin Eloff for his amazing ability to get crazy ideas done fast, and to come up with great new ideas. Benjamin van Niekerk also never failed to provide valuable ideas, interpretation of results, and assistance with some of my research efforts. A great many of the other lab members helped in their own ways, but an additional special thanks to Werner van der Merwe, Leanne Nortje, and Christiaan Jacobs for their support.

I would also like to thank my funder, the Harry Crossley foundation, for funding my thesis. In particular, I would like to thank Rhodene Amos from the university administration team for always being prompt and on top of organization and communication between myself and the foundation.

A great many close friends, acquaintances, and even strangers online helped contribute to make this work possible. From ideas raised in github issues, to regular social gatherings, to general life support, a great many number of people have helped me bring this thesis to completion. I would like to extend a sincere thank you to all of them. And the gamers, for the memes (you know who you are), thank you.

Finally, I would like to thank my family for helping me through this chapter of my life. I am forever grateful to my parents, Ria and Andre Baas, and my brothers, Jordan and Kobus, as well as their partners for being an invaluable source of support for me throughout the years. Thank you as well to my extended family for your advice and assistance along the way, their help has and continues to be truly appreciated.

Abstract

A central goal in systems that produce speech is to easily control high-level characteristics of the speech while retaining naturalness. If we had such a system, it would enable a range of speech processing applications, from more realistic speech assistants, to assistive applications for those with speech disfluencies.

There is, however, a tension: most of the best existing speech processing methods rely on the explicit disentanglement of speech, but we know that humans process speech as a purely continuous signal without explicit factorizations. The explicit nature of the former means that they typically identify a handful of meaningful characteristics of speech beforehand and then carefully design systems to measure, disentangle, and modify these characteristics, recombining them to produce output speech. For example, speech synthesis systems might design modules to model speaker identity, prosody, and emotion separately from one another. This has a key limitation: explicitly identifying the discrete set of aspects that comprises speech is a contested and open-ended task, with some aspects intrinsically tied to one another, prohibiting explicit demarcation (e.g. phonemic identity is tied to timing in certain languages).

We observe the recent progress in other domains whereby injecting knowledge from domain experts when designing a model is inferior to more general methods that make fewer assumptions about the data and task. Instead, better performance is obtained when the disentanglement is learnt from the data rather than externally imposed. This leads to our primary aim: to investigate how we might bridge this tension between explicit and implicit disentanglement in speech processing. Our main claim is that continuous methods that implicitly learn the various aspects of speech can yield improved generalization compared to discrete methods with explicit demarcations of speech characteristics. However, we also observe that discrete methods offer easier training and scaling over purely continuous ones.

This thesis is divided into four parts. The first part investigates disentanglement in unconditional speech synthesis. We propose a new generative adversarial network (GAN) based approach for unconditional speech synthesis that generates speech purely from a continuous latent space without explicit conditioning. We introduce new techniques to optimize our model for learning a disentangled latent space whereby linear subspaces correspond to meaningful characteristics of speech. In experiments in a constrained setting of limited-vocabulary speech, we confirm that the learnt latent space is more disentangled than existing methods. And, critically, we demonstrate a key benefit of learning a disentangled, continuous latent space by showing that our GAN can perform several speech processing tasks unseen during training. Specifically, we show that – with simple linear operations in the latent space – we can perform voice conversion, speech enhancement, speaker verification, and keyword classification, in some cases to a similar level to task-specific baselines. This investigation shows that learning good continuous latent spaces enables generalization to unseen tasks.

The second part uses the insights of the first investigation to develop a model for a hard, practical task: any-to-any voice conversion. The result is k-nearest neighbors voice conversion (kNN-VC), a method which uses the linearly disentangled nature of features produced by existing speech representation models to perform voice conversion. As the name suggests, it is a method whereby provided speech is converted to sound like a desired target speaker by simply replacing each feature from the source with the mean of the k-nearest neighbors from the target. Despite its simple nature, compared to top-performing existing methods, kNN-VC achieves a new state-of-the-art for voice conversion. Not using discrete speaker labels enables the model to interpolate between voices, perform inference on unseen languages, and even be adapted to sample new speakers from a text prompt.

The third part of this thesis attempts to tackle the tension from the other side: can we give discrete methods similar benefits to continuous methods? We investigate this by attempting to incorporate disentangled continuous units for a task that necessitates discrete outputs: speech recognition. Specifically, we propose the first discrete diffusion model for speech recognition. Using the disentangled features from the prior investigation as conditioning, we iteratively refine a multinomial distribution over characters until we arrive at a final coherent transcript. We demonstrate comparable performance to existing state-of-the-art contrastive models on the LibriSpeech speech recognition benchmark. Compared to the dynamic programming algorithms necessary to decode from contrastive models, the output produced by our discrete diffusion model is readily interpretable. It also allows for the extensions afforded by various diffusion decoding techniques. This shows that adapting continuous domain methods (denoising diffusion) and disentangled continuous features for discrete domains yields certain benefits.

In the fourth part, we demonstrate the practical usefulness of the knowledge obtained from the first three parts by applying the lessons and methods to improve several existing speech processing tasks. Concretely, we demonstrate how voice conversion applied to unseen languages can be used to improve speech recognition in very low-resource settings. We also investigate how voice conversion can aid those with speech disfluencies by correcting stuttered speech, and test its generalization limits by investigating human-to-instrument conversions.

In summary, this thesis shows that the learnt disentanglements provided by continuous speech processing models allow for simpler generalization and control, but that discrete/explicit disentanglement methods still retain benefits in terms of ease of training and scalability. The question remains open as to what the best way is to combine the strengths of both approaches – or if it is even truly possible.

Contents

Nomenclature

Acronyms and abbreviations

ASR	automatic speech recognition
CPC	contrast predictive coding
CTC	connectionist temporal classification
DDPM	denoising diffusion probabilistic model
G2P	grapheme-to-phoneme
GMM	Gaussian mixture model
HGST	hierarchical global style token
HMM	hidden Markov model
IPA	International Phonetic Alphabet
kNN	k-nearest neighbors
LDA	linear discriminant analysis
LM	language model
MLM	masked language modelling
MLP	multi-layer perceptron
PDF	probability density function
SSL	self-supervised learning
TTS	text-to-speech
VC	voice conversion

Chapter 1

Introduction

The field of speech synthesis has a clear goal: to produce systems that can convincingly generate human speech. These systems have a long history, with early methods focusing on concatenative approaches for text-to-speech [1], then more recent approaches adopting autoregressive based approaches [2], and to the latest models – where diffusion-based and non-autoregressive approaches dominate various speech synthesis tasks [3–5].[1]

Even with these recent efforts to improve speech quality and diversity, there is still a disparity between speech and the two other main generative fields: text synthesis and image synthesis. This is particularly the case for generalization capabilities, where certain text language models like Mistral 7B [6] perform well on a variety of language processing tasks like named entity recognition and co-reference resolution, despite only being trained on a simple next-token-prediction task. Similarly for image synthesis, diffusion models like Imagen [7] can generate realistic images for a diverse spectrum of user descriptions and generalize to other image processing tasks like image in-painting, denoising, and compression [8].

But, with speech-producing models, we are only beginning to see such generalization capabilities. For example, we can ask a language model *"write a story about a sunny day at the beach with friends"*, or an image model for *"a photograph of a sunny day at the beach with friends"* and obtain consistent convincing outputs. But, in the speech domain, it is not clear how to generate a similarly convincing output for *"make the speech track for a sunny day at the beach with friends"*. While some hybrid methods exist (e.g. [9]), they rely on existing, pretrained, text language models internally to provide the long-form coherence, only relying on shallow text-to-speech (TTS) models to vocalize the transcript. New codec-based autoregressive methods [10] form a parallel effort to this thesis, and are also promising to bridge this generalization gap.

This thesis aims to investigate one cause of this shortfall: the tension between explicit vs implicit disentanglement of speech. Most top-performing speech models are designed to separate out a predetermined set of aspects of speech from one another (e.g. extracting linguistic content or speaker identity). However, human speech is processed as a pure continuous signal without such explicit logical separations. If we can better reconcile this tension, the design of speech models will better reflect the continuous nature of the signal being modelled. Ultimately, as we aim to show in the following chapters, designing speech models to implicitly learn the aspects that comprise speech (as opposed to explicitly specifying them beforehand) allows for better control and generalization of speech models.

[1]For this thesis, 'speech synthesis' is defined as any task involving the production of speech. Text-to-speech is treated as a subset of speech synthesis, where textual input is provided as conditioning.

1.1. Motivation

Intuitively, *disentanglement* refers to how the common factors of speech variation are represented inside a model (this is explained in detail in Chapter 2). E.g. to represent the differences between rhythm and speaker identity, one could construct separate architectures to independently model the characteristics and combine the result, as done in [11, 12]. Alternatively, one could attempt to learn the concepts of rhythm and speaker identity jointly in a single latent space, as done in [13]. The former corresponds to explicit disentanglement, whereby speech is explicitly factorized into a predetermined set of characteristics of interest. The latter, meanwhile, corresponds to implicit disentanglement, whereby the factorization of speech is implicit and learnt from the data, not guided by domain expert knowledge.

So, why focus on the tension between explicit and implicit disentanglement? Most current speech synthesis systems design components to disentangle explicitly identified aspects of speech (e.g. timing and linguistic content) – i.e. they model different aspects as perfectly distinguishable. But, we know this is not always true (e.g. timing and linguistic content in tonal languages). Also, other machine learning domains have observed that injecting knowledge from domain experts is inferior to more general methods that make fewer assumptions [14]. So, speech systems designed to learn strong implicit disentanglements of speech should yield improved control and generalization compared to explicit disentanglement methods that impose discrete distinctions on the characteristics of speech.

Further, if a model is to learn to modify and synthesize speech without speech-specific design assumptions, it should form an internal representation of speech that allows the user/model to easily – and ideally *linearly* – control different aspects of speech during inference. And, since we know certain aspects of speech are tied to one another as speech is a continuous signal, it suggests this internal representation should be *continuous*. In other words, a strong implicit disentanglement implies a latent space whereby certain linear subspaces capture the variations of certain characteristics of speech (explained further in Section 2.2). If we can design a model to learn or make use of such a space, then it is modelling speech with fewer domain-specific assumptions, which we posit leads to improved generalization capabilities.

1.2. Aims

This thesis aims to investigate and explore the benefits and drawbacks of designing speech processing models with a focus on using continuous, disentangled latent spaces. Concretely, the aim is:

Primary research aim

To investigate and develop methods that realize the benefits of implicitly disentangled continuous representations of speech over explicit factorizations of speech.

This broad goal can be roughly split into three separate efforts: (1) understanding existing methods for disentanglement, (2) attempting to learn more ideal continuous disentangled representations, and (3) attempting to apply these representations to various speech processing tasks to assess their practical benefit. Concretely, (2) is on the theoretical side, where we aim to investigate and develop more smooth and linear disentangled latent spaces without considering downstream applications. Meanwhile, (3) is on the application side, where we want to realize the hypothesized benefits of disentangled representations in practical applications. The first item we look at in Chapter 2, and we spread the second and third efforts across four investigations.

1.3. Investigations

This thesis is divided into four major investigations to try and probe the tension in our primary research aim from different angles. For each, we try to answer a central question which helps better understand and answer the main claim of this thesis – that continuous methods that implicitly learn the various aspects of speech can yield improved control compared to discrete methods with explicit factorizations of speech characteristics.

1.3.1. Unconditional synthesis

Research question 1

Why can we not simply apply disentanglement methods from the image domain to speech?

In the domain of image synthesis, we have seen a related version of our hypothesis validated: there are image synthesis models that make minimal assumptions about images, and learn a continuous disentangled latent space [15, 16]. The pioneering work in disentanglement for image synthesis focused on generative adversarial networks (GANs) [17] and the generalization benefits of such a latent space was illustrated by the StyleGAN family of models [15, 16, 18]. However, we have not seen similar such generation and performance capabilities in speech synthesis. Our first research question aims to investigate why.

We look at a task where the goal is to model the speech signal purely from a latent space – unconditional speech synthesis. This is the task of synthesizing coherent speech without any conditioning inputs such as text or speaker labels [19]. The current best-performing methods for this are diffusion models such as [20, 21]. However these do not have a strongly disentangled latent space, whereby linear translations in the latent space do not correspond to meaningful edits of the output speech. We want to increase the disentanglement to allow for the proposed generalization benefits.

So, we attempt to apply the disentanglement approaches from image synthesis to unconditional speech synthesis in Chapter 3. As we show, naively porting the methods from vision to speech does not work, and several modifications are required. This investigation culminates with a proposal of a new unconditional synthesis model – AudioStyleGAN. AudioStyleGAN maps a linearly disentangled latent vector to a coherent speech waveform. We perform an array of evaluations against existing models and demonstrate improved performance compared to baselines. Due to the challenging nature of direct synthesis from a latent space, existing models and our own are restricted to a limited vocabulary setting.

In the latter half of Chapter 3 we investigate the latent space and measure how linearly disentangled it is. After confirming substantial disentanglement, we go on to demonstrate how our model – because of its linearly disentangled latent space – can be applied to several speech synthesis tasks unseen during training. Concretely, by simple linear operations in its latent space, we can get the model to perform voice conversion, speech enhancement, speaker verification, and keyword spotting despite the model having only been trained to generate the raw unlabelled speech. We evaluate these zero-shot applications of our model against existing task-specific baselines to demonstrate reasonable performance. Relating this back to our aim, the results of Chapter 3 aim to show that disentangled latent spaces do provide generalization improvements, and give us some insight into *how to build better (i.e. more linearly) disentangled latent spaces* for speech processing.

1.3.2. Voice conversion

Research question 2

Given that there are models that have learnt to implicitly disentangle speech, why have we not seen speech synthesis models (voice conversion in particular) exploiting this to achieve the benefits posited in our motivation?

As we will discuss in Chapter 2, there are models which aim to learn good disentangled representations of speech – often known as self-supervised learning (SSL) speech models [13, 22, 23]. They do not synthesize speech, but rather typically use contrastive methods to convert a speech utterance into a sequence of continuous, implicitly disentangled vectors. These models are increasingly well studied, with benchmarks indicating strong linear disentanglement in the vectors produced by the latest models [24]. So, as our research question suggests – why are all the current best speech synthesis models not directly exploiting these features' strongly disentangled nature?

To investigate this, we look at the task of voice conversion in Chapter 4 – transforming speech spoken by a particular speaker into a target voice, keeping the content unchanged [25].

Concretely, to test whether our intuition behind disentanglement extends beyond the constrained setting of the first investigation, we look at a challenging variant: general purpose any-to-any voice conversion, where the source and target speaker are unrestricted and may be unseen during training.

As a first step, we use the intuition that – for latent spaces produced by certain SSL models – linear distances between disentangled vectors should correspond to the *linguistic differences* between those parts of the utterance. With this intuition, we attempt to achieve voice conversion with a simple k-nearest neighbors (kNN) algorithm, mapping each vector from the source representation to the Euclidean nearest vector from the target speaker. The output representation is then converted back to speech using an existing neural vocoder architecture. To our surprise, this simple nearest neighbor matching algorithm, which we dub k-nearest neighbors voice conversion (kNN-VC), achieves a new state-of-the-art in the challenging task of any-to-any voice conversion. In fact, we go on to show how kNN-VC – with all its components having only been designed or trained on English – can generalize to other languages and be adapted for other unseen speech synthesis tasks. This investigation is presented in Chapter 4. By pushing the limits of performance of voice conversion with a simple nearest neighbor algorithm, we showcase the evidence that *exploiting disentanglement does lead to better generalization.*

1.3.3. Speech recognition

Research question 3

Can methods that necessitate a discrete output benefit from exploiting continuous disentangled representations of speech?

For the third investigation, we look at the continuous/discrete tension from the other side by attempting to give discrete methods – where they are unavoidable – similar benefits to continuous methods. We do this by adapting two key continuous techniques to the domain of automatic speech recognition (ASR), where the transcription necessarily consists of discrete units (characters/words). Existing methods are typically based on contrastive learning tasks [26], and attempt to learn the transcript directly from the speech or spectrograms, not making use of the disentangled nature of SSL speech features, and requiring non-trivial decoding techniques to convert the model's predictions to the final transcript [26].

In Chapter 5 we attempt to propose an alternate approach to ASR by combining two key ideas from the continuous domain: strongly disentangled features produced by SSL models [23] and denoising diffusion probabilistic models (DDPMs) [27]. Specifically, we propose the first discrete diffusion model for speech recognition. Using the disentangled features from the prior investigation as conditioning, we iteratively refine a multinomial distribution over characters until we arrive at a final coherent transcript. To understand its performance characteristics versus traditional contrastive models, we run evaluations and ablations with existing state-of-the-art contrastive models on the LibriSpeech speech recognition benchmark.

The goal of this investigation is to show that by building a very different speech recognition model based on adapting continuous methods (diffusion) and disentangled latent spaces (SSL features), we can achieve comparable performance to existing models. And, compared to the fairly complex dynamic programming necessary to decode from contrastive models, the output produced by discrete diffusion models is readily interpretable and allows for several extensions in the decoding process that have been developed for DDPMs [28], which we explore in our evaluations. The investigation is presented in Chapter 5, where we attempt to answer the above research question in the affirmative, aiming to better understand *if and how continuous disentanglements can also benefit discrete-output models* with the concrete example of ASR.

1.3.4. Applied disentanglement: downstream applications

Research question 4

Given what we have found, can our focus on disentanglement translate into benefits in downstream speech processing tasks?

With the previous investigations, we aim to show that we can learn disentangled latent spaces (in a constrained setting), and then exploit large-scale disentangled latent spaces to improve two speech processing tasks (voice conversion and ASR). Now, with our motivation validated in these more well-explored settings, in Chapter 6 we set out to see if and how our motivation around 'designing-for-disentanglement' can yield improvements in less-explored, downstream applications. Concretely, we investigate how we can design a voice conversion system for improving ASR on very low-resource languages. The idea is to apply our voice conversion system to languages unseen during training to augment a small ASR training dataset. The voice conversion model we propose explores a different method of learning a disentangled latent space than those of SSL models used in kNN-VC. Specifically, it focuses on learning hierarchical style tokens [29] for one aspect of speech – speaker identity, since this is of central importance in voice conversion. We attempt to augment training data for four low-resource South African languages using this model. We then proceed to evaluate the improvements in ASR performance when trained on the additional augmented data. In addition to comparisons to existing augmentation methods, we ablate model design choices to understand if our disentanglement motivation around hierarchical style tokens is critical for cross-lingual performance.

Looking at even more underexplored tasks, we also investigate how generalizable the benefits are from the voice conversion model of the second investigation (Section 1.3.2). The main motivation behind the simple kNN procedure and usage of SSL features was that their disentangled nature should enable greater generalization. So, to test this, we attempt to apply kNN-VC to two very out-of-domain tasks: assisting those with speech disfluencies by correcting stuttered speech, and attempting to perform musical instrument-to-instrument conversions. We formulate the former as voice converting from a speaker without disfluencies to a target speaker with disfluencies, thereby retaining the user's voice but removing the disfluencies. Similarly, we formulate the latter as voice conversion with the source and/or target utterance replaced by a recording of a musical instrument (i.e. not human speech). We also show how kNN-VC can be slightly adapted at inference time for prompt-to-voice, whereby we convert a textual description of a speaker to a target speaker used in the conversion process. By exploring these tasks in Chapter 6, we aim to better qualify the extent to which our motivation of Section 1.1 is ultimately beneficial in a range of downstream applications.

1.4. Contributions

The contributions of this thesis can be broadly categorized into three parts: (1) research contributions around using and learning disentangled, continuous latent spaces for speech processing tasks, (2) code repositories and guides on how to reproduce, use, and build upon the research contributions, and (3) publications.

1.4.1. Research contributions

We highlight the following main contributions:

- The results of the investigation into Research Question 1 yielded a new top-performing model on the Google Speech Commands dataset in terms of latent space disentanglement and synthesis diversity. Unlike previous methods, which focus purely on unconditional synthesis [19] or don't have a disentangled latent space [21], our proposed AudioStyleGAN does have a disentangled latent space and so can be applied directly to unseen tasks.

- The second investigation on voice conversion yielded a new state-of-the-art any-to-any voice conversion model, kNN-VC. Top-performing existing methods are increasingly complex [30–32] and are challenging to build upon and compare to. Yet, kNN-VC outperforms these existing models with a simple nearest neighbor matching algorithm built on existing SSL features. Because its operation is so simple, it makes minimal assumptions about the input speech, allowing for greater generalization for cross-lingual conversion, stutter correction, and other out-of-domain tasks.

- The third investigation on applying continuous methods to speech recognition yielded, to the best of our knowledge, the first formulation of speech recognition as a conditional diffusion task. And so, our proposed model is the first multinomial DDPM for speech recognition and achieves comparable performance to similarly sized contrastive models. We also propose several new DDPM decoding techniques specific for speech recognition to boost performance.
- We propose a new voice conversion model for data augmentation on low-resource languages, whereby it can increase the diversity of a small pool of training data to increase downstream ASR model performance. We show how it can be used in combination with existing augmentation approaches, and profile the settings where the augmentation is beneficial.
- We develop and profile the downstream performance of kNN-VC. Specifically, we propose an extension to kNN-VC to allow it to be used to generate textually described voices, where the speaker identity is specified by a textual description. There has been relatively little work into textually described voices [33], with minimal well-performing publicly disclosed methods available. We also apply kNN-VC to stuttered speech correction and show how voice conversion can be used to correct recordings with speech disfluencies. Even further out-of-domain, we demonstrate the generalization of kNN-VC by applying it to non-speech conversion, showing how it can be used to convert between instruments and human/animal conversions with varying levels of performance.

1.4.2. Code repositories and guides

An auxiliary goal of this thesis is to ensure it is easy and simple to reproduce and build upon our work. To this end, several of our contributions are in the form of public code, model checkpoints, and other guides. We highlight the following external resources produced in this thesis:

- `https://github.com/RF5/simple-asgan` : Audio demos, code, and trained models for AudioStyleGAN, with extra information about some of the experiments performed.
- `https://bshall.github.io/knn-vc` : Audio demos, code, and trained checkpoints for the kNN-VC developed while answering Research Question 2.
- `https://github.com/RF5/transfusion-asr` : Audio demos, code, and trained checkpoints for the multinomial diffusion model introduced for Research Question 3.
- `https://rf5.github.io/interspeech2022` : Audio demos and ablation samples for the hierarchical style token voice conversion model proposed while answering Research Question 4.

- https://rf5.github.io/sacair2023-knnvc-demo : Audio demos and trained models for the kNN-VC extensions to stutter correction, cross-lingual conversion, instrument conversion, and textually described voice conversion.
- https://github.com/RF5/simple-speaker-embedding : A robust reimplementation of the generalized end-to-end loss for speaker verification [34], often abbreviated 'GE2E loss'. Many of the evaluations and projects in this thesis required a robust method to extract speaker embeddings, and existing systems were too complex or not robust enough. So, we provide performant and robust speaker embedding models based on the GE2E loss, as well as benchmarks on standard datasets.
- https://gist.github.com/RF5/eabb93ba85b763746d404afc9626e5d1 : A guide for the Stellenbosch GPU HPU cluster for students using it for deep learning. Many new electronic engineering skripsie/masters students consult this guide when using the Stellenbosch HPC GPUs.
- Two re-implementations of existing papers which we use as baseline methods to compare to in some of our investigations. Hoping others may be able to use our re-implementations for their future comparisons, we provide code and checkpoints for AutoVC (https://github.com/RF5/simple-autovc) and several classifiers for the Google Speech Commands dataset (https://github.com/RF5/simple-speech-commands).

1.4.3. Publications

Large parts of this thesis correspond to published journal and conference articles, listed below:

Publications

Conference papers:

Paper 1: M. Baas, H. Kamper, "Voice Conversion Can Improve ASR in Very Low-Resource Settings," in *Proceedings of Interspeech*, 2022.

Paper 2: M. Baas, H. Kamper, "GAN You Hear Me? Reclaiming Unconditional Speech Synthesis from Diffusion Models," in *IEEE Spoken Language Technology Workshop*, 2022.

Paper 3: M. Baas, B. van Niekerk, H. Kamper, "Voice Conversion With Just Nearest Neighbors," in *Proceedings of Interspeech*, 2023.

Journal papers:

Paper 1: M. Baas, K. Eloff, H. Kamper, "TransFusion: Transcribing Speech with Multinomial Diffusion," *Springer Communications in Computer and Information Science*, vol. 1734, pp. 231–245, 2022.

Paper 2: M. Baas, H. Kamper, "Voice Conversion for Stuttered Speech, Instruments, Unseen Languages and Textually Described Voices," *Springer Communications in Computer and Information Science*, vol. 1976, pp. 136-150, 2023.

Paper 3: M. Baas, H. Kamper, "Disentanglement in a GAN for Unconditional Speech Synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 1324-1335, 2024.

Non-first-author contributions:

Paper 1: B. van Niekerk, L. Nortje, M. Baas, and H. Kamper, "Analyzing Speaker Information in Self-Supervised Models to Improve Zero-Resource Speech Processing," in *Proceedings of Interspeech*, 2021.

Paper 2: B. van Niekerk, M.A. Carbonneau, J. Zaïdi, M. Baas, H. Seuté, H. Kamper, "A Comparison of Discrete and Soft Speech Units for Improved Voice Conversion," in *Proceedings of IEEE ICASSP*, 2022.

Chapter 2

Background

This chapter aims to give the reader an understanding of three things:

1. The nomenclature and definitions we use in this thesis.
2. The facets of the discrete/continuous tension and its relation to implicit and explicit disentanglement.
3. The existing techniques and efforts in speech modelling, with a focus on their relationship to the concept of disentanglement.

Up till now, we have been somewhat vague about disentanglement and how it relates to discrete vs continuous approaches to speech processing, having only briefly discussed the connection in Section 1.1. This chapter aims to address this by giving more concrete definitions and relations between these concepts. We do this by splitting up the tension between how humans produce speech and how machines produce speech into two: the difference between explicit/implicit factorizations of speech, and the distribution representation problem. The former focuses on the difference between learning what aspects comprise speech and using existing domain expert knowledge when designing speech synthesis models. Meanwhile, the latter focuses on the difficulties of representing diverse distributions of high-dimensional data.

The second half of this chapter looks at some of the key existing techniques to both synthesize speech and learn/utilize disentanglements – both of which will form the starting point for our investigations into each of the research questions posed in Section 1.3. By the end of this chapter, the reader should feel confident in the concepts used throughout this thesis and roughly know the laylines connecting speech synthesis, disentanglement, and the other fields our work draws upon. This background chapter assumes the reader is already familiar with the basics of machine learning, probability theory, and artificial neural networks.

2.1. Terminology and definitions

Within the field of speech synthesis, there is much variation in terminology, notations, and even definitions. For example, a symbol X_t can refer to a matrix, a vector, a scalar, or a random process, depending on which author is consulted. This is even the case in very specific sub-fields like TTS using DDPMs, where the above example is taken from. So, in the following, we try and clarify some key terms as we use them in this thesis. We acknowledge that due to the conflicting conventions in existing works we build upon in the different chapters, we may not be globally consistent across all chapters. However, within each investigation we should remain highly consistent.

2.1.1. Terminology

To contextualize the techniques used for speech synthesis, we begin with a description of the primary concepts referenced throughout this proposal. Note that the term 'disentanglement' has a more complex history and is defined separately in Section 2.1.3.

- **Speech synthesis**: Speech synthesis is the task of generating realistic, intelligible synthetic human speech. While humans can synthesize speech, typically when referenced in the engineering domain it refers to machines that can create sounds that resemble human speech. The field has a long history, with several systems being made before electronic signal processing [35] (discussed in the next section).
- **Utterance**: a sequence of audio waveform samples. Typically it is a short recording (1-30 seconds) of a person speaking intelligible human speech into a microphone. Utterances are often encoded in digital format (e.g. in a `.wav` file) and amassed into large collections comprising a speech dataset. Depending on the context, an 'utterance' may either be referring to the sequence of discrete-time samples comprising the digital waveform, the original continuous signal, or a derived representation of the original signal (e.g. a spectrogram).
- **Speech characteristic**: a single aspect of speech that humans can identify well. Typical examples include speech content (the transcript of the words spoken), speaker identity, tone, timbre, volume, word timing, and noise levels. Since speech is ultimately a continuous signal produced in the real world, these concepts are fuzzy and do not have fine demarcations except where introduced by ourselves to simplify analyses and model design. Even certain components of language that are definitionally discrete (such as the transcript of an utterance) are still fuzzy when reconstructing from a continuous speech signal – with the famous example of the disagreement of whether an utterance contains the word "yanny" or "laurel".[1]
- **Conditional vs unconditional speech synthesis**: conditional speech synthesis refers to speech synthesis which uses some user-provided information to *condition* its output, whereas unconditional speech synthesis refers to speech synthesis which uses no user-provided information to influence its output. That is, its output is

[1]See reporting on the phenomenon in 2018 at this link.

essentially arbitrary speech that is uncontrolled. Common examples of conditional speech synthesis include TTS systems (conditioned on text) and voice conversion (VC) systems (conditioned on speaker identity and source utterance). The term typically refers to the way the models are *trained*, and not how they are used at inference time, where unconditional models are rephrased in special ways to allow conditioning input (as otherwise they would be largely useless for controllable speech synthesis).

- **Speech diversity** refers to how varied the speech is in terms of all common speech characteristics. For example, how varied is the text transcript, or how many speakers are there among the utterances under consideration?

- **Speech quality** refers to how realistic and intelligible the speech sounds. Measures for speech quality typically revolve around how well a human can distinguish a synthesized utterance from a real utterance produced by a human.

- **Latent space** refers to a vector space wherein there exists some manifold (akin to a high dimensional surface) where different regions on the manifold correspond to different signal properties for some relationship between high dimensional signals and points on this manifold. Common examples include word embeddings, where (e.g. through backpropagation) we learn a mapping between words and points in a vector space. And, if the model is performing well, ideally semantically similar words (e.g. 'truck' and 'car') should reside near each other in the vector space (i.e. lie close to one another on the manifold). We are concerned with those manifolds which are highly disentangled, ideally ones that are comprised of linear sub-spaces for common characteristics of speech variation.

2.1.2. Notation conventions

We use the following notation conventions in this thesis:

- Lower case, non-bold variables are *scalars.* E.g. α, d, m.

- Lower case, bold variables are *vectors.* E.g. $\mathbf{s}, \mathbf{x_t}$.

- Upper case, non-bold variables with subscripts are typically *matrices.* E.g. X_t, Q_i.

- Upper case, non-bold variables without subscripts are typically *scalars*, often indicating counts or sequence lengths. E.g. T, N.

- Probability density and mass functions are typically indicated with $p(\cdot)$ or $f(\cdot)$. Random processes are likewise indicated but with an subscript, e.g. $p_t(\cdot)$.

- Neural networks / non-linear functions are typically indicated as capitalized letter functions. E.g. $G(\cdot)$, $E(\cdot)$.

2.1.3. Disentanglement

With the previous terms and conventions now defined, we move on to the key term of this thesis: disentanglement. The idea of speech disentanglement has a long history, with initial arguments about whether a separation between *speech* and *language* is even possible playing out between Chomsky and other linguistics researchers [36]. More formalized arguments like those proposed in [36] begin to draw more concrete ideas of speech disentanglement, where they roughly reason that (a) speech can be separated into an "external representation" (the utterance) and the "radically different internal object of representation" (the mental representation of the utterance before it is spoken). And, (b) they reason that this internal mental representation of speech must have a phonology with a "mentally constructed grammar". The idea that a mental representation of speech must have a grammar – i.e. a structure, must be rephrased for digital speech systems which do not have a mental state. One possible place in digital systems where such structure can arise is in a *latent space*. Specifically, we can say that for a speech synthesis system, the latent space which represents speech must have a structure, or manifold akin to a grammar for the associated speech it can produce. The definition of disentanglement then arises from the idea that we can simplify this structure to make it easier to analyze and control the speech being produced. Concretely, we define disentanglement similar to other contemporary work in the image synthesis domain [15] for speech:

Definition: Disentanglement

Disentanglement is the separation and simplification of a speech synthesis model's internal representation (i.e. latent space) such that common factors of speech variation correspond to low-dimensional manifolds (ideally linear subspaces) within its latent space.

This definition is closely related to that proposed by Liu et. al. [37], where they also reason that for a model to be good at predicting speech, two factors of speech variation (phonetic and speaker information) present in the internal representation should not only reside in separate linear subspaces, but *orthogonal* subspaces. While they find strong evidence for this in SSL speech models, we opt for the weaker requirement of linear – but not necessarily orthogonal – subspaces to make it hold for more factors of speech variation that are more closely tied together (e.g. speaker identity and gender).

Using our definition above allows us to directly claim that improved disentanglement allows for more controllable speech synthesis. This is because, if the structure of the latent space is simple, to control factors such as speaker identity will involve simple (and ideally linear) operations on a point in the model's latent space. With the recent explosion in image synthesis models, much work on objectively measuring disentanglement of latent spaces/model representations comes from the image synthesis domain [15, 16, 18], with some related work in the speech SSL benchmarking domain [24].

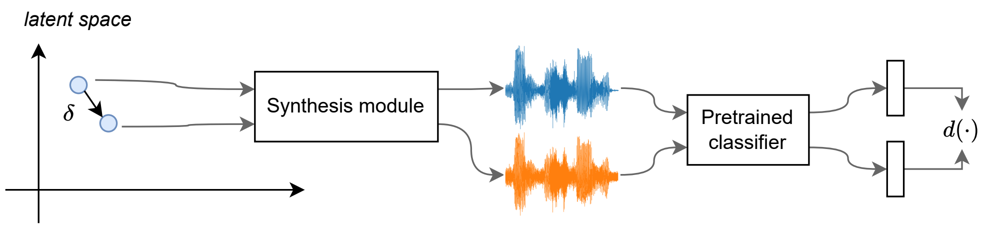


Figure 2.1: Diagram indicating how the perceptual path length is computed for measuring disentanglement of a latent space. An utterance (as a point in the latent space) is perturbed by adding a small noise vector $\boldsymbol{\delta}$, producing a second latent point. Both are synthesized to audio with the model under evaluation, and then passed through a speech classifier model. A path length estimate is formed as the cosine distance $d(\cdot)$ between the intermediate activations of the classifier model.

Broadly, these measures attempt to quantify how 'close' to a linear subspace the internal representation is, relative to some set of speech factors. For example, the SUPERB benchmark [24] for SSL models quantifies disentanglement by the performance of *linear* models on predicting various characteristics of speech (e.g. phonemes, speaker, emotion). These linear models are trained on the latent representation of utterances from an enrollment/training set. Now, because the models are only linear, for their performance to be better over all the tasks in the SUPERB benchmark, the latent space must be more linearly disentangled.

Meanwhile, the image synthesis literature [15, 16] focuses more on the synthesis aspect, whereby they measure disentanglement as the perceptual path length in the internal activations of an existing classifier model. The setup for computing the perceptual path length is shown in Figure 2.1. Intuitively, the disentanglement of the latent space is measured as how perceptually 'smooth' small movements in the latent space are – the more disentangled, the more a small perturbation in the latent space should not significantly change the perceived qualities of the produced speech. To obtain a numerical approximation for this, a pretrained classifier is used (image classifier for image synthesis, speech classifiers for speech synthesis). The process for computing a single 'path length' value is shown in Figure 2.1. The final perceptual path length estimate reported in model comparisons is then the average 'path length' computed over typically 100 000 (test utterance, perturbation) pairs. Sometimes, as in [15], the starting point in the latent space is not set as the latent point of a test utterance, but rather as a random point along the straight line joining the latent points of two test utterances.

Finally, it is worth noting that disentanglement does not imply synthesis. That is, a speech representation model (e.g. HuBERT, Wav2vec [13, 22]) can achieve disentanglement without being able to synthesize speech. Specifically, to learn a disentangled latent space, a model needs to learn the relationship between utterances and its internal representation. This can be done by either modelling a mapping *from* speech *to* the latent space (speech representation systems, e.g. [22]), *from* the latent space *to* speech (speech synthesis systems, e.g. [3]), or both (e.g. autoencoder systems like [38]). So for a model with a latent space to perform speech synthesis, we must have some way to map points from its latent space back to time-domain waveforms that resemble speech. This is an important requirement for using disentanglement for controllable speech synthesis.

Now that we have an idea of what speech disentanglement is and how it is measured, in the next section we will look at how it relates to the tension between discrete and continuous approaches to modelling speech.

2.2. The continuous-discrete tension

As mentioned in Chapter 1, there is a divide between the continuous way humans use speech and the current best-performing speech synthesis methods which rely on discrete approaches to speech. So, key questions of our thesis motivation are "why can't we just use purely continuous methods to model speech? Why are discrete approaches dominant?" We highlight two key tensions that help give some insight to these questions and set up the challenges that we attempt to overcome in our investigations in the upcoming chapters (particularly the second tension).

2.2.1. Tension 1: representations for high dimensional distributions

The problem formulation is this: we want to train a model that learns to synthesize some high-dimensional data. Statistically, we want to tune the model parameters θ to maximize the likelihood of the training data given the model, i.e. $\max p_\theta(\texttt{data})$. The question arises: how can we represent a probability distribution over high dimensional data? There are two main approaches to this:

Quantization

We can quantize the high dimensional space into a tractable, discrete set of options. Then, the model can predict the probabilities of a categorical distribution over these discrete options. This is the *discrete approach*, used most frequently in SSL training (e.g. Wav2vec2 [22], HuBERT [13]), generative spoken language models [39], and other speech synthesis models utilizing discrete units [40]. Despite the quantization errors this introduces, these methods are currently the most successful for many speech synthesis tasks, particularly unconditional speech synthesis [41]. This is primarily because a categorical distribution is extremely flexible, and can represent very complex discrete distributions while remaining easy to learn and store, even for very high dimensional data.

Approximating with a continuous prior

If we wish to stay in the continuous domain, we need a way to make the $p(\texttt{data})$ tractable when the data can be a high dimensional vector, or even sequence of vectors, such as when iteratively predicting a mel-spectrogram. Unlike the quantization above where the categorical distribution probabilities can encode an arbitrary distribution over the discrete data, in the continuous domain it is not clear how we can construct an arbitrarily flexible distribution over all real numbers. So, most existing methods (in image or speech synthesis) impose a prior on either a latent space or the final prediction space. The most simple and common one is a Gaussian prior, whereby model designers will assume that points in a latent space conform to a (typically isotropic) Gaussian distribution or a Gaussian mixture. Then, the model will be trained to either map samples from a Gaussian mixture to the data (such as in StyleGAN models [15]), or to predict the parameters of a Gaussian mixture to sample from (such as in voice conversion attempts [42, 43]). The former imposes a prior on a latent space and trains the model to learn the transformation of the prior distribution to approximate a distribution over the vector space spanned by the data, while the latter imposes a prior on the output space and trains the model to learn the parameters of the distribution.

While approximating continuous spaces with a prior allows us to train and scale models in the continuous domain, the prior distribution imposed is much less flexible than the categorical distribution for discrete modelling. A large reason for this is the simplifying assumptions introduced to the priors to make the problem more tractable, such as only using a small number of Gaussians in a mixture model, or – most commonly – restricting Gaussians or other priors to be isotropic. That is, for each dimension of the high-dimensional data (or each item in a sequence) to be sampled *independently*. Here is where the problem arises: if each dimension/sequence index of the output is being modelled effectively independently, the model is learning to predict all the *marginal distributions*, and not the full joint distribution. The problems these simplifying assumptions introduce can be illustrated by a few examples:

1. If the data follows a non-isotropic Gaussian distribution (i.e. is governed by a covariance matrix with non-zero off-diagonal elements), then modelling it with any isotropic distribution prevents a model from being able to express the dependency between dimensions during sampling.

2. If the data being modelled can be formulated as a non-stationary random process – such as a sequence of spectrogram frames – then designing a model to predict each of the frames autoregressively in a purely continuous way will accumulate error. Whether the model is trained through a deterministic point-loss (e.g. L_2-loss with the next spectrogram frame) or by maximizing the likelihood of a Gaussian (e.g. predicting $\boldsymbol{\mu}, \Sigma$ for the following spectrogram frame), during inference, at each

sequence the quality of the signal will be further degraded, as the generated sequence diverges from the true data distribution as error accumulates.

3. If the data has a one-to-many relationship, and we train the model as a continuous regression problem, this is equivalent to modelling the data as a Gaussian distribution, even when a Gaussian may not be appropriate [44, Chapter 5.5]. For example in TTS (a one-to-many problem, where the transcript can have multiple spoken realizations), if we predict the target spectrogram and train with a direct L_2 regression loss with the target spectrogram frames, this is equivalent to phrasing the model as predicting a distribution [44] $p(\mathbf{y}|\texttt{text}) = \mathcal{N}(\mathbf{y}; \text{Model}(\texttt{text}), \Sigma)$, where $\mathbf{y}$ is the target spectrogram frame, and $\text{Model}(\texttt{text})$ is the predicted spectrogram frame. But spectrogram frames do – in general – not follow a Gaussian distribution, and training TTS models with regression leads to an “over-smoothed” prediction (blurry spectrogram) [45].

For the cases where continuous models still prevail (such as in image generation or speech enhancement), there are special techniques used to circumvent these shortcomings. For example, the DDPMs used extensively in speech enhancement try to overcome these restrictions by performing repeated inference calls on the input, slightly denoising it each time in such a way that – while each individual inference may be limited by the above – the entire reverse Markov chain of inference can model much more arbitrary complex distributions (DDPMs are explained in Section 2.6).

Another example is GANs, whereby they try to overcome the requirements by attempting to not optimize the data using an imposed prior, but rather attempt to ‘learn’ the prior of the data on the fly with a discriminator, which – by virtue of being a large neural network – can implicitly encode more complex distributions (more on GANs in Section 2.3.2). In summary: formulating a way to optimize the likelihood of high dimensional continuous data is hard, while optimizing arbitrarily complex distributions of discrete data is comparatively easy.

2.2.2. Tension 2: explicit versus implicit factorization of speech

The first tension above looks at the challenges of representing continuous signals as inputs or outputs from a model. The second tension focuses on the model itself, specifically how the model design incorporates assumptions about the nature of the underlying signal (speech or some feature thereof). This is the tension of implicit versus explicit factorization of speech: our decisions about what information pathways are setup inside an architecture assumes that speech can be fully modelled with those pathways, which is not always true. For example, even high-performing recent speech synthesis models such as SpeechSplit [11] models speech under the assumption that the linguistic content of speech is separate from the pitch content of speech. However, for tonal languages like Mandarin Chinese, the pitch contour used over words changes the linguistic meaning.

This tension is separate from the continuous/discrete tension. We can see this clearly if we compare SpeechSplit [11] (a largely continuous model, aside from speaker identity) to the discrete speech synthesis model [46]. The latter also attempts to decompose speech into a couple of distinct factors, but models speech internally using discrete sequences instead of continuous methods. Despite this, it still falls foul of the same issue as above – it models pitch in an entirely separate information stream than the linguistic content. Both models introduce an *explicit* factorization of speech, modelling the pitch as separate from the linguistic content – which we know for tonal languages is not fully accurate.

Meanwhile, if we look at image synthesis domain, methods like [15] model images without assuming a particular factorization of the aspects of images. By training a model without an explicit factorization, the model is able to *learn* a factorization, which can be more apt than an imposed factorization that we apply using our domain knowledge of the signal. In other words, by allowing the model to learn an appropriate disentanglement of the aspects of speech, it *implicitly* factorizes speech. And other work in the image and text domain has also provided evidence that learning the structure of data is often better than using existing domain knowledge to explicitly specify the structure of the signal being modelled [14].

Regardless of whether the factorization of speech is manually specified or learned, the structure of the model imposes assumptions and limits on how it can learn the separation between different characteristics of speech – i.e. the disentanglement of speech. So, we introduce two terms to tie the concept of speech factorization to disentanglement:

Definition: explicit disentanglement

Explicit disentanglement refers to disentanglement that is achieved when one explicitly identifies the set of meaningful characteristics that comprise speech and how these characteristics are delineated from one another in a model architecture.

Definition: implicit disentanglement

Implicit disentanglement refers to disentanglement that is achieved without explicitly identifying characteristics that comprise speech and how they are delineated from one another in a model architecture.

With these terms, we can state the second tension concretely: many tasks in speech synthesis are dominated by models using domain knowledge to *explicitly* disentangle speech, but other domains have shown that models which *implicitly* learn a disentanglement (by making fewer domain assumptions about the data) generalize and scale better [14]. So, why do we not yet see a similar trend in speech synthesis?

We try to probe the reasons and possible solutions for both these tensions in the upcoming chapters. With the definitions, problem, and idea of disentanglement now clear, the remainder of this chapter gives an introduction to existing methods in speech modelling and synthesis.

2.3. Speech modelling techniques

We will start with a description of classical methods, and thereafter we will give a summary of the more recent techniques at the forefront of speech synthesis and representation. This section will assume the reader's familiarity with basic deep learning concepts such as backpropagation, optimization, common neural network layers, and the notion of model training and loss functions.

2.3.1. Classical methods

Before the recent advances in deep learning, there were several signal processing techniques which attempted to disentangle and synthesize speech. Many early methods focused on disentangling background noise from human speech due to the more predictable (i.e. statistically ergodic) properties of many forms of noise, ultimately in an attempt to remove the noise from a speech signal. In 1990, [47] attempted to use two parallel hidden Markov models (HMM) to concurrently model the noise and speech signal in an utterance in an attempt to decompose (i.e. disentangle) the signal into speech and noise. This work falls under attempts at explicit disentanglement (defined in Section 2.4) as it explicitly identifies and demarcates general speech into two distinct factors: intelligible speech signal and noise. Even earlier in the 1980s, works such as [48] designed two filters to try and separate speech into pitch (or long-term frequency envelope) and content (short-term frequency envelope associated with the formants of an utterance). Again this technique falls under explicit disentanglement, as they have made a hard distinction in filter design between pitch and content.

Then in the late 1990s, works such as [49] started to move to slightly more general systems which made less stringent assumptions on speech by modelling speech using a filter designed to mimic the human vocal tract. Like in previous works, they attempt to analyze the frequency spectrum of speech to disentangle speech into two components: a periodic component and an aperiodic component. They learn this decomposition using a specialized iterative filter which tries to approximate the waveform using a random (aperiodic) component and a deterministic (periodic) component. Again the explicit demarcation is clear – speech is discretely decomposed into two expert-defined classes of signals, with systems attempting to learn how to disentangle speech into these defined factors.

In the early 2000s and 2010s, the TTS field began to focus on more concatenative and unit-selection methods [1]. The idea behind concatenative synthesis and unit selection is simple: if we have several exemplars of clean parts of speech (and often a label for each speech segment, such as the phoneme spoken in that segment), the methods of concatenative synthesis attempt to ask "how can we concatenate different exemplars together to synthesize the desired speech, such as those corresponding to a text input?" In more sophisticated TTS models, for example, a hidden Markov model would be used to select which next exemplar to concatenate to produce the appropriate next sound [1, Chapter 15 & 16]. These methods still required a substantial amount of hand-tuning and expert labelling of data (e.g. phonetic stress, duration) to find good methods of deciding which next speech unit is optimal in the concatenation process.

While some of these concatenative methods could be considered as attempting to learn an implicit disentanglement, and even using unit selection with pure Euclidean distances between speech features, they soon fell out of favor. That is, despite their innovative nature at the time, these HMM or unit-selection methods were vastly poorer in synthesis quality compared to more modern, deep-learning-based models [50]. In Chapter 4, we revisit concatenative synthesis to see whether incorporating modern disentangled SSL features can provide better disentangled features for unit selection. As we proceed to look at more modern techniques, we encourage the reader to keep an eye out for the trend – models that make fewer domain-expert assumptions about speech but can scale to larger model sizes and datasets nearly always lead to significant improvements.

2.3.2. Generative adversarial networks

One of the earlier breakthroughs in deep learning for continuous signals came with the introduction of generative adversarial networks (GANs) – a class of neural network training configuration introduced by Ian Goodfellow in 2014. Concretely, a GAN comprises of a *generator* which in some way synthesizes new data from a desired dataset or distribution, and a *discriminator*, which attempts to discriminate real data samples from those generated by the generator [17]. While the original GAN only had a single network for the generator and discriminator with a single loss term, there have been many extensions and progress in training GANs since 2014 [51, 52], with some works now opting to use multiple generators and discriminators [53].

The key part of the original GAN setup is the idea of the adversarial non-saturating logistic loss [17] – the loss function through which GANs are trained. Specifically for a generator G, discriminator D, and dataset item X (e.g. mel-spectrogram), the generator should attempt to map a latent variable $\mathbf{z}$ to an output $G(\mathbf{z})$ that the discriminator cannot distinguish from X drawn from the dataset. The discriminator D maps a data item to a scalar $D(X)$, and is trained to raise its output for X sampled from real data, while minimizing its output for data sampled from the generator $D(G(\mathbf{z}))$. The generator, meanwhile, is trained to do the opposite. Concretely this is formalized in a loss when updating the generator:

$$\mathcal{L}_G = -\log(D(G(\mathbf{z})) \tag{2.1}$$

Where the generator does well (loss is low) when it fools the discriminator into producing a high score (near 1) for the synthetic output. Meanwhile, the discriminator is trained to do the opposite:

$$\mathcal{L}_D = -\left[\log(D(X)) + \log(1 - D(G(\mathbf{z}))\right] \tag{2.2}$$

The generator and discriminator are typically iteratively updated using gradients derived from these loss functions. In light of Tension 1 (Section 2.2.1), GANs avoid imposing a restrictive prior on the data by attempting to learn it jointly with the generative model. That is, the discriminator implicitly encodes a complex distribution by learning how to distinguish high-dimensional data as being drawn from the real data distribution, or a fake/imposter distribution (produced by the generator). While this avoids the simplified prior issue, it does so at a cost: training of GANs is notoriously unstable, and often struggles to converge without additional tricks and loss terms to stabilize training [52].

For example, a typical problem is that the discriminator or generator learns much faster than its counterpart (e.g. the discriminator always correctly classifies generated data as fake), which in turn provides a very poor learning signal (read: numeric gradient) for the counterpart to continue improving. This often requires several patch-in techniques such as [54, 55] to circumvent. Ultimately, while GANs are powerful, as they are presently formulated, they do not resolve our one continuous/discrete tension (Section 2.2.1) out-of-the-box, but simply trade out one problem (introduction of oversimplified priors) for another (optimization instability). In our attempt to improve GANs for unconditional speech synthesis in Chapter 3, we introduce our own model designs where we tweak this regular GAN setup and change the way we schedule updates to G and D in an attempt to overcome these optimization challenges.

2.3.3. Masked language modelling

Masked language modelling (MLM) is a loss formulation used to train state-of-the-art speech representation models [13,23]. Speech representation models are those models which employ self-supervised learning to produce high-level representations of speech, typically as a sequence of vectors, with each vector representing 10-20 ms of an utterance [24]. As a reminder, speech representation models do not perform speech synthesis, but only attempt to learn good (i.e. disentangled) representations of speech as a sequence of vectors useful in downstream tasks such as intent classification or ASR [24]. However – as we will show later in Chapter 4 – special models can be trained to reconstruct audio from these representations (i.e. to *vocode* such representations), thereby allowing them to be repurposed to aid in speech synthesis.

Concretely, MLM is a discriminative task whereby a model G generates a sequence of output vectors $\tilde{X} = [\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_{T-1}]$ from a sequence of input vectors $X = [\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{T-1}]$ – i.e. $\tilde{X} = G(X)$. These output vectors are projected to form a sequence of probability mass functions over a discrete set of classes[2]. This results in a sequence of probability vectors $[\tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1, ..., \tilde{\mathbf{k}}_{T-1}]$.

This sequence of probability vectors is then trained to maximize the likelihood of a sequence of observed classes. The target classes of this sequence are derived in some way from the input X such that each element of the input sequence has a single class associated with it, yielding a sequence of target classes $[k_0, k_1, ..., k_{T-1}]$. For example, for training HuBERT (an SSL speech representation model), if the inputs X are spectrogram frames, then the classes are the cluster indices for a K-means clustering algorithm trained on spectrogram frames [13]. To make the task more challenging, random indices of the input sequence are masked before being fed to the model. A self-supervised loss is then formed as the maximum likelihood objective of the target classes for the masked timesteps. Intuitively, this forces the model to predict the masked portions of the sequence. This is computed as the mean cross entropy (CE) between $\tilde{\mathbf{k}}_t$ and target class k_t over all such masked indices t. Schematically this setup is shown in Figure 2.2.

2.3.4. Autoregressive modelling

The third major method used in training speech synthesis models is autoregressive modelling, whereby a model iteratively predicts the next discrete token in a sequence. The setup is restricted in a similar way to MLM, where we have a sequence X that we wish to model, so that ultimately we can sample new plausible sequences or produce disentangled representations associated with X.

Unlike MLM, however, autoregressive models are *causal* and the values of the sequence are often discrete (i.e. $\mathbf{x}_i$ is a one-hot vector among a fixed number of classes). An autoregressive model iteratively predicts $\mathbf{x}_t$ given all $\mathbf{x}_0, ..., \mathbf{x}_{t-1}$ as conditioning. This is done by predicting, at a particular sequence index $t-1$, a probability distribution over the *next* token, i.e. $\tilde{\mathbf{k}}_t$. The model is trained to maximize the likelihood of the actual next token class $\mathbf{x}_t$.

The quintessential example of autoregressive modelling in the speech synthesis domain is that of Tacotron [56] and Tacotron2 [2], whereby a recurrent neural network iteratively predicts the next frame in a spectrogram. It is a continuous version of autoregressive modelling whereby the next frame is predicted using an L_2 loss. During training, the

[2]The method of projection varies for different models, but typically is a form of multi-layer perceptron mapping each $\tilde{\mathbf{x}}$ to a probability vector [13, 23].

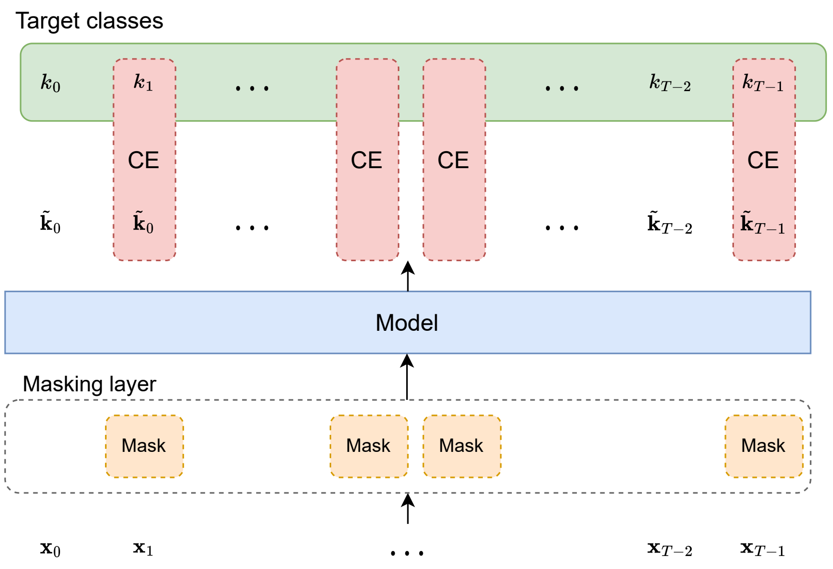


Figure 2.2: Masked language modelling (MLM) diagram. A model accepts an input sequence (bottom) where random positions have been masked, and produces a sequence of output probability distributions $[\tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1, ..., \tilde{\mathbf{k}}_{T-1}]$. These output probability distributions are compared to ground-truth target classes $[k_0, k_1, ..., k_{T-1}]$ (top) derived from the input to compute a cross-entropy (CE) loss between the target classes and output distributions for the **masked timesteps** only.

correct history for each frame (i.e. $\mathbf{x}_0, ..., \mathbf{x}_{t-1}$) is passed to the model, in a concept known as 'teacher forcing' [2]. During inference, the predicted spectrogram frame is iteratively fed back into the model until the entire output spectrogram is predicted. Of course, as discussed in our look at the first tension in Section 2.2.1, this imposes a Gaussian prior over mel-spectrogram frames, causing the spectrograms to be over-smoothed. This reduces the quality of samples produced by Tacotron, and is one of the main reasons why it has fallen out of favor since 2017/2018 when it was initially proposed. At present, GANs, DDPMs (discussed later), and discrete autoregressive models are the top-performing approaches to speech synthesis [4, 32, 39].

2.3.5. Transformers and attention

While not strictly a modelling approach, a critical feature of most modern machine learning architectures is the attention operation, and the transformer architecture which abounds in nearly all generative domains [7, 24, 57]. So, we provide a brief overview of them here.

Both the attention and transformer were introduced in [58], and intuitively provide a way for a model to learn to dynamically attend to some information (read: pass it on to the next layer in the network) while ignoring other information (read: filter it in the activations to the next layer). The attention operation acts over a 3-tuple sequence:

Q : a query sequence of vectors

K : a key sequence of vectors

V : a value sequence of vectors, of the same length as K

Using these, it produces a weighted combination over the sequence V proportionally to how well each associated item in the key sequence K matches with each item in the query sequence Q [58]:

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

Where W_Q, W_K, W_V are learnable square matrices of the same dimension as each vector in the query/key/value tuple, and d_k is the dimensionality of each key vector. Most often, a variant called multi-head attention is used, whereby a separate attention layer is applied to different dimensions of each sequence. The output of these sequences of chunked vectors is then concatenated to yield the final output. E.g. 16 head multi-headed attention would chunk each Q, K, V vector into 16 smaller vectors, apply a separate attention operation and weights to each of the 16 3-tuples, and then concatenate the results.

The transformer is simply the cascading of these attention operations into a full architecture. While there are several variants used in different circumstances, from encoder/decoder-only transformers [39] to full sequence-to-sequence transformers [59], with associated self- and cross-attention, broadly they follow the same pattern. Concretely, they form a 'block' by the sequential application of a multi-headed attention layer, an instance normalization layer, and a small feed-forward network with 2 linear layers and a single non-linearity [58]. These blocks are then repeated many times in various configurations to produce networks for use in various domains.

One of the main reasons for the explosion in popularity and use of transformers and attention is their scalability – the formulation of attention in Equation 2.3 allows the training to be done in parallel across the sequence. This allows for scaling models to many billions of parameters [57,60] since training can still be done efficiently. Also, another effect of Equation 2.3 being agnostic to the ordering of query/key/value vectors is that – when performing sequential modelling where order does matter – we must introduce explicit sequence/time information. This is typically done by adding a *positional embedding* to the sequence of vectors. There are many kinds, but the simplest is a learned positional embedding, whereby we add a different learned vector to each index in the sequence. E.g. in a transformer trained with a maximum sequence length of 200, the model will include 200 learnable vectors that are added to each input sequence to provide a way for the model to encode timing information. One last caveat is that the attention operation of

Equation 2.3 can be made causal (e.g. to do autoregressive modelling as in language models) or anti-causal by masking out appropriate values of the $(QW_Q)(KW_K)^T$ matrix [58].

2.4. Explicit disentanglement

With the key ideas and techniques of speech modelling and synthesis now clear, we proceed to give an overview of the important related work to our investigations. We start with prior efforts to explicitly disentangle different parts of speech, since an understanding of these ideas is important to understand the following section on generalizing them to methods for implicit disentanglement (Section 2.5).

2.4.1. Voice conversion

Several of our investigations look at voice conversion: the task of explicitly disentangling and replacing the speaker of a given utterance. In voice conversion (VC), an utterance spoken by one speaker is processed and re-synthesized so that it sounds as if it was spoken by a different target speaker, while keeping the content unchanged [25]. Within the context of disentanglement, VC explicitly identifies speaker identity as the characteristic of speech to isolate, modify, and recombine with the rest of speech to synthesize new utterances. As such, all models designed and trained purely for the voice conversion task are explicit disentanglement models, since they bake-in in the explicit assumption that speech can be factored into two things: speaker identity, and linguistic content. One can already see a severe limitation: where should prosody be obtained from? The typical prosody of reference speaker? Or the input utterance? This is tension 2 manifested, and it causes the output of several modern VC systems to be brittle in the face of inputs having non-trivial prosody (e.g. whispers) [38, 46]. We investigate this further in Chapter 4.

Historically, older VC methods typically could only convert to or from a single speaker (known as one-to-many or many-to-one VC models [32]). Like with the original noise disentanglement models discussed in Section 2.3.1, classical VC works also attempt to impose a simple statistical model to the contribution of speaker identity to speech. For example, works from the 1980's and 1990's such as [42] and [43] use Gaussian mixture models (GMMs) to separately model the 'spectral envelope' (pitch) of the source and target speaker and use dynamic time warping (DTW) – a technique to align two sequences of vectors – to map the pitch of one source utterance to that of the target utterance. Other models looked more closely to model the human vocal tract [61] or attempted to identify rules relating to formant identification and separation in the frequency domain for different speakers, trying to apply these rules to modify frequency bands to change the formants of the source speaker to sound like those of the target speaker [62, 63]. Unfortunately, in comparison to more modern methods, all such classical methods are very poor in terms of speech quality and in attaining accurate target speaker identity [32]. More recent VC approaches are becoming more practical in terms of speed [53], naturalness [32, 46], and performance on unseen speakers [38, 64]. Some newer methods even provide finer-grained ways to control different properties of the generated speech [46, 65], and can be used cross-lingually [66]. Most of the methods can be broadly classified into GAN-based methods [53, 64, 67], diffusion methods [68], or autoencoder-based methods [38, 46, 65] – all trained as large neural networks. Some of the recent methods can even convert to and from speakers during model training (known as zero-shot or any-to-any VC)

Of particular interest to us are the techniques these models develop to capture diverse speaker characteristics in their representations of speaker identity. Or phrased differently, how they learn to disentangle different aspects of *speaker identity*. Techniques such as the hierarchical global style token (HGST) can learn to disentangle high-level speaker characteristics (e.g. speaker gender) from low-level speaker characteristics (e.g. speaker volume, speaker microphone quality) not by explicitly defining these aspects of speaker identity, but by learning them from the data [29]. We will return to applying this technique to voice conversion in Chapter 6.

2.4.2. Generative spoken language modelling

Later on in Section 2.5 we will be interested in the task of *unconditional speech synthesis* (defined in Section 2.1.1), and it is important to distinguish this task (such as those undertaken in [20, 21]) from *generative spoken language modeling* (GSLM) – another task which involves speech synthesis without conditioning inputs. GSLM intuitively tries to apply the design principles of textual language models to speech using quantization.

Concretely, in GSLM, a large autoregressive model is typically trained on some discrete units (e.g. HuBERT features [13] or clustered spectrogram frames), similar to how a language model is trained on text data [40, 41]. The architectures used in GSLM are typically of the same design as text language models and are typically trained using either the MLM or autoregressive task (Section 2.3.3 & 2.3.4) – i.e. large transformers [39]. While this setup enables the generation of speech without any conditioning input, GSLM implies a specific model structure consisting of an encoder to quantize the continuous speech into discrete units, a language model, and a decoder [39]. I.e. you are bound by the discrete units in the model; for instance, it is not possible to interpolate between two utterances in a latent space, to map the relationships between two utterances, or to directly control speaker characteristics during generation. If this is desired, the model designer must explicitly design components to allow for control of these aspects explicitly [41]. In this way, the model still cannot truly perform disentanglement as the ‘structure’ and relationships – recall Section 2.1.3 – of a *discrete* latent space cannot have linear subspaces (i.e. low-dimensional manifolds corresponding to factors of speech variation). Intuitively, since it operates on discrete units, it cannot disentangle the fundamentally fuzzy and continuous characteristics of speech.

In contrast, when we refer to the task of 'unconditional speech synthesis' we are interested in the less developed area of implicit disentanglement, where we do not assume knowledge of particular aspects of speech beforehand. Intuitively, the task of 'unconditional speech synthesis' is the continuous variant of GSLM, whereby we do not assume that speech can be expressed as a fixed number of discrete clusters. Typically unconditional speech synthesis models use noise to generate speech directly, often via some latent representation.

The latent space should ideally be disentangled implicitly, allowing for modelling and control of the generated speech. Hence, for unconditional speech synthesis, we do not feed in expert-identified aspects of speech to condition the model's output, but rather attempt to learn them during training. In some sense, this is a more challenging task than GSLM, which is why most unconditional speech synthesis models are still evaluated on short utterances of isolated words [19] (as we also do in our first investigation in Chapter 3).

2.5. Implicit disentanglement

Contrasted with explicit disentanglement, implicit disentanglement entails attempting to learn what characteristics comprise speech and fuzzy distinctions between them from a set of utterances. Intuitively, instead of baking in domain expert assumptions about what the characteristics of speech are and how they are distinct from one another, we attempt to *learn* them the data. Our investigations in the upcoming chapters build upon the following key techniques and insights for implicit disentanglement.

2.5.1. Speech representation

As described earlier, speech representation models (which we often refer to as SSL models) attempt to process an utterance into a sequence of vectors such that the vectors are useful for many downstream tasks [24]. This entails that the latent space is disentangled, as the more disentangled the high-level characteristics of speech are in this latent space, the better they will generalize to arbitrary downstream speech processing tasks. Further, as mentioned in Section 2.1.3, speech representation models only find a disentangled latent space without performing speech synthesis. However, as we do in Chapter 4, vocoding models can be trained on top of representation models to allow mapping from the latent space back to time-domain waveforms.

Speech representation models are typically trained as large transformers with either a contrast predictive coding (CPC) loss [69] – a loss which aims to learn representations of speech by distinguishing future observations from a set of negative examples – or, more recently, with an MLM task (Section 2.3.3). Concretely, top-performing speech representation models (as evaluated on the SUPERB benchmark [24]) such as HuBERT

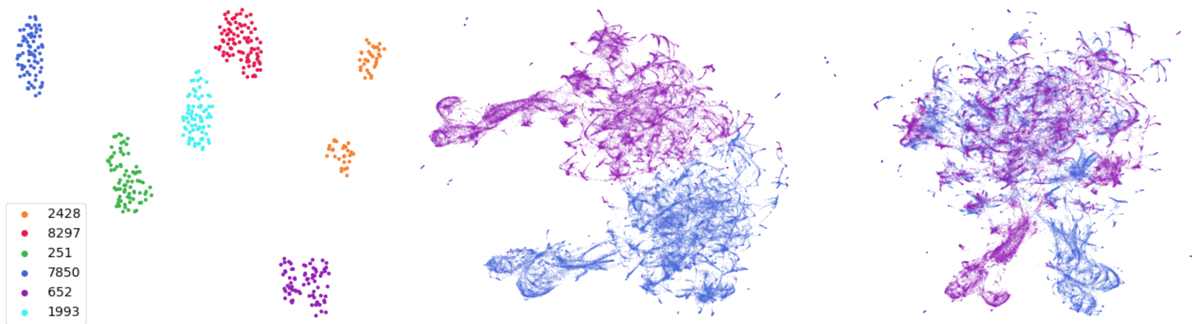


Figure 2.3: UMAP visualizations of CPC features, taken from [70]. (a) The per-utterance means of CPC features for six speakers. (b) Per-frame CPC features for the blue and purple speakers in (a). (c) Per-frame CPC features (standardized per utterance) for the same speakers.

[13] and WavLM [23] use this MLM task on clustered spectrogram frames or clustered activations from an intermediate layer within the network. The latent spaces of such models are typically evaluated on downstream tasks instead of being viewed through the context of disentanglement. So, to gain an insight into how factors of speech are represented in these SSL models, in [70][3], we investigate the speaker disentanglement properties of a high-performing CPC-based model trained as the baseline for the ZeroSpeech challenge [71]. Concretely, [70] looks at a CPC model trained on 6000 h of speech sampled at 16kHz. The model produces a 512-dimensional representation vector for every 10 ms of audio, which we refer to as *CPC features*. Using this, [70] performed two experiments to look at how one particular aspect of speech is represented: speaker identity.

Visualizing CPC feature speaker information

While previous work [69] has shown that such CPC features capture both phonetic and speaker information, it is unclear *how* the representation structures this information – i.e. the extent of disentanglement in the latent space. One expert assumption we can use to delineate speech from the rest of the utterance is to assume that speaker identity is constant within the same utterance. That is, we hypothesize that the per-utterance mean of the features captures a large degree of the speaker information. This is reasonable under the assumption that speaker identity remains constant over an utterance with other phonetic and prosodic content varying over shorter time scales [72].

As a first step towards validating this hypothesis, we explore the CPC features using the non-linear UMAP [73] projection method. Qualitatively, Figure 2.3(a) shows the per-utterance mean of CPC features for six speakers selected from the LibriSpeech `dev-clean` set [74]. The different speakers are clearly separated, showing that the mean does indeed

[3]Matthew and Herman are both non-first-authors on this publication, so we use the first-person ‘we’ in reference to the work. However, we cite all appropriate claims and extracts from the publication as if it were any other unassociated related work since it is not a first-author publication.

capture speaker information. Figure 2.3(b) then plots the UMAP projection of CPC features for individual frames for two speakers (colored blue and purple). Although the UMAP embeddings for the two speakers exhibit similar structure, they are still separated based on speaker identity. The hypothesis is then tested by standardizing the features from each speaker using the mean and variance of all CPC features from that speaker. The UMAP plot of the standardized features are shown in Figure 2.3(c), where the structures for both speakers are now much more aligned.

This actually shows that speaker information is *linearly* encoded in the CPC features, or phrase differently, that the CPC features linearly disentangle speaker information. Intuitively, this can be seen by the linear operation of standardization causing the (non-linear) UMAP projection of the two speakers' features to be largely overlapping. So, this experiment indicates that – to a large extent – SSL models like CPC *do* linearly disentangle speaker identity from the rest of speech, without being explicitly designed with speaker identity in mind.

Quantifying CPC feature speaker information

While Figure 2.3 from [70] provided a qualitative measure of the disentanglement of speaker information, the task of finding a quantitative measure of disentanglement remains. Several experiments are performed to objectively measure the linear separation of phones, speaker identity, and speaker gender in [70], however we omit them in this thesis for brevity. Summarizing their results, they largely agree and justify the qualitative visualization seen in Figure 2.3. Additional experiments in [70] on clustered CPC features also further validate our motivation for the inadequacy of discretized latent spaces for achieving linear disentanglement (Section 2.4.2). In particular, clustering CPC features with K-means causes the resulting discretized latent space to capture primarily phonetic information, discarding other aspects of speech like speaker identity.

From these experiments and visualizations, we can see that it is possible to learn a reasonably good (i.e. mostly linear) disentanglement of common speech characteristics (phone content, speaker identity, gender). What remains, however, is to observe how to *control* these representations for speech synthesis. For utterance-level characteristics like speaker identity it is fairly straightforward (Chapter 4), but for other characteristics it is non-trivial, as the sequential nature of most speech representations means that we are still introducing a discretization into the latent space. Specifically, we are forcing the representations to be discrete-time (e.g. 10 ms frames for most CPC models, 20 ms for HuBERT), which means that the linear subspaces associated with different speech characteristics may drift or change at different timesteps, making control of them difficult. We attempt to jointly model a full utterance with our first investigation (Chapter 3) on the task of unconditional speech synthesis, explained next.

2.5.2. Unconditional speech synthesis

In contrast to speech representation models, another class of models which attempts to directly model a single latent space for entire utterances is unconditional speech synthesis. Concretely, it is the task of generating coherent speech without any conditioning inputs such as text or speaker labels [19].

The task is the speech-domain analog of unconditional image synthesis, which is the task of synthesizing realistic images. And, as further explained in Chapter 3, part of our motivation for this proposal is to apply and adapt the recent techniques from the image synthesis domain [8, 16, 18] to the speech domain. Further, as in unconditional image synthesis [51], a well-performing unconditional speech synthesis model would have several useful applications: from latent interpolations between utterances and fine-grained tuning of different aspects of the generated speech to audio compression and better probability density estimation that captures different speech characteristics in a dataset.

Unconditional speech synthesis models typically fall under one of three categories: (i) Autoregressive models which iteratively synthesize one waveform sample at a time, modelling a probability distribution over a set of quantized audio levels for each sample (e.g. with Mu-law encoding). These works are typically no longer preferred in favor of the following two methods, as their speed is severely hindered by the need to perform many inference calls to synthesize an utterance. (ii) Diffusion models, trained to iteratively de-noise a random signal until it resembles coherent speech (explained in the next section). And (iii) GANs, whereby the generator maps sampled Gaussian noise to a disentangled latent vector which is then mapped to a sequence of audio features (e.g. waveform samples, spectrogram frames, or HuBERT features).

All these methods are unified in that they model the probability density (or mass, for autoregressive models) of speech, and permit the calculation of likelihood scores for a synthesized utterance. Spurred on by recent improvements in diffusion models [27] for images [7, 8, 75], there has been a substantial improvement in the quality of unconditional speech synthesis models in the last few years. The current best-performing approaches are all trained as diffusion models [20, 21]. Before this, most studies used GANs [17] that map a latent vector to a sequence of speech features with a single forward pass through the model. However, performance was limited [19, 76] due to GAN techniques available at the time, leading to GANs falling out of favour for this task. The GAN and diffusion-type models typically use noise to generate speech directly via some latent representation. Recall the broad motivation of performing the seemingly useless task of synthesizing audio without conditioning information: to learn a disentangled latent space, which would allow for more generalizable performance in both analysis and control of the generated speech.

2.6. Denoising diffusion probabilistic models

One of the newer techniques that is rising in popularity for speech, image, and even text synthesis is that of denoising diffusion probabilistic models (DDPMs), or simply 'diffusion models'. Since the math involved in deriving the diffusion equations is fairly lengthy, we provide only a high-level description here. Concretely, diffusion models [27] defines a Markov process of T steps from $t = 0...T-1$. The modelled data (e.g. waveforms, images, ...) is defined at the signal at the first timestep X_0, and the last timestep is defined as a known noise distribution (e.g. $X_T = \mathcal{N}(\mathbf{0}, \mathbf{I})$). The *forward* diffusion process then consists of a function $q(X_t|X_{t-1})$ which iteratively adds noise to a signal until at timestep T it resembles pure noise. Then, the *reverse* diffusion process $p(X_{t-1}|X_t)$ is then parameterized with a large neural network [27].

The diffusion network (the neural net associated with the reverse diffusion process p) is trained to predict the noise added through the forward diffusion process. This can be shown to be equivalent to minimizing an 'evidence lower bound' for the likelihood of the data [27]. In this way, during inference, the model iteratively de-noises a pure noise signal X_T into a structured waveform X_0 (or image, or any other domain). At each inference step $p(X_{t-1}|X_t)$, the original noise signal is slightly de-noised until – in the last step $t = 0$ – it resembles coherent speech. Since the introduction of DDPMs in 2015, there has been an explosion of the variety and formulations of this 'iterative denoising' approach. For images, denoising diffusion implicit models have become popular with their ability to perform the diffusion task in a smaller dimensional space than the base images [77]. In TTS, score-based generative models have become popular, and rephrase the standard DDPM task in terms of ordinary and stochastic differential equations where the timestep t becomes continuous [78]. Even further afield, recent work on Poisson flow models attempts to re-interpret diffusion (developed from the theory of thermodynamics) through the lens of electrostatics, giving slightly different iterative denoising equations with some favorable properties [79]. In this thesis we stick to standard DDPMs to keep complexity to a minimum.

What makes diffusion somewhat unique is that it does not neatly fall into a 'continuous' or 'discrete' category. Depending on the distribution chosen for the terminal distribution X_T, a DDPM can be either. For example, if predicting spectrograms like GradTTS [3], a continuous Gaussian prior is used, making the diffusion a continuous process. Meanwhile, [80] assumes a uniform categorical distribution for X_T and performs discrete diffusion over text, whereby at each timestep, random characters in the text are either swapped for another random character (in the forward process) or replaced with the model's prediction of a denoised character (in the reverse process). In this way, so-called 'multinomial diffusion' with categorical distributions can be used to make diffusion a discrete modelling tool. We explore how this framework can be combined with SSL features and adapted for ASR in Chapter 5.

One drawback of DDPMs and other diffusion-type models is that typically they are slow. This is due to the need for repeated forward passes through the models during inference to get from $t = T$ to $t = 0$. There are some techniques, however, to speed up the sampling process, typically trading off the output quality in return [81]. In terms of latent spaces for diffusion models, since the distribution of X_T is a known isotropic distribution, it is still largely entangled. So, what is typically done is to define an intermediate step (e.g. $t = \frac{T}{2}$) as the latent space where different components of the signal are – to an extent – disentangled [82]. So, when we compare to the 'latent space' of diffusion models as we do in Chapter 3, we are comparing to the vector space spanned by samples at a specified intermediary timestep X_t.

The main benefit of diffusion models is that they scale very well to large model sizes and datasets [7, 8], and likely will play an increasing role in model designs in the coming years. As hinted in Tension 1 (Section 2.2.1), diffusion models help overcome the challenge of representing complex high dimensional probability distributions by breaking the problem down into T smaller steps. At each step, the model transforms the data from a simpler distribution (from the simplest prior at $t = T$) to a more complex distribution (until the real data distribution at $t = 0$). It is this repeated inference and denoising paradigm that allows the model to learn to effectively map an extremely simple distribution (e.g. isotropic Gaussian) to a very complex one (e.g. mel-spectrogram) by slowly refining it over many incremental steps.

2.7. Summary

Recall our motivation in Section 1.1: we wish to improve speech processing and synthesis by attempting to understand the gap between discrete approaches that explicitly demarcate factors of speech (favored in current systems) versus the way humans process speech as a continuous signal without such explicit factorizations. To grapple with this, we introduced four investigations (the remaining chapters in this thesis) to probe and tackle this tension from different angles. Before these investigations can be performed, we must understand the concrete nature of this tension and be on the same page with regard to our terminology and toolbox of continuous and discrete modelling techniques.

To achieve this, this chapter began with an overview of the terminology we use and what specifically is meant by disentanglement (Section 2.1), both conceptually and how it is typically measured. Then, the nature of the continuous-discrete tension is explored in Section 2.2, highlighting two distinct facets that make it difficult to learn linearly disentangled representations with purely continuous methods. Next, an overview of some of the key techniques used in speech processing is given, followed by a description of a few important prior efforts at learning both explicit and implicit disentanglement of speech.

The techniques and efforts highlighted will be important in the upcoming chapters, where we will combine, adapt, and build upon the techniques discussed here to try answer our research questions posed in Section 1.3.

Chapter 3

Unconditional speech synthesis

Publications

This chapter is adapted from our publications:

Published conference paper: M. Baas, H. Kamper, "GAN You Hear Me? Reclaiming Unconditional Speech Synthesis from Diffusion Models," in *IEEE Spoken Language Technology Workshop*, 2022.

Published journal article: M. Baas, H. Kamper, "Disentanglement in a GAN for Unconditional Speech Synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 1324-1335, 2024.

This chapter presents our investigation into research question 1 (Section 1.3): why can we not learn disentanglement by adapting methods from the image synthesis domain? To answer this question, we build upon techniques proposed for image synthesis to develop a new model for unconditional speech synthesis – AudioStyleGAN (ASGAN). The model is designed to perform unconditional speech synthesis by learning a disentangled latent space. Concretely, it maps sampled noise to a disentangled latent vector which is then mapped to a sequence of audio features in such a way that signal aliasing is suppressed at every layer in the model (improving disentanglement). To successfully train ASGAN, we introduce a number of new techniques, including a modification to adaptive discriminator augmentation which probabilistically skips discriminator updates. We benchmark it on the small-vocabulary Google Speech Commands digits dataset, where it achieved state-of-the-art results in unconditional speech synthesis at the time of our first publication [83]. It is also substantially faster than prior top-performing diffusion models.

But recall our primary aim: we wish to investigate whether the process of designing for disentanglement aids generalization. To assess this, we go on to confirm that ASGAN's latent space is highly disentangled with both objective measures and applications to downstream tasks. We demonstrate how simple linear operations in the latent space can be used to perform several tasks unseen during training – voice conversion, speech enhancement, speaker verification, and keyword classification. Its performance on these unseen tasks indicates that (1) implicitly disentangled latent spaces *can* be used to aid generalization to unseen tasks, and (2) techniques from image synthesis *can* be modified to aid in learning disentanglement in the speech domain. This chapter has associated code, pretrained models, and audio samples: https://github.com/RF5/simple-asgan/.

3.1. Related work and image synthesis techniques

Recall from Section 2.5.2: unconditional speech synthesis is the task of producing speech without conditioning inputs, directly from some continuous space. The current best-performing approaches for unconditional speech synthesis are all based on autoregressive [50] or diffusion modelling [20, 21]. Motivated by the recent developments in the StyleGAN literature [15,16,18] for image synthesis, we aim to infuse GANs with modelling techniques for disentangling images in order to improve unconditional speech synthesis. We are particularly interested in image synthesis models' ability to learn continuous, disentangled latent spaces. So, we adapt the style layers present in StyleGAN3 to remove signal aliasing caused by the non-linearities in the network [18], as well as adapting other GAN training techniques to improve robustness for audio.

Earlier studies in the unconditional speech synthesis domain [19, 76] attempted to use GANs [17] for unconditional speech synthesis, which has the advantage of requiring only a single pass through the model. While results showed some initial promise, performance was poor in terms of speech quality and diversity, with the more recent diffusion models performing much better [21]. However, there have been substantial improvements in GAN-based modelling for image synthesis in the intervening years [15, 16, 54]. So, we start from the framework of older GAN-based approaches and attempt to improve them by adapting lessons from these recent image synthesis studies.

Some of these innovations in GANs are modality-agnostic: R_1 regularization [55] and exponential moving averaging of generator weights [84] can be directly transferred from the vision domain to speech. Other techniques, such as the carefully designed anti-aliasing filters between layers in StyleGAN3 [18] require specific adaptation; in contrast to images, there is little meaningful information in speech below 90 Hz, necessitating a redesign of the anti-aliasing filters.

In a related research direction, Beguš et al. [76, 85, 86] and Chen and Elsner [87] have been studying how GAN-based unconditional speech synthesis models internally perform lexical and phonological learning, and how this relates to human learning. Most of these studies, however, rely on older GAN synthesis models. We hope that by developing better-performing GANs for unconditional speech synthesis, we can contribute to improving such investigations. Recently, [88] attempted to directly use StyleGAN2 for conditional and unconditional synthesis of emotional vocal bursts. This further motivates a reinvestigation of GANs, but here we look specifically at the generation of speech rather than paralinguistic sounds. Lastly, in even more recent follow-up work by Zhu et. al. [89], they benchmark our model against their newly proposed diffusion-based approach, showing that their new diffusion method yields improvements on the results we show in this chapter.

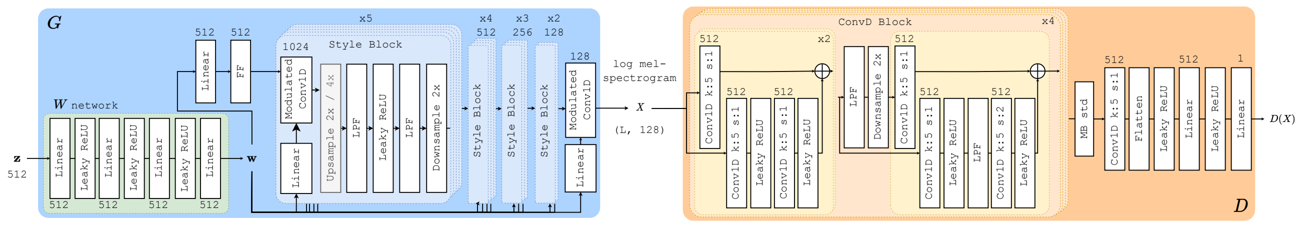


Figure 3.1: The ASGAN generator G (left) and discriminator D (right). FF, LPF, Conv1D indicate Fourier feature [18], low-pass filter, and 1D convolution layers, respectively. The number of output features/channels are indicated above linear and convolutional layers. Stacked blocks indicate a layer repeated sequentially.

3.2. Audio Style GAN

Our model is based on the StyleGAN family of models [15] for image synthesis. We adapt and extend the approach to audio, and therefore dub our model AudioStyleGAN (ASGAN).

The model follows the setup of a standard GAN with a single generator network G and discriminator network D [17]. The generator G accepts a vector $\mathbf{z}$ sampled from a normal distribution and processes it into a sequence of speech features X. In this work, we restrict the sequence of speech features X to always have a fixed pre-specified duration. The discriminator D accepts a sequence of speech features X and yields a scalar output. Recalling Section 2.3.2, our model uses a non-saturating logistic loss [17] whereby D is optimized to raise its output for X sampled from real data and minimize its output for X produced by the generator. Meanwhile, the generator G is optimized to maximize $D(X)$ for X sampled from the generator, i.e. when $X = G(\mathbf{z})$. The speech features X are converted to a waveform using a pretrained HiFiGAN vocoder [53]. During training, a new adaptive discriminator updating technique is added to ensure stability and convergence. Each component is described in detail below.

3.2.1. Generator

The architecture of the generator G is shown on the left of Figure 3.1. It consists of a latent mapping network W that converts $\mathbf{z}$ to a disentangled latent space, a special Fourier feature layer which converts a single vector from this latent space into a sequence of cosine features of fixed length, and finally a convolutional encoder which iteratively refines the cosine features into the final speech features X.

Mapping network

The mapping network W is a simple multi-layer perceptron consisting of several linear layers with leaky ReLU activations between them. As input, it takes in a normally distributed vector $\mathbf{z} \sim Z = \mathcal{N}(\mathbf{0}, \mathbf{I})$; in all our experiments we use a 512-dimensional multivariate normal vector, $\mathbf{z} \in \mathbb{R}^{512}$. Passing this vector through the mapping network produces a latent vector $\mathbf{w} = W(\mathbf{z})$ of the same dimensionality as $\mathbf{z}$. As explained in [15], the primary purpose of W is learn to map noise to a linearly disentangled W space, as ultimately this will allow for more controllable and understandable synthesis. W is coaxed into learning such a disentangled representation because it can only *linearly modulate* channels of the cosine features in each layer of the convolutional encoder (explained below). So, if $\mathbf{w}$ is to linearly shape the speech features, W must learn a mapping that organizes the random normal space $\mathbf{z}$ into one which linearly disentangles common factors of speech variation.

Convolutional encoder

The convolutional encoder begins by linearly projecting $\mathbf{w}$ as the input to a Fourier feature layer [90] as shown in Figure 3.1. Concretely, we use the Gaussian Fourier feature mapping from [90] and incorporate the transformation proposed in StyleGAN3 [18]. This layer samples a frequency and phase from a Gaussian distribution for each output channel (fixed at initialization). The layer then linearly projects the input vector to a vector of phases which are added to the initial random phases. The output sequence is obtained as the plot of a cosine function with these frequencies and phases, one frequency/phase for each output channel. Intuitively, the layer provides a way to learn a mapping between a single vector ($\mathbf{w}$) into a *sequence* of vectors at the output of the Fourier feature layer, which provides the base from which the rest of the model can iteratively operate on the feature sequence to eventually arrive at a predicted mel-spectrogram.

The sequence produced by the Fourier feature layer is iteratively passed through `Style Blocks`, which is based on the encoder layers of the StyleGAN family of models. In each layer, the input sequence and style vector $\mathbf{w}$ are passed through a modulated convolution layer [16]: the final convolution kernel is computed by multiplying the layer's learnt kernel with the style vector derived from $\mathbf{w}$, broadcasted over the length of the kernel. In this way, the latent vector $\mathbf{w}$ linearly modulates the kernel in each convolution.

To ensure the signal does not experience aliasing due to the non-linearity, the leaky ReLU layers are surrounded by layers responsible for anti-aliasing (explained below). All these layers comprise a single `Style Block`, which is repeated in groups of 5, 4, 3, and finally 2 blocks. The last block in each group upsamples by $4\times$ instead of $2\times$, thereby increasing the sequence length by a factor of 2 for each group. A final 1D convolution layer projects the output from the last group into the audio feature space (e.g. log mel-spectrogram or HuBERT features, see Section 2.5.1), as illustrated in the middle of Figure 3.1.

Anti-aliasing filters

From image synthesis with GANs [18], we know that the generator must include anti-aliasing filters for the signal propagating through the network to approximately satisfy the Nyquist-Shannon sampling theorem. This is why, before and after a non-linearity, we include upsampling, low-pass filter (LPF), and downsampling layers in each `Style Block`. The motivation from [18] is that non-linearities introduce arbitrarily high-frequency information into the output signal. The signal we are modelling (speech) is continuous, and the internal discrete-time features that are passed through the network is therefore a digital representation of this continuous signal. From the Nyquist-Shannon sampling theorem, we know that for such a discrete-time signal to accurately reconstruct the continuous signal, it must be bandlimited to 0.5 cycles/sample. If not, [18] showed that the generator learns to use aliasing artefacts to fool the discriminator, to the detriment of quality and control of the final output. To address this, we follow [18]: we approximate an ideal continuous LPF by first upsampling to a higher sample rate, applying a discrete LPF as a 1D convolution, and only then applying the non-linearity. This high-frequency signal is then passed through an anti-aliasing discrete LPF before being downsampled again to the original sampling rate.

Because of the practical imperfections in this anti-aliasing scheme, the cutoff frequencies for the LPFs must typically be much lower than the Nyquist frequency of 0.5 cycles/sample. We reason that the generator should ideally first focus on generating coarse features before generating good high-frequency details, which will inevitably contain more trace aliasing artifacts. So we design the filter cutoff to begin at a small value in the first `Style Block`, and increase gradually to near the critical Nyquist frequency in the final block. In this way, aliasing is kept fairly low throughout the network, with very high frequency information near the Nyquist frequency only being introduced in the last few layers.

3.2.2. Discriminator

The discriminator D has a convolutional architecture similar to [16]. It takes a sequence of speech features X as input and predicts whether it is generated by G or sampled from the dataset. Concretely, D consists of four `ConvD Blocks` and a network head, as show in Figure 3.1. Each `ConvD Block` consists of 1D convolutions with skip connections, and a downsampling layer with an anti-aliasing LPF in the last skip connection. Since the output of D is a single number (and not a discrete-time sequence), we simply set the LPF cutoff to the Nyquist frequency for all layers. The number of layers and channels are chosen so that D has roughly the same number of parameters as G. D's head consists of a minibatch standard deviation [84] layer and a 1D convolution layer before passing the flattened activations through a final linear projection head to arrive at the logits. Both D and G are trained using the non-saturating logistic loss (Equation 2.1 and 2.2).

3.2.3. Vocoder

The generator G and discriminator D operate on sequences of speech features and not on raw waveform samples. Once both G and D are trained, we need a way at convert these speech features back to waveforms. For this we use a pretrained HiFi-GAN vocoder [53] that vocodes either log mel-scale spectrograms [91] or HuBERT features [13]. HuBERT is a self-supervised speech representation model (see Section 2.5.1) that learns to encode speech in a 50 Hz vector sequence using a masked token prediction task. These learnt features are linearly predictive of several high-level characteristics of speech such as phone identity, making it useful as a feature extractor when trying to learn disentangled representations.

3.2.4. Implementation

We train two variants of our model: a log mel-spectrogram model and a HuBERT model. And, extending our initial investigation in [83], we train additional variants of the HuBERT-based model to understand key design choices.

ASGAN variants

The log mel-spectrogram model architecture is shown in Figure 3.1, where mel-spectrograms are computed with 128 mel-frequency bins at a hop and window size of 10 ms and 64 ms, respectively. Each 10 ms frame of the mel-scale spectrograms is scaled by taking the natural logarithm of the spectrogram magnitude. The HuBERT-based model is identical except that it only uses only half the sequence length (since HuBERT features are 20 ms instead of the 10 ms spectrogram frames) and has a different number of output channels in the four groups of `Style Blocks`: `[1024, 768, 512, 512]` convolution channels instead of the `[1024, 512, 256, 128]` used for the mel-spectrogram model. This change causes the HuBERT variant to contain 51M parameters, as opposed to the 38M parameters in the mel-spectrogram model.

Vocoder variants

The HiFi-GAN vocoder for both HuBERT and mel-spectrogram features is based on the original author's implementation [53]. The HuBERT feature HiFi-GAN is trained on the LibriSpeech `train-lean-100` multispeaker speech dataset [74] to vocode activations extracted from layer 6 of the pretrained HuBERT `Base` model provided with fairseq [92]. The mel-spectrogram HiFiGAN is trained on the Google Speech Commands dataset. Both are trained using the original V1 HiFi-GAN configuration (number of updates, learning and batch size parameters) from [53].

Optimization

Both ASGAN variants are trained with Adam [93] ($\beta_1 = 0, \beta_2 = 0.99$), clipping gradient norms at 10, and a learning rate of $3 \cdot 10^{-3}$ for 520k iterations with a batch size of 32. As discussed in Section 2.3.2, GANs are notoriously tricky to train. So, to ensure model convergence, we use several critical tricks to stabilize GAN training: (i) equalized learning rate is used for all trainable parameters [84]; (ii) leaky ReLU activations with $\alpha = 0.1$; (iii) exponential moving averaging for the generator weights (for use during evaluation) [84]; (iv) R_1 regularization [55]; and (v) a 100-times smaller learning rate for the mapping network W, since it needs to be updated slower compared to the convolution layers in the main network branch [18].

Adaptive discriminator updates

We also introduce a new technique for updating the discriminator. Concretely, we first scale D's learning rate by 0.1 compared to the generator as otherwise we find it overwhelms G early on in training. Additionally we employ a dynamic method for updating D, inspired by adaptive discriminator augmentation [54]: during each iteration, we skip D's update with probability p. The probability p is initialized at 0.1 and is updated every 16th generator step or whenever the discriminator is updated. We keep a running average r_t of the proportion of D's outputs on real data $D(X)$ that are positive (i.e. that D can confidently identify as real). Then, if r_t is greater than 0.6 we increase p by 0.05 (capped at 1.0), and if r_t is less than 0.6 we decrease p by 0.05 (limited at 0.0). In this way, we adaptively skip discriminator updates. When D becomes too strong, both r_t and p rise, and so D is updated less frequently. Conversely, when D becomes too weak, it is updated more frequently. We found this new modification to be critical for ensuring that D does not overwhelm G during training.

We also use the traditional adaptive discriminator augmentation [54] where we apply the following transforms to the model input with the same probability p: (i) adding Gaussian noise with $\sigma = 0.05$; (ii) random scaling by a factor of 1 ± 0.05; and (iii) randomly replacing a subsequence of frames from the generated speech features with a subsequence of frames taken from a real speech feature sequence. This last augmentation is based on the fake-as-real GAN method [94] and is important to prevent gradient explosion later in training.

Anti-aliasing filters

For the anti-aliasing LPF filters we use windowed `sinc` filters with a width-9 Kaiser window [95]. For the generator (all variants), the first `Style Block` has a cutoff at $f_c = 0.125$ cycles/sample which is increased in an even logarithmic scale to $f_c = 0.45$ cycles/sample in the second-to-last layer, keeping this value for the last two layers to fill in the last high frequency detail. Even in these last layers we use a cutoff below the Nyquist frequency to ensure the imperfect LPF still sufficiently suppresses aliased frequencies. For the discriminator we are less concerned about aliasing as it does not generate a continuous signal, so we use a $f_c = 0.5$ cycles/sample cutoff for all `ConvD Blocks`.

All models are trained using mixed FP16/FP32 precision on a single NVIDIA Quadro RTX 6000 using PyTorch 1.11. Trained models and code are available at `https://github.com/RF5/simple-asgan/`.

3.3. Experimental setup: unconditional synthesis

3.3.1. Data

To compare to existing unconditional speech synthesis models, we use the Google Speech Commands dataset of isolated spoken words [96]. As in other studies [19–21], we use the subset corresponding to the ten spoken digits “zero” to “nine” (called SC09). The digits are spoken by various speakers under different channel conditions. This makes it a challenging benchmark for unconditional speech synthesis. All utterances are roughly a second long and are sampled at 16 kHz. Utterances less than a second are padded to a full second.

3.3.2. Evaluation metrics

We train and validate our models on the official training/validation/test split from SC09. We then evaluate unconditional speech synthesis quality by seeing how well newly generated utterances match the distribution of the SC09 test split. We use metrics similar to those for image synthesis; they try to measure either the *quality* of generated utterances (realism compared to actual audio in the test set), or the *diversity* of generated utterances (how varied the utterances are relative to the test set), or a combination of both.

These metrics require extracting features or predictions from a supervised speech classifier network trained to classify the utterances from SC09 by what digit is spoken. While there is no consistent pretrained classifier used for this purpose, we opt to use a ResNeXT architecture [97], similar to previous studies [20, 21]. The trained model has a 98.1% word classification accuracy on the SC09 test set, and we make the model code and checkpoints available for future comparisons.[1] Using either the classification output or 1024-dimensional features extracted from the penultimate layer in the classifier, we consider the following metrics.

[1] `https://github.com/RF5/simple-speech-commands`

- *Inception score* (IS) measures the diversity and quality of generated samples by evaluating the Kullback-Leibler (KL) divergence between the label distribution from the classifier output and the mean label distribution over a set of generated utterances [52].

- *Modified Inception score* (mIS) extends the diversity measurement aspect of IS by incorporating a measure of intra-class diversity (in our case over the ten digits) to reward models with higher intra-class entropy [98].

- *Fréchet Inception distance* (FID) computes a measure of how well the distribution of generated utterances matches the test-set utterances by comparing the classifier features of generated and real data [51].

- *Activation maximization* (AM) measures generator quality by comparing the KL divergence between the classifier class probabilities from real and generated data, while penalizing high classifier entropy samples produced by the generator [99]. Intuitively, this attempts to account for possible class imbalances in the training set and intra-class diversity by incorporating a term for the entropy of the classifier outputs for generated samples.

But, recall our overall goal for this chapter: to explore whether we can learn and utilize a disentangled latent space to improve generalization. So, a major motivation for ASGAN's design is latent-space disentanglement. After the headline results, we therefore perform two sets of experiments: one to intrinsically measure how disentangled the latent space of ASGAN is, and one to measure its generalization capabilities on downstream tasks. We start with explaining how the latent space disentanglement is measured below, and then investigate downstream tasks later in Section 3.5.

Following from the discussion in Section 2.1.3, we intrinsically evaluate disentanglement using two metrics on the Z and W latent spaces:

- *Path length* measures the mean L_2 distance moved by the classifier features when the latent point ($\mathbf{z}$ or $\mathbf{w}$) is randomly perturbed slightly, averaged over many perturbations [15]. A lower value indicates a smoother latent space.

- *Linear separability* utilizes a linear support vector machine to classify the digit of a latent point. The metric is computed as the additional information (in terms of mean entropy) necessary to correctly classify an utterance given the class prediction from the linear support vector machine [15]. A lower value indicates a more linearly disentangled latent space.

These metrics are averaged over 5000 generated utterances for each model. As in [15], for linear separability we exclude half the generated utterances for which the ResNeXT classifier is least confident in its prediction.

To give an indication of naturalness, we compute an estimated mean opinion score (eMOS) using a pretrained `Wav2Vec2 small` baseline from the VoiceMOS challenge [100]. This model is trained to predict the naturalness score that a human would assign to an utterance from 1 (least natural) to 5 (most natural). We also perform an actual subjective MOS evaluation using Amazon Mechanical Turk to obtain 240 opinion scores for each model with 12 speakers listening to each utterance. Finally, the speed of each model is also evaluated to highlight the benefit that GANs can produce utterances in a single inference call, as opposed to the many inference calls necessary with autoregressive or diffusion models.

3.3.3. Baseline systems

We compare to the following unconditional speech synthesis methods: WaveGAN [19], DiffWave [20], autoregressive SaShiMi and Sashimi+DiffWave [21]. Last-mentioned was the current best-performing model on SC09 at the time of our original publication [83]. For WaveGAN we use the trained model provided by the authors [19], while for DiffWave we use an open-source pretrained model.[2] For the autoregressive SaShiMi model, we use the code provided by the authors to train an unconditional SaShiMi model on SC09 for 1.1M updates [21].[3] Finally, for the SaShiMi+DiffWave diffusion model, we modify the autoregressive SaShiMi code and combine it with DiffWave according to [21]; we train it on SC09 for 800k updates with the hyperparameters in the original paper [21].[3]

The autoregressive SaShiMi is originally evaluated using a form of rejection sampling [21] to retain only high probability generated samples for evaluation. We could also perform sampling tricks on all the models (since both diffusion and GAN models also have tractable likelihood measures). However, in the interest of an fair and simple comparison of the inherent performance of each model, we opt to keep the same and most general sampling method for each model (including ASGAN). So, in all experiments, we perform direct sampling from the latent space for the GAN and diffusion models according to the original papers. For the autoregressive models, we directly sample from the predicted distribution for each time-step.

3.4. Results: unconditional speech synthesis

3.4.1. Comparison to baselines

We present our headline results in Table 3.1, where we compare previous state-of-the-art unconditional speech synthesis approaches to the proposed ASGAN model. As a reminder, IS, mIS, FID and AM measure generated speech diversity and quality relative to the test

[2] `https://github.com/RF5/DiffWave-unconditional`

[3] `https://github.com/RF5/simple-sashimi`

Table 3.1: Results measuring the quality and diversity of generated samples from unconditional speech synthesis models together with train/test set toplines for the SC09 dataset. Subjective MOS values with 95% confidence intervals are shown. IS, mIS, FID, AM are averaged over 5000 generated utterances generated from each model (or drawn from the train/test set for topline scores).

Model	IS ↑	mIS ↑	FID ↓	AM ↓	eMOS ↑	MOS ↑
Train set	9.37	237.6	0	0.20	2.41	3.74 ± 0.12
Test set	9.36	242.3	0.01	0.20	2.43	3.88 ± 0.12
WaveGAN [19]	4.45	34.6	1.77	0.81	1.06	2.88 ± 0.16
DiffWave [20]	5.13	49.6	1.68	0.68	1.66	3.43 ± 0.14
SaShiMi [21]	3.74	18.9	2.11	0.99	1.58	3.19 ± 0.15
SaShiMi+DiffWave	5.44	60.8	1.01	0.61	1.89	3.33 ± 0.12
ASGAN (mel-spec.)	7.02	162.8	0.56	0.36	1.76	3.51 ± 0.13
ASGAN (HuBERT)	**7.67**	**226.7**	**0.14**	**0.26**	**1.99**	$\mathbf{3.68 \pm 0.13}$

set; eMOS and MOS are measures of generated speech naturalness. We see that both variants of ASGAN outperform the other models on most metrics. The HuBERT variant of ASGAN in particular performs best across all metrics. The improvement of the HuBERT ASGAN over the mel-spectrogram variant is likely because the high-level HuBERT speech representations make it easier for the model to disentangle common factors of speech variation. The previous best unconditional synthesis model, SaShiMi+DiffWave, still outperforms the other baseline models, and it appears to have comparable naturalness (similar eMOS and MOS) to the mel-spectrogram ASGAN variant. However, it appears to match the test set more poorly than either ASGAN variant on the other diversity metrics.

The latent space disentanglement metrics and generation speed are given in Table 3.2. These results are more mixed, with WaveGAN being the fastest model and the one with the shortest latent path length in the Z-space. However, this is somewhat misleading since WaveGAN's samples have low quality (naturalness, as measured by eMOS/MOS) and poor diversity compared to the other models (Table 3.1). This means that WaveGAN's latent space is a poor representation of the true distribution of speech in the SC09 dataset, allowing it to have a very small path length as most paths do not span a diverse set of speech variation.

In terms of linear separability, ASGAN again yields substantial improvements over existing models. The results confirm that ASGAN has indeed learned a disentangled latent space – a primary motivation for the model's design. Specifically, this shows that the idea from image synthesis of using the latent $\mathbf{w}$ vector to linearly modulate convolution kernels can also be applied to speech. Reflecting back on our Research Question 1 for this chapter (Section 1.3), this gives us the first chunk of evidence to answer in the affirmative: we *can* – to an extent – adapt image disentanglement techniques to the speech domain. But, to

Table 3.2: Latent-space disentanglement and speed metrics. Speed is measured as the number of samples that can be generated per unit time on a single NVIDIA Quadro RTX 6000 using a batch size of 1, given in ksamples/sec. Some models do not have a W-space (WaveGAN) or any continuous latent space (SaShiMi).

	Path length ↓		Separability ↓		
Model	Z	W	Z	W	Speed ↑
WaveGAN [19]	**1.03**	—	4.86	—	**2214.71**
DiffWave [20]	$2.72 \cdot 10^6$	$7.27 \cdot 10^5$	6.09	6.58	0.83
SaShiMi [21]	—	—	—	—	0.14
SaShiMi+DiffWave	$2.89 \cdot 10^6$	$1.24 \cdot 10^6$	4.07	2.34	0.47
ASGAN (mel-spec.)	6.77·10	3.21·10	1.81	1.01	875.45
ASGAN (HuBERT)	3.50·10	**1.84·10**	**1.40**	**1.00**	816.27

assess whether it aids in generalization, we need to see how ASGAN performs when applied to tasks unseen during training, evaluated later in Section 3.7. Regardless of performance, the speed of all the convolutional GAN models (WaveGAN and ASGAN) is significantly better than the diffusion and autoregressive models, as reasoned in Section 3.3.2.

3.4.2. Ablation experiments

While the previous comparisons demonstrated the overall success of ASGAN's design, we are still not certain which specific decisions from Section 3.2 are responsible for its performance and degree of disentanglement. So, we perform several ablations of the HuBERT ASGAN model with specific components removed from the full model. Concretely, we ablate four key design choices: we train a variant without adaptive discriminator updates (Section 3.2.4), a variant without adaptive discriminator augmentation (Section 3.2.4), and a variant without any anti-aliasing filters (Section 3.2.1). Finally, we also train a variant without modulated convolutions (Section 3.2.1) such that **w** only controls the initial features passed to the generator's convolutional encoder.

Table 3.3 shows the results for the ablated ASGAN approaches on a subset of the metrics. We see that on the quality and diversity metrics, the base ASGAN is best, while the models without adaptive discriminator updates and augmentation have better latent space disentanglement. However, recall that the main reason for including these was *not* to optimize disentanglement, but rather to ensure training stability and performance. As reasoned in Section 3.2, without either adaptive updates or augmentation, the discriminator has a much easier task and begins to dominate the generator, confidently distinguishing between real and generated samples. So, while this makes optimization easier (leading to a smoother latent space), it means that the generator does not effectively learn from the adversarial task. A similar phenomenon can be seen with WaveGAN in Table 3.2, where it scored well on the disentanglement metrics but had poor output quality in Table 3.1.

Table 3.3: Ablations of model design choices, comparing quality, diversity, and latent-space disentanglement of several ASGAN variants.

	Quality & diversity			Path length ↓	
Variant	mIS ↑	FID ↓	eMOS ↑	Z	W
ASGAN (HuBERT)	**226.7**	**0.14**	**1.99**	35.0	18.4
w/o adaptive D updates	1.4	19.27	0.69	10.8	7.6
w/o adaptive D augmentation	2.3	13.72	0.68	**8.4**	**5.0**
w/o anti-aliasing filters	102.7	3.31	1.81	75.8	63.1
w/o modulated convolution	2.4	10.96	0.63	111.4	48.5

When anti-aliasing filters are removed, both latent space disentanglement and synthesis quality are reduced, being slightly worse in all metrics compared to the full model. This validates our design motivation for the inclusion of low-pass filters to suppress aliased high-frequency content in the layer activations in Figure 3.1. Finally, the variant without the linear influence of $\mathbf{w}$ on each layer's activations (i.e. without the modulated convolutions) is also worse than the baseline model in all metrics considered.

Overall, we can see from Table 3.3 that each of the key design aspects from Section 3.2 are necessary to achieve both high latent space disentanglement and synthesis quality in a single model – the main requirement for it to be performant on unseen downstream tasks, which we look at next.

3.5. Unseen tasks and linear latent operations

We have now shown intrinsically that ASGAN leads to a disentangled latent space. In this section and the ones that follow we show that ASGAN can also be used extrinsically to perform tasks unseen during training through linear manipulations of its latent space. From this point onwards we use the HuBERT ASGAN variant.

As a reminder, the key aspect of ASGAN's design is the latent space associated with the vector $\mathbf{w}$ is linearly disentangled. The idea is that, since the $\mathbf{w}$ vector can only linearly affect the output of the model through the Fourier feature layer and modulated convolutions (Figure 3.1), W must learn to linearly disentangle common factors of speech variation. If this turns out to be true – i.e. the space is indeed disentangled – then these factors should correspond to linear directions in the W latent space. As originally motivated in studies on image synthesis [15, 16], this would mean that linear operations in the latent space should correspond to meaningful edits in the generated output. In our case, this would mean that if we know the relationship between two utterances, then the *linear* distance between the latent vectors $\mathbf{w}$ of those utterances should reflect that relationship. E.g. consider a collection of utterances differing only in noise level but otherwise having the same properties. If the space disentangles noise levels, then we should expect the latent points of these utterances to lie in the same linear subspace, reflective of the level of noise.

3.5.1. Projecting to the latent space

Before defining how unseen tasks can be phrased as latent operations, we first need to explain how we invert a provided utterance to a latent vector $\mathbf{w}$. We use a method similar to [16] where we optimize a $\mathbf{w}$ vector while keeping G and the speech feature sequence X fixed. Concretely, $\mathbf{w}$ is initialized to the mean $\bar{\mathbf{w}} = \mathbb{E}[W(\mathbf{z})]$ vector over 100k samples and then fed through the network to produce a candidate sequence $\tilde{X}$. An L_2 loss is then formed as the mean square distance between each feature in the candidate sequence $\tilde{X}$ and the target sequence X. Optimization follows [16], using Adam for 1000 iterations with a maximum learning rate of 0.1, and with Gaussian noise added to $\mathbf{w}$ in the first 750 iterations. The variance of this noise is set to be proportional to the average square L_2 distance between the mean $\bar{\mathbf{w}}$ and the sampled $\mathbf{w}$ vectors.

3.5.2. Downstream tasks

We look at several downstream tasks, each of which can be phrased as linear operations in the latent W-space.

Style mixing

We can perform voice conversion or speech editing by using style mixing [15] of the latent vectors $\mathbf{w}$. Concretely, we can project two utterances X_1 and X_2 to their latent representations $\mathbf{w}_1$ and $\mathbf{w}_2$. Then – recalling the architecture in Figure 3.1 – we can use different $\mathbf{w}$ vectors as input into each `Style Block`. According to our design motivation of the anti-aliasing filters in Section 3.2.4, the *coarse styles* are captured in the earlier layers, and the *fine styles* are introduced in later layers. So we can perform voice conversion from X_1's speaker to X_2's speaker by conditioning later layers with $\mathbf{w}_2$ while still using $\mathbf{w}_1$ in earlier layers. This causes the generated utterance to inherit the speaking style (fine) from the target utterance X_2, but retain the word identity (coarse) from X_1. By doing the opposite we can also do speech editing: having the speaker from X_1 say the word in X_2 by conditioning earlier layers with $\mathbf{w}_2$ while keeping $\mathbf{w}_1$ in later layers. Furthermore, because the W latent space is continuous, we can interpolate between retaining and replacing the coarse and fine styles to achieve varying degrees of voice conversion or speech editing. We do these style mixing experiments in Section 3.7.1. In these experiments, we treat the $\mathbf{w}$ vector inputs to the last five `Style Block`'s as the fine styles, and all earlier $\mathbf{w}$ inputs as the coarse styles.

Speech enhancement

Speech enhancement is the task of removing noise from an utterance [101]. Intuitively, if ASGAN's W-space is linearly disentangled, there should be a single direction corresponding to increasing or decreasing the background noise in an utterance. Given several utterances only varying in degrees of noise, we can project them to the W-space and compute the direction in which to move to change the noise level. Concretely, to denoise an utterance X_0, we can generate N additional utterances by adding increasingly more Gaussian noise, providing a list of utterances $X_0, X_1, ..., X_N$, with $X_n = X_0 + \mathcal{N}(\mathbf{0}, n\sigma^2\mathbf{I})$. We then project each utterance to the latent space, yielding $\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_N$. To get a single vector corresponding to the direction of decreasing noise, we compute the average unit vector from the higher-noise vectors to the original latent vector:

$$\boldsymbol{\delta} = \frac{1}{N} \sum_{n=1}^{N} \frac{\mathbf{w}_0 - \mathbf{w}_n}{||\mathbf{w}_0 - \mathbf{w}_n||_2}$$

Now we can denoise the original utterance X_0 by moving in the direction of $\boldsymbol{\delta}$ in the latent space. In Section 3.7 we evaluate this method and investigate how far we can move in the $\boldsymbol{\delta}$-direction.

Speaker verification and keyword classification

The previous downstream tasks were generative in nature. The W latent space also allows us to perform discriminative tasks such as speaker verification and keyword classification: determining which speaker or word is present in a given utterance, respectively. For both tasks we use the linear nature of disentanglement: given enrollment utterances containing labelled speakers (for speaker verification) or words (for keyword classification), we invert them to their $\mathbf{w}$ vectors. Then we find the linear projection within the W latent space that maximizes the separation of the labeled characteristic using linear discriminant analysis (LDA). For inference on new data, we invert the input, project it along the LDA axes, and make a decision based on linear distances to other points in the LDA-projected latent space.

In speaker verification, we need to predict a score corresponding to whether two utterances are spoken by the same speaker [24], where the speaker of both utterances are both unseen during training. So, we project all enrollment and test utterances to the W latent space and then to the LDA axes (with axes fit on training data to maximize speaker separation). The speaker similarity score is then computed as the cosine distance between two vectors in this space. For evaluation, these scores can be used to compute an equal error rate (EER) [24]. Similarly, for keyword classification, we project everything to the LDA axes maximizing content (i.e. what digit is spoken), compute the centroids of points associated with each digit (since we know the word labels beforehand), and then assign new utterances and their corresponding projected points to the label of the closest centroid by cosine distance. We compare our latent-space LDA-based speaker verification and keyword classification approaches to task-specific models in Section 3.7.

3.6. Experimental setup: unseen tasks

3.6.1. Evaluation metrics

To evaluate ASGAN's performance in each unseen task, we use the standard objective metrics from the literature:

- *Voice conversion*: We measure conversion intelligibility following [39, 102], whereby we perform voice conversion and then apply a speech recognition system to the output and compute a character error rate (CER) and F_1 classification score to the word spoken in the original utterance. Speaker similarity is measured as described in [102] whereby we find similarity scores between real/generated utterance pairs using a trained speaker embedding model, and then compute an EER with real/generated scores assigned a label of 0 and real/real pair scores assigned a label of 1. The optimal value of this metric is 50%, indicating that a speaker embedding model is no better than random in distinguishing between the speaker of a generated utterance and that of a ground truth utterance.
- *Speech enhancement*: Given a series of original clean and noisy utterances, and the models' denoised output, we compute standard measures of denoising performance: narrow-band perceptual evaluation of speech quality (PESQ) [103] and short term objective intelligibility (STOI) scores [104].
- *Speaker verification*: Using the similarity scores on randomly sampled pairs of utterances from matching and non-matching speakers, we compute an EER as the measure of performance. We pair each evaluation utterance with another utterance from the same or different speaker with equal probability.
- *Keyword classification*: We use the standard classification metrics of accuracy and F_1 score.

3.6.2. Baseline systems

For each downstream task, we compare ASGAN to a strong task-specific baseline trained on the SC09 dataset. For voice conversion, we compare against AutoVC [38] – a well-known voice conversion model. Specifically, we use the model and training setup defined in [38] and trained on the SC09 dataset[4]. For speech enhancement, we compare to

[4] https://github.com/RF5/simple-autovc

MANNER [105], a recent high-performing speech enhancement model operating in the time-domain. Since we have no paired clean/noisy utterances for the SC09 dataset, we follow the technique from [106] to construct a speech enhancement dataset. Concretely, we assume each utterance in the SC09 dataset is a clean utterance and randomly add additional Gaussian noise to each utterance corresponding to a signal-to-noise ratio of 0 to 10 dB, sampled uniformly. We then train MANNER using the settings and code provided by the original authors[5] on this constructed dataset. For speaker verification, we use an x-vector model from [34], trained using the code and optimization settings from our open-source implementation[6] on the SC09 training set, taking the final checkpoint with best validation performance. Finally, for keyword classification we use the same ResNeXT classifier described in Section 3.3.2.

3.7. Results: unseen tasks

We apply ASGAN to a range of tasks that it didn't see during training, comparing it to established baselines. Note, however, that the goal isn't to outperform these tailored systems on every task, but rather to show that a single model, ASGAN, can generalize to a range of tasks that it was never trained on due to its disentangled latent space.

3.7.1. Voice conversion

Following Section 3.5.2, we use ASGAN for voice conversion. We compare its performance to that of AutoVC (Section 3.6.2). To test the models, we sample reference utterance from a different target speaker for each utterance in the SC09 test set, yielding 4107 utterance pairs on which we perform inference. Through initial validation experiments, we found that the optimal interpolation amount for the *fine styles* was to set the input to the later `Style Blocks` as $\mathbf{w}_1 + 1.75(\mathbf{w}_2 - \mathbf{w}_1)$ for $\mathbf{w}_1$ from the source utterance and $\mathbf{w}_2$ from the reference.

The results are shown in Table 3.4. In terms of similarity to the target speaker, AutoVC is superior to ASGAN, while ASGAN is superior in terms of intelligibility. Recall that our goal is not to outperform specialized models trained for specific tasks, but to demonstrate that ASGAN can generalize to diverse tasks unseen during training. While not superior to AutoVC, the scores in Table 3.4 are competitive in voice conversion.

To give better intuition to the linear nature of the latent space, we show an example of interpolating the *fine styles* smoothly from one speaker to another in Figure 3.2: we project the latent $\mathbf{w}$ points from many utterances in the test set using LDA fit on the speaker label. The figure raises two interesting observations. First, we see that speaker identity is

[5]https://github.com/winddori2002/MANNER
[6]https://github.com/RF5/simple-speaker-embedding

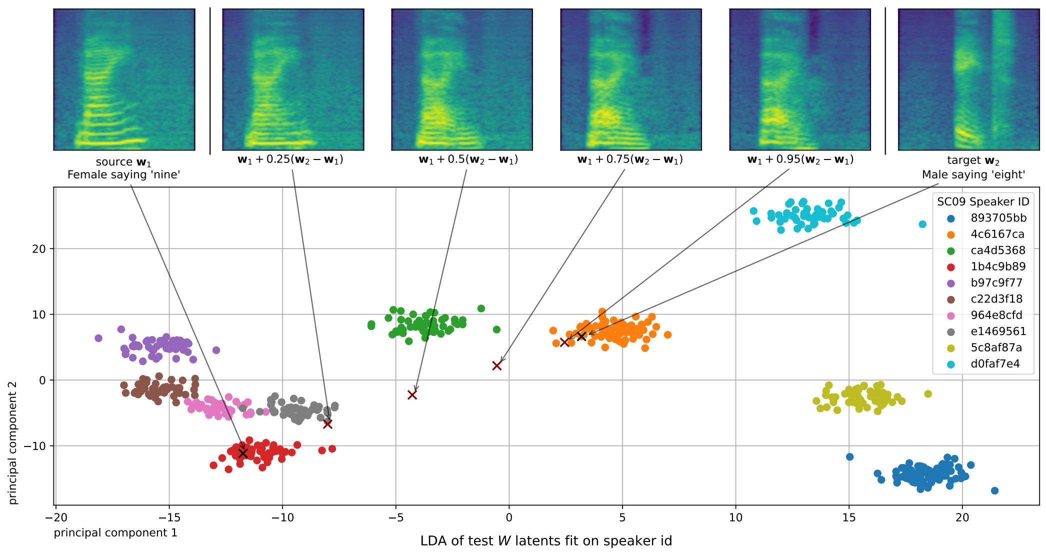


Figure 3.2: Voice conversion interpolation in the $W-$latent space. Given a source (top left) and target utterance (top right), we smoothly convert the speaker from the source to the reference by linearly interpolating the projected $\mathbf{w}$ vectors from $\mathbf{w}_1$ (source) to $\mathbf{w}_2$ (target) for use in the fine styles. The W-space interpolation is illustrated as a 2D linear discriminant analysis (LDA) decomposition of the SC09 test set.

Table 3.4: Unseen generative task performance of ASGAN compared to task-specific systems (AutoVC [38] for voice conversion, MANNER [105] for speech enhancement).

	Voice conversion (%)			Speech enhancement	
Model	EER ↑	CER ↓	F1 ↑	PESQ ↑	STOI ↑
Task-specific baseline	**32.3**	64.8	29.1	**2.13**	**81.4**
ASGAN (HuBERT)	29.7	**37.6**	**61.3**	0.95	21.2

largely a linear subspace in the latent space because of the strong LDA clustering observed. Second, as we interpolate from one point to another in the latent space, the intermediary points are still intelligible and depict realistic mixing of the source and target speakers, while leaving the content unchanged. We encourage the reader to listen to audio samples at https://rf5.github.io/slt2022-asgan-demo.

3.7.2. Speech enhancement

Following Section 3.5.2, we use ASGAN for speech enhancement, with $N = 10$, $\sigma^2 = 0.001$. Using the process described in Section 3.6.2, we evaluate both ASGAN and the task-specific MANNER system on the SC09 test set in Table 3.4. From the results we see that ASGAN performs worse than MANNER. As was the case in voice conversion, this isn't unexpected, given that MANNER is a specialized task-specific model, trained specifically for speech enhancement. Meanwhile, ASGAN is only trained to accurately model the density of utterances in the SC09 dataset. The performance of ASGAN on this task indicates some

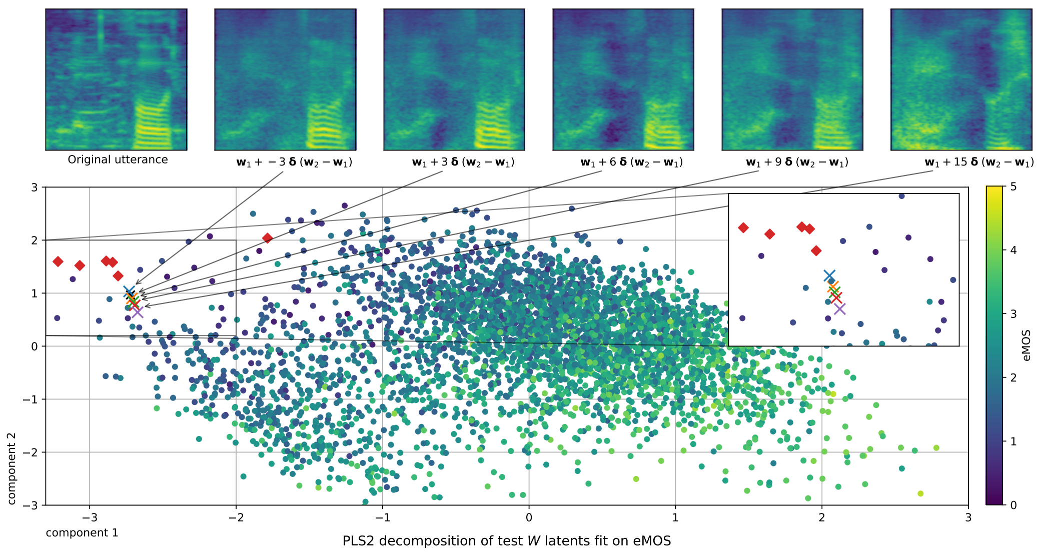


Figure 3.3: Speech enhancement in the W-latent space. Given an input utterance (top left) we add noise to it several times and find average direction of decreasing noise. Denoising or increasing noise is then performed by traversing in the latent space by varying amounts in the denoising direction $\boldsymbol{\delta}$ (top). The W-space interpolation is illustrated as a 2D partial least-squares regression (PLS2) of all the latent points from the SC09 test set fit on eMOS value as a indication of noise level. Latent variables corresponding to added noisy utterances used to estimate $\boldsymbol{\delta}$ are shown in red (♦).

level of denoising. We suspect some of the performance limitations are due to the relatively simple inversion method of Section 3.5.1, which allows for only a crude approximation of the ideal vector $\mathbf{w}$ corresponding to a speech utterance X. Using a more sophisticated latent projection method such as pivot tuning inversion [107] would likely reduce the discrepancy between $\tilde{X}$ and X. Our goal here, however, was simply to illustrate ASGAN's denoising capability without any additional fine-tuning of ASGAN's generator.

In a qualitative analysis, the denoising process is graphically illustrated in Figure 3.3, where we plot the original utterance together with interpolations by subtracting multiples of $\boldsymbol{\delta}$ (increasing noise) or adding multiples of $\boldsymbol{\delta}$ (decreasing noise). Alongside this, we find a 2D decomposition using partial least-squares regression (PSL2) fit on eMOS of utterances in the test set as a proxy for amount of noise present. The key aspect of Figure 3.3 is that we can see our motivation validated: the utterances constructed by adding more Gaussian noise (shown as ♦ in Figure 3.3) broadly lie in the direction of decreasing eMOS in the PSL2 projection. So, when computing $\boldsymbol{\delta}$ from Section 3.5 and moving in this direction, in the 2D projection it corresponds to moving roughly in the direction of increasing eMOS. And, from the associated spectrograms in Figure 3.3, we see that we can increase and decrease noise to a moderate degree without significantly distorting the content of the utterance. A caveat of this method is that we cannot keep moving in the $\boldsymbol{\delta}$ direction indefinitely – beyond $9 \times \boldsymbol{\delta}$ we start to observe increasing distortion to the speaker and content of the utterance (remember the 2D plot in Figure 3.3 contains *all* SC09 test latent

Table 3.5: Unseen discriminative task performance (%) of ASGAN compared to task-specific systems (GE2E RNN [34] for speaker verification, ResNeXT classifier [97] for keyword classification).

	Speaker verification	Keyword classification	
Model	EER ↓	Accuracy ↑	F1 ↑
Task-specific baseline	**8.43**	**98.20**	**98.20**
ASGAN (HuBERT)	41.54	72.58	72.78
ASGAN (HuBERT) + LDA	30.96	90.82	90.82

points). So, for the metrics computed in Table 3.4 we fix the denoising operation as interpolating with $9 \times \boldsymbol{\delta}$.

3.7.3. Speaker verification

We next consider a discriminative task: speaker verification, as outlined in Section 3.5.2. Using the evaluation protocol of Section 3.3.2, we obtain the results in Table 3.5 on the SC09 test set. The first ASGAN entry performs the naive cosine comparison, while the '+ LDA' entry compares distances after first applying LDA as described in Section 3.5.2. As with the generative tasks above, we see that ASGAN is competitive, but not superior to the task-specific GE2E model [34]. The naive comparison in the regular latent space achieves fairly poor results. However, comparing the naive comparison method to the LDA projection approach, we see an over 10% EER improvement, with ASGAN achieving an EER of 30.96% despite never seeing any of the speakers in the test set. This reinforces the observations from Figure 3.2, where we can see that the LDA projection of latent $\mathbf{w}$ vectors disentangles speaker identity from other aspects of speech. And, because LDA is a linear operation, we have not discarded any content information. The speaker verification performance could likely be improved if a deep model was trained to cluster speakers in the latent space. But, again, our goal is not state-of-the-art performance on this task, but to show that ASGAN can achieve reasonable accuracy on a task for which it was not trained—owing to its linearly disentangled latent space.

3.7.4. Keyword classification

Finally we apply ASGAN to keyword classification. We fit LDA on digit spoken using the SC09 validation set as enrollment, and compute predictions following Section 3.5.2. The results compared to the ResNeXT classifier are given in Table 3.5 using both the naive and LDA comparison method. As before, we see a large performance jump when using the LDA projection, with the best ASGAN method only performing roughly 7% worse than the specialized classification model. A graphical illustration of how word classes are disentangled through LDA is given in Figure 3.4. We show both the LDA projection of

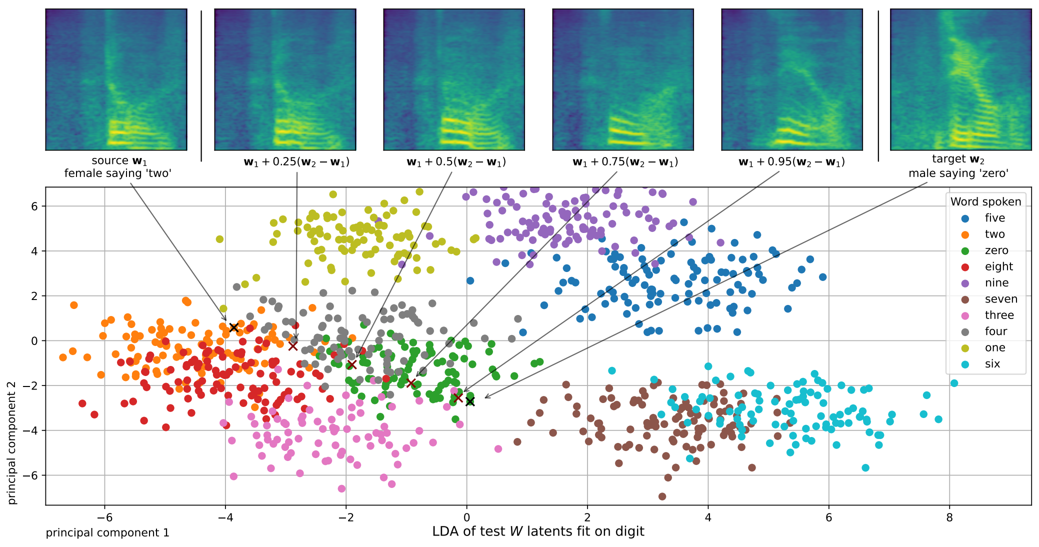


Figure 3.4: Keyword classification and speech editing in the W-latent space. Given a source (top left) and target utterance (top right), we smoothly convert the content from the source to the reference by linearly interpolating the projected $\mathbf{w}$ vectors from $\mathbf{w}_1$ (source) to $\mathbf{w}_2$ (target) for use in the coarse styles. The W-space interpolation is illustrated as a 2D LDA decomposition on the SC09 test set fit on spoken digits, showing its usefulness for keyword classification.

latent points (colored by their ground truth labels) and also how this disentanglement can be used to perform speech editing using the *coarse styles* described in Section 3.5.2, where we replace the word spoken but leave the speaker identity and other characteristics of the utterance intact. It allows us to smoothly interpolate between one spoken word and another. This could potentially be useful, e.g., for perceptual speech experiments. From both the LDA plot in Figure 3.4 and Table 3.5, we again observe strong evidence that our design for disentanglement is successful.

3.8. Summary and conclusion

This chapter aimed to answer the question of 'why can we not simply apply disentanglement methods from the image domain to speech?' (Section 1.3). To this end, we introduced ASGAN, a model for unconditional speech synthesis designed to learn a disentangled latent space. We adapted existing techniques from the image domain and incorporated new GAN design and training methods to enable ASGAN to learn a continuous, linearly disentangled latent space in order to outperform previous autoregressive and diffusion models. Experiments on the SC09 dataset demonstrated that ASGAN outperforms previous state-of-the-art models on most unconditional speech synthesis metrics, while also being substantially faster.

But, the reason for asking our main research question relates to disentanglement: from our motivation in Section 1.1, we want to know whether this better-performing model now also leads to better *generalization.* So, in further experiments, we tested how well ASGAN generalized to four unseen speech processing tasks. Namely, we demonstrated the benefit of ASGAN's disentangled latent space – it can, without any additional training, perform these unseen tasks in a zero-shot fashion through purely linear operations in its latent space, showing reasonable performance in voice conversion, speech enhancement, speaker verification, and keyword classification. These results taken together allow us to answer our research question in the affirmative: we *can* apply image techniques to improve speech disentanglement, and it *does* lead to improved generalization capabilities. Further, relating this to our overall research aim, this improvement was achieved using purely continuous methods to learn an implicit disentanglement of speech, and did not require any explicit factorization of speech when designing the model.

However, these benefits are not without drawbacks. One major limitation of our work is scale: once trained, ASGAN can only generate utterances of a fixed length. When applied to datasets with longer utterances ASGAN still struggles to generate full, coherent sentences. This is a limitation shared by existing unconditional synthesis models [19–21], but not by models following the discrete GSLM framework (Section 2.4.2). So, while the continuous methods devised here might aid in control and generalization, discrete methods like GSLM still scale to more diverse data in less restricted settings. Another, less fundamental, limitation in this investigation has been our method for applying the model to unseen tasks.

A crucial step in our way of phrasing the downstream tasks is the projection of an utterance to the model's latent space. The conversion process used in our investigation here it is still fairly crude, being a naive noisy optimization of a latent vector with the utterance fixed, leading to imperfect projections. This process could likely be improved by making use of some of the more recent techniques. E.g. [107] recently proposed an improved method for this projection, which might aid ASGAN's downstream performance.

Chapter 4

Voice conversion

Publications

This chapter is adapted from our publication:

Published conference paper: M. Baas, B. van Niekerk, H. Kamper, "Voice Conversion With Just Nearest Neighbors," in *Proceedings of Interspeech*, 2023.

In this chapter we ask the question posted in Section 1.3: why have we not seen speech synthesis models exploiting implicitly disentangled representations to achieve the benefits posited in our motivation? Recall, the motivation of this thesis is that systems designed to learn implicit disentanglements of speech should yield improved control and generalization compared to those focused on explicit disentanglement. But, several key speech processing tasks are still dominated by these explicit methods. Hence our question for this chapter: why have we not seen implicit methods yield superior results? For this investigation we specifically look at the task of any-to-any voice conversion, where we aim to transform source speech into a target voice with just a few examples of the target speaker as a reference (Section 2.4.1).

The reason for choosing this task, in addition to its difficulty, is because it provides a good case study of whether disentangled representations can make the complexity of domain-expert methods unnecessary. Recent voice conversion methods produce convincing conversions, but at the cost of increased complexity – making results difficult to reproduce and build on. But, we know from the literature around SSL models (Section 2.5.1) that we already have models that can convert speech into disentangled representations. So, linear operations in the SSL latent space should be able to distinguish between speaker and content, making voice conversion possible with just linear operations in the latent space, without any explicit conversion model. If this reasoning holds, it means that the SSL features studied for speech representation can also be used for speech *synthesis* tasks, and we could answer our research question 2 in the affirmative – using implicitly disentangled features does improve performance in certain speech processing tasks.

To see if does hold, we develop k-nearest neighbors voice conversion (kNN-VC): a straightforward yet effective method for any-to-any conversion. The method is simple: first, we extract SSL representations of the source and reference speech. To convert to the target speaker, we replace each frame of the source representation with its nearest neighbor (in the latent space) from the reference. Finally, a pretrained vocoder synthesizes audio from the converted representation. We then subject kNN-VC to an array of objective and subjective evaluations against existing explicit-disentanglement-based voice conversion models, where we show that kNN-VC improves speaker similarity and retains similar intelligibility compared to existing methods.

Our experiments indicate that (a) we *can* use SSL models to perform voice conversion without an explicit conversion model[1], and (b) it *does* lead to superior performance compared to existing conversion methods based on explicit disentanglement.

This chapter has associated code, trained models, and audio samples, which we highly encourage the reader to listen to or experiment with their own voices: `https://bshall.github.io/knn-vc`. Our main contribution in this chapter is to show that complexity is not necessary for any-to-any voice conversion – just applying nearest neighbors to implicitly disentangled, continuous SSL features gives convincing, if not superior, results.

4.1. Related work

While we give a historical overview of voice conversion in Section 2.4.1, here we give a brief overview of the current state-of-the-art systems to which we compare. Typical voice conversion systems try to separate speaker identity from content. The speaker information can then be replaced to convert to a target voice. However, as we saw in the last chapter, learning disentangled representations jointly with the synthesis task is challenging [32]. Recent systems require complex information bottlenecks [30, 38, 108], normalization methods [109], or data augmentation [30], but still struggle with speaker similarity. Alternatively, some methods use text annotations as an intermediate information bottleneck to improve the explicit disentanglement [31, 110]. However, transcribing large speech corpora is costly and time-consuming. Additionally, the bottleneck imposed by text can remove prosody information which is beneficial in voice conversion applications. We therefore focus on text-free methods.

Since our proposed kNN-VC constructs the output SSL feature sequence by simply replacing each source feature with its nearest neighbor in the reference, it can be seen as a form of concatenative voice conversion. The idea in concatenative voice conversion [111,112] is to stitch together units of target speech that match the source content. By constructing the output exclusively from target speech, concatenative methods ensure good speaker similarity. A key component in concatenative systems is unit selection, which generally requires parallel data [112] or time-aligned transcriptions [113]. Our method can also be seen as a unit selection approach. But instead of using explicit human-defined units, we use implicitly disentangled features from SSL models. The key idea is that, with SSL

[1]Arguably the linear distance nearest neighbor operation is our conversion model, but kNN is a non-parametric algorithm, not requiring training or weights, making it substantially simpler than existing models.

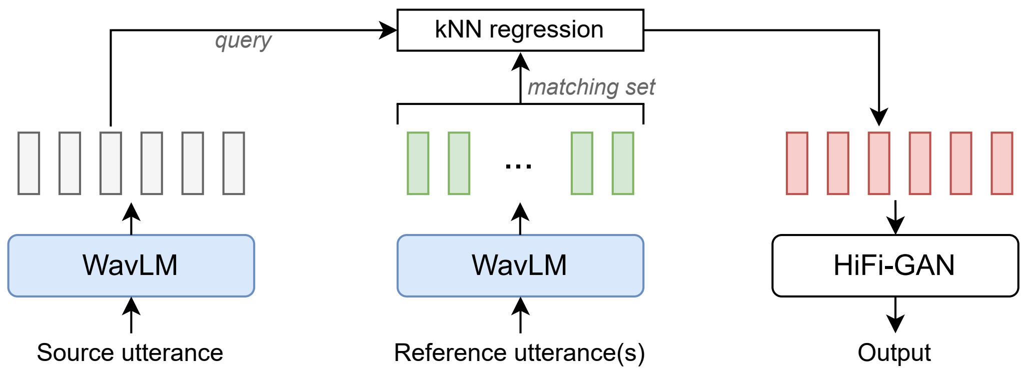


Figure 4.1: kNN-VC: The source and reference utterance(s) are encoded into self-supervised features from a pretrained WavLM model [23]. Each source feature is assigned to the mean of the k closest features from the reference. The resulting feature sequence is then vocoded using HiFi-GAN to arrive at the converted waveform output.

features, linear distances are indicative of differences in high-level speech characteristics. Concretely, recent studies have shown that SSL representations linearly predict many properties of speech [114, 115]. Specifically, for certain SSL features, nearby (in terms of cosine distance) features indicate shared phonetic content [116]. This leads us to our key idea for kNN-VC: linearly finding the nearest feature from the reference finds the target speaker feature with the most similar phonetic content. By replacing all source features in this way, we have kept phonetic content largely similar, but replaced the speaker identity and other characteristics of speech to be that from the supplied reference.

4.2. kNN-VC

We propose k-nearest neighbors voice conversion (kNN-VC) in Figure 4.1. It follows an encoder-converter-vocoder structure [117]. First, the encoder extracts self-supervised representations of the source and reference speech. Next, the converter maps each source frame to its nearest neighbor in the reference. Finally, the vocoder generates an audio waveform from the converted features. Each of these components is described below.

4.2.1. Encoder

kNN-VC starts by extracting the feature sequence of the source utterance, which we call the *query sequence* (Figure 4.1). We also extract the feature sequences of one or more utterances from the target speaker, which get shuffled together into a large pool of self-supervised feature vectors. We call this bag-of-vectors the *matching set*. The goal of this encoder is to extract representations where nearby features have similar phonetic content. Recent self-supervised models [22, 23] are good candidates since they score well on phone discrimination tests, i.e., they encode instances of the same phone closer to each other than to different phones [116]. The encoder model is never fine-tuned or trained further for kNN-VC – it is only used to extract features.

4.2.2. k-nearest neighbors matching

To convert to the target speaker, we apply k-nearest neighbors (kNN) regression to every vector in the query sequence. Specifically, we replace each query frame with the average of its k-nearest neighbors in the matching set. Since similar self-supervised speech features share phonetic information [23, 115], performing kNN preserves the content of the source speech while converting speaker identity. Similarly to concatenative methods, we construct the converted query directly from target features, guaranteeing good speaker similarity. The kNN regression algorithm [118] is also non-parametric and requires no training, making this method easy to implement.

4.2.3. Vocoder

The vocoder translates the converted features into an audio waveform. Instead of conditioning on spectrograms, we adapt a conventional vocoder to take self-supervised features as input. However, there is a mismatch between inputs during training and inference. For inference, we condition the vocoder on kNN outputs, i.e., the mean of features selected from the matching set. We pick these features from various points in time with different phonetic contexts, leading to inconsistencies between adjacent frames. Section 4.4.2 shows that these inconsistencies cause artifacts, reducing the intelligibility of converted speech.

4.2.4. Prematched vocoder training

To address this issue, we propose *prematched training.* This prematching process is shown in Figure 4.2. The idea is to improve robustness by training the vocoder on data resembling what it encounters during inference. Specifically, we reconstruct the vocoder training set using k-nearest neighbors. Using each training utterance as a query, we build a matching set using the remaining utterances from the same speaker. Then we apply kNN regression to reconstruct the query sequence using the matching set. We train the vocoder to predict the original waveforms from these prematched features. In Section 4.4.2, we show that prematched vocoder training improves intelligibility and speaker similarity.

4.3. Experimental setup

To determine whether our new non-parametric voice conversion method can compete with recent neural network conversion models, we run two experiments on the LibriSpeech development and test sets. The first experiment compares kNN-VC to state-of-the-art voice conversion systems. The second investigates the effect of target data size and prematched

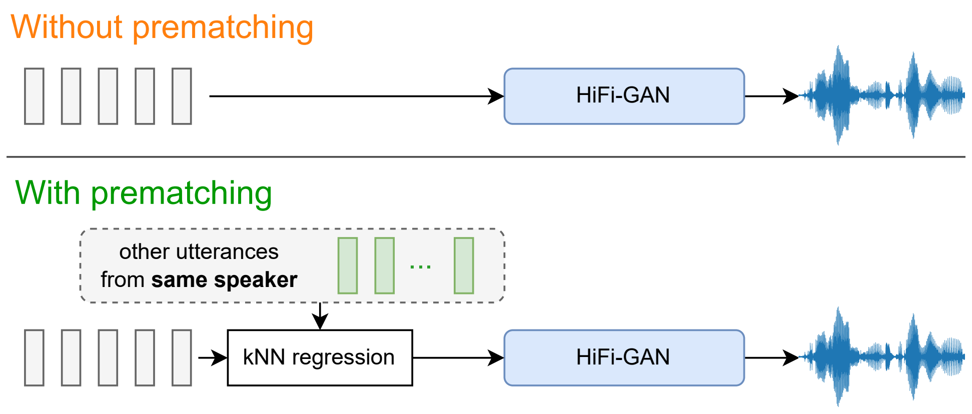


Figure 4.2: Process of training vocoder without prematching (top) and with prematching (bottom). Without prematching, training SSL features do not contain artifacts present at inference from the kNN operation. With prematching, vocoder training samples are mapped through a kNN regression to other utterances of the same speaker before input into the vocoder, better mimicking inference conditions.

vocoder training. For both experiments, the source and target speakers are unseen during training. The LibriSpeech development and test clean sets consist of 40 speakers apiece, each contributing about 8 minutes of 16 kHz spoken English [74]. The diversity of speakers and audio quality in LibriSpeech make this a challenging benchmark for any-to-any voice conversion.

4.3.1. kNN-VC implementation

Encoder

We use the pretrained WavLM-Large encoder from [23] to extract features for the source and reference utterances. In preliminary experiments, we used features from later layers (22, 24, and the mean of the last several layers), which perform well on linear phone recognition tasks [23]. The idea was to improve nearest neighbors mapping by including more content information. However, this led to worse pitch and energy reconstruction. Recent work [115] confirms that later layers give poorer predictions of pitch, prosody, and speaker identity. Based on these observations, we found that using a layer with high correlation with speaker identification – layer 6 in WavLM-Large – was necessary for good speaker similarity and retention of the prosody information from the source utterance. So, for all the following experiments, we use features extracted from layer 6 of WavLM-Large, which produces a single vector for every 20 ms of 16 kHz audio. Note here that, while layer 6 has high correlation with speaker identity (i.e. the earlier layers in the SSL model still retain more pitch and speaker identity information), linear distance comparisons still are primarily indicative of phonetic differences. If this were not the case, our nearest neighbor method would not retain the same linguistic content.

kNN regression

For all our experiments, we set $k = 4$ with a uniform weighting and use cosine distance to compare features. In preliminary experiments, we found that kNN-VC is quite robust to a range of values around $k = 4$. Namely, when more reference audio is available (e.g. $\geq$10 mins), the conversion quality may even be improved by using larger values of k (in the order of $k = 20$). Similarly, if relatively little data is available ($\leq$30 sec), smaller values of k may yield the best results.

Vocoder

We train the HiFi-GAN V1 architecture from [53]. We modify it to accept the 1024-dimensional input vectors from WavLM and to vocode 16 kHz audio using 128-dimensional mel-spectrograms with a 10 ms hop length and 64 ms Hann window. We train the model using the same optimizer, steps, and other hyperparameters as [53] on the LibriSpeech train-clean-100 dataset. We train two variants: one trained on pure WavLM-Large layer 6 features, and one trained on prematched layer 6 features (as explained in Section 4.2.4). With these design choices, inference with 8 minutes of reference audio on a consumer 8GB VRAM GPU is faster than real-time.

4.3.2. Baselines

We compare kNN-VC to three any-to-any voice conversion systems: VQMIVC[2], FreeVC[3], and YourTTS[4] (in text-free voice-conversion mode). VQMIVC uses vector quantization with mutual information minimization to disentangle speaker identity from content [108]. FreeVC combines a variational autoencoder with data augmentation to discard speaker information [30]. YourTTS uses text transcriptions to create an intermediate information bottleneck during training [31]. To convert to the target speaker, all three models are conditioned on speaker embeddings extracted from the reference speech using a speaker encoder. During inference, we average the speaker embeddings across all utterances from the target speaker to maximize performance. We use the publicly available, pretrained models provided by the respective authors as the baselines.

4.3.3. Evaluation metrics

We assess naturalness, intelligibility, and speaker similarity in subjective and objective evaluations. We report results on the LibriSpeech test-clean subset which consists of unseen speakers not contained in any other subset of LibriSpeech.

[2]https://github.com/Wendison/VQMIVC
[3]https://github.com/OlaWod/FreeVC
[4]https://github.com/Edresson/YourTTS

Objective evaluations

To construct an evaluation set, we sample 200 utterances (5 per speaker) from the LibriSpeech test-clean set. We convert each utterance to the 39 other speakers, giving a total of 7800 outputs per model. Following previous studies [32, 108, 110], we evaluate intelligibility by calculating the word/character error rate (W/CER) of an ASR system applied to the converted speech. Lower error rates indicate better intelligibility. We use the trained Whisper-base ASR model [59] with default decoding parameters to transcribe the converted utterances.

To measure speaker similarity, we follow [102] and calculate an equal error rate (EER) using a trained speaker verification system [119]. Given an input utterance, the verification system outputs a x-vector representing the speaker identity. We compute the cosine similarity between x-vectors to determine whether two utterances are from the same speaker. First, we record similarity scores for each converted sample paired with an enrollment utterance from the target speaker. Then, we obtain an equal number of 'ground-truth' similarity scores by sampling target speaker utterances and computing the similarity to a different enrollment target speaker utterance. Finally, we calculate an EER over the combined set of scores, assigning a label of 1 to the ground-truth pairs and 0 to pairs containing converted speech. A higher EER indicates that it is more difficult to distinguish converted speech from genuine examples of the target speaker, i.e., higher EER corresponds to better speaker similarity, with a maximum of 50%.

Subjective evaluations

Using Amazon Mechanical Turk, we run subjective evaluations to measure naturalness and speaker similarity. For naturalness, we report mean opinion scores (MOS) on a 1-to-5 scale. We sample and evaluate 60 utterances for each model, adding an equal number of ground-truth examples from the LibriSpeech test-clean set. We filter out ratings from listeners who judge the ground-truth examples as very unnatural (a MOS of 1), resulting in a total of 2144 ratings from 61 listeners.

For speaker similarity, we follow the protocol from the Voice Conversion Challenge 2020 [32]: given a pair of utterances, we ask listeners to rate the similarity between the speakers on a 1-to-4 scale. For the evaluation, we sample 20 source utterances from the LibriSpeech test-clean set. We convert each utterance to three target speakers using each model, resulting in 60 examples per model. We ask raters to evaluate the similarity between the converted outputs and a genuine example of the target speaker. We again filter out raters that judge the ground-truth target speaker pair as very different (a SIM of 1), resulting in a total of 1636 ratings over 208 listeners. Based on these ratings, we report a mean similarity (SIM) score for each model.

4.4. Results

We report the results of the comparative experiment in Section 4.4.1, followed by the ablation experiment in Section 4.4.2. In the first experiment, we test the voice conversion systems using the maximum available target data (about 8 minutes of audio per speaker). For kNN-VC, we use all the target data as our matching set. For the baselines, we average the speaker embeddings of each target utterance. The second experiment evaluates kNN-VC as we vary the target data size. We also investigate how prematched training affects intelligibility and speaker similarity.

4.4.1. Voice conversion

Table 4.1 reports intelligibility, naturalness, and speaker similarity results for each model. For kNN-VC, we use the HiFi-GAN trained on prematched data. We observe that kNN-VC achieves similar naturalness (MOS) and intelligibility (W/CER) to the best baseline, FreeVC. However, kNN-VC significantly improves speaker similarity (EER and SIM). Given the simplicity of kNN-VC, these results confirm our research hypothesis: increased complexity is not necessary for high-quality voice conversion.

A stronger conclusion is that our simple kNN method, combined with disentangled speech features, leads to improved speaker similarity compared to existing models. A major benefit of kNN-VC is that it does not require an explicit speaker embedding model, in contrast to all the baselines we compare to here.

Because we do not rely on a speaker embedding model, an interesting question is: how far away can the reference utterances be from the training distribution? We leave a full investigation to Chapter 6, but include conversions to unseen languages, whispered speech, and even non-speech sounds on the demo page for this chapter: https://bshall.github.io/knn-vc. For example, in our cross-lingual demo, we convert German source speech to a Japanese target speaker. The conversion is intelligible despite the system having only seen English during its design and training.

Table 4.1: Results measuring the intelligibility (W/CER), naturalness (MOS), and speaker similarity (EER, SIM) on the LibriSpeech test-clean subset. Subjective MOS and SIM values with 95% confidence intervals are shown.

Model	WER ↓	CER ↓	EER ↑	MOS ↑	SIM ↑
Testset topline	5.96	2.38	50.00	4.24 ± 0.07	3.19 ± 0.09
VQMIVC [108]	59.46	37.55	2.22	2.70 ± 0.11	2.09 ± 0.12
YourTTS [31]	11.93	5.51	25.32	3.53 ± 0.09	2.57 ± 0.12
FreeVC [30]	7.61	3.17	8.97	$\mathbf{4.07 \pm 0.07}$	2.38 ± 0.11
kNN-VC	**7.36**	**2.96**	**37.15**	$\mathbf{4.03 \pm 0.08}$	$\mathbf{2.91 \pm 0.11}$

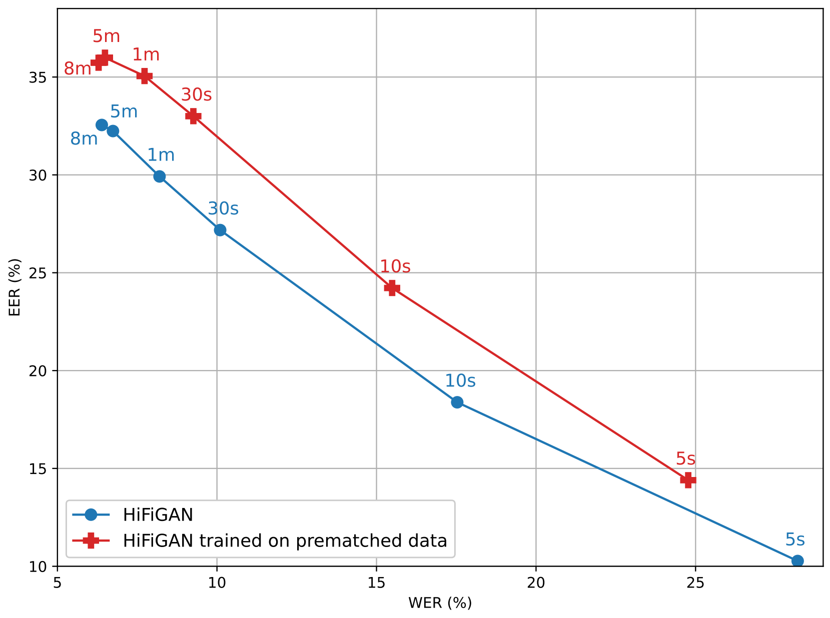


Figure 4.3: Intelligibility (WER, lower is better) vs speaker similarity (EER, higher is better) for varying amounts of target speaker data on the LibriSpeech dev-clean subset. The optimal result is to be in the upper-left corner.

4.4.2. Ablation: Prematching and amount of reference data

While the main research question is answered, we would like to know how much of the improvements come from HiFi-GAN being trained on prematched data (Section 4.2.4), and how much the quantity of target speaker data influences intelligibility and speaker similarity. To elucidate this, Figure 4.3 plots WER against EER across different target speaker data sizes for both HiFi-GAN variants.

Prematched data

To recap from Section 4.2.4, prematching works by mapping each HiFi-GAN training utterance's WavLM features to the nearest k vectors from other utterances of the same speaker (rather than just training on WavLM features directly). The higher EER and lower WER in Figure 4.3 shows that prematching gives a non-negligible improvement, regardless of the amount of target speaker data.

Amount of reference data

It is clear from Figure 4.3 that as amount of target speaker speech decreases, both the intelligibility and speaker similarity of the converted utterance deteriorates. This is to be expected since in the extreme case of e.g. 5 or 10 seconds of reference audio, it is unlikely that the matching set would cover all phones or biphones necessary for smooth conversion. We see, however, that the performance gains diminish after a certain point: using the maximum speaker data in LibriSpeech (roughly 8 minutes per speaker) gives very similar performance to only using 5 minutes of target speaker data. Again, this is likely because – beyond a certain point – there are enough vectors in the matching set to cover all possible phones and biphones present in the source utterance.

Comparison to baselines

Even with the plain HiFi-GAN vocoder not trained on prematched data, the scores that kNN-VC achieves in Figure 4.3 are still competitive with existing models in terms of intelligibility and still superior in terms of speaker similarity (comparing to the scores in Table 4.1). This demonstrates that, while training the vocoder on prematched data helps, using the simplest setup of a pretrained self-supervised speech model and vocoder together with kNN regression is still a powerful any-to-any voice conversion approach. However, with limited target data (less than 30 s), intelligibility and target speaker similarity decrease to a point where the more complex baselines perform better (Table 4.1).

4.5. Summary and conclusion

This chapter introduced kNN-VC, a general purpose any-to-any voice conversion model designed around utilizing disentangled SSL features. The goal was to answer research question 2 in Section 1.3, where we are interested in understanding why SSL features have not been used in synthesis tasks yet. We had to investigate this to reconcile it with our thesis's motivation, where we claim that using disentangled latent spaces should allow better generalization in speech synthesis tasks. Finding a lack of effective prior attempts to utilize the disentangled nature of recent SSL models for voice conversion, we proposed kNN-VC.

Unlike prior methods which relied on complex neural architectures to produce an output feature sequence when converting to the target speaker (largely based on domain-expert knowledge), we simply replace each source feature with the nearest reference feature based on distance in the implicitly disentangled SSL feature space. We then compare our simple approach based on the kNN algorithm to several recent high-performing voice conversion models in a challenging any-to-any conversion scenario, and found that we achieve similar intelligibility and naturalness scores while improving speaker similarity.

This confirms our hypothesis that voice conversion can be done with simpler methods that make use of the disentangled feature space learned by SSL speech representation models, despite these SSL models having never been trained for voice conversion or speech synthesis, but purely a masked token prediction task (Section 2.5.1). So, concretely, our experiments here have answered research question 2 in the affirmative: we *can* apply SSL models to voice conversion, and it *does* lead to improved generalization performance.

Moreover, if we compare this to our first investigation in Chapter 3, we can make an interesting observation. Here we showed that if we can learn a disentangled latent space on diverse data, as is done with SSL models, then the generalization and simplification benefits that come from designing-for-disentanglement do not come at the cost of worse scalability or difficulty of training. Here, the actual matching network is the non-parametric kNN algorithm, which is trivial to scale or fit to existing data.

Finally, we also investigated an improvement to the basic vocoding of the matched kNN features, whereby we introduced the idea of training a vocoder on prematched feature data to better approximate inference conditions, which improved performance. To better understand the limits of our approach, we looked at the effect of the amount of reference data on final voice conversion performance. We found that, with as little as five seconds of target speaker audio, we can still retain moderate intelligibility and speaker similarity.

Because the core of our approach is the vanilla kNN algorithm and our code and methods are open-source, we hope that kNN-VC can serve as a robust, easy-to-implement baseline for voice conversion.

Several interesting works building upon kNN-VC have recently been proposed by other authors where our idea shows promise for yielding improvements in related fields. For example, [120] noted how our kNN operation can distort the pitch and loudness contour of the converted utterance. To rectify this, they introduced a special signal generator model which took an additional target pitch and loudness contour as input, thereby allowing the model to better reconstruct the output. Other work by [121] noted how the need for several minutes of audio for the matching set can be a severe limitation in some situations. So, they experimented with adding a variational autoencoder to hallucinate additional matching set vectors to improve the quality of the kNN output. Aside from these two recent efforts, there continues to be interest in expanding kNN-VC to improve its performance and capabilities, likely due to its simplicity and ease of experimentation.

Chapter 5

Disentanglement for Speech Recognition

Publications

This chapter is adapted from our publication:

Published journal article: M. Baas, K. Eloff, H. Kamper, "TransFusion: Transcribing Speech with Multinomial Diffusion," *Springer Communications in Computer and Information Science*, vol. 1734, pp. 231–245, 2022.

Although the previous two chapters have focused on speech synthesis tasks, not all speech processing tasks permit tackling with purely continuous methods. Some tasks, such as automatic speech recognition (ASR), necessarily require a discrete sequence output (e.g. characters of the transcript). But, from the prior chapter, we have now seen and confirmed the benefit of applying implicitly disentangled SSL features to tasks for which the SSL models were not trained. So, our third research question (Section 1.3) arises: is there a way to exploit implicitly disentangled features and continuous methods for speech tasks with discrete outputs? This chapter tries to tackle this question by formulating ASR in a new way, combining two key modelling tools typically reserved for speech synthesis tasks – disentangled SSL features and diffusion.

The current paradigm for high-performance ASR involves the use of supervised training of large neural networks with a connectionist temporal classification (CTC) loss [26]. Intuitively, these models predict the probability of a character occurring in a particular time window within an utterance. While this method is the current state-of-the-art for ASR [23, 122], they don't make use of the strong continuous techniques we have seen in previous chapters, so the question remains whether better methods might exist.

To answer this, we attempt to formulate ASR as a conditional diffusion task. Conditioned on disentangled speech features from an SSL speech representation model (Section 2.5.1), our system attempts to iteratively denoise a random character sequence to ultimately resemble the transcript of the utterance associated with the speech features. From the prior chapters we know SSL features are highly disentangled and make it very easy for a model to learn to extract or manipulate information in the latent space. So, our model uses these SSL features as conditioning in a *multinomial diffusion* task [80] – a discrete variant of diffusion, as text is fundamentally discrete – whereby the model predicts a distribution of characters occurring at each position in a transcript. Since our model transcribes speech with a diffusion task, we dub it TransFusion. To the best of our knowledge, we are the first to apply diffusion to the task of ASR. Furthermore, the typical decoding methods used in ASR are not readily applicable to our new type of model. So, we also go on to propose initial new techniques for improving sampling of diffusion-type ASR acoustic models.

To see how well our idea fares, we compare our model to existing CTC-type models such as wav2vec 2.0 [22] on the standard LibriSpeech ASR benchmark [74]. We do not use a language model or external lexicon, as our primary interest is to understand whether our new formulation of ASR acoustic models – that now incorporates two key continuous modelling techniques – can compete with the highly developed existing CTC-type ASR models. In our evaluations, we demonstrate comparable word error rate (WER) performance to existing high-performing CTC-type models of similar size (`test-other` WER of 8.8%) despite the dearth of decoding and sampling heuristics available for our new diffusion-type ASR approach. These results lead us to make two key findings: first, the scaling properties of diffusion models in other domains *are* also present in ASR, confirming the benefit of applying diffusion (a typically continuous method) to this discrete task. Second, using implicitly disentangled SSL features as conditioning *can* benefit discrete tasks by allowing for a new formulation of tasks like ASR, being performant enough so as to compete with existing highly developed CTC models. This chapter has associated code, trained models, and usage demonstrations: `https://github.com/RF5/transfusion-asr`.

5.1. Related work and techniques

Modern high-performance ASR systems operate in the time-domain, typically using end-to-end deep neural networks to transcribe an utterance. In particular, current state-of-the-art methods such as [13, 22, 122] first use a large convolutional encoder to downsample a waveform into a vector sequence with each vector typically corresponding to 10 ms to 50 ms of audio. This sequence is then refined in a large transformer variant [58] to yield output features. These models are also typically trained in two phases: a pretraining phase with unlabeled speech, and a fine-tuning phase with labeled speech (i.e. audio where the transcript is known). The pretraining phase is typically formulated as a MLM task as discussed in Section 2.5.1, while the fine-tuning process is done with CTC, discussed next.

Meanwhile, diffusion models (Section 2.6) have been almost exclusively applied to continuous domains such as image synthesis [7, 8] and music or audio synthesis [20]. Movellan et al. (1999) [123] was the first to apply diffusion with textual data, where they attempted to classify the word spoken in short videos of people saying one of four possible words. However, the diffusion framework referenced in this work is *not* a DDPM as in Section 2.6. Rather, they define their own concept of 'diffusion networks' as a stochastic version of recurrent neural networks [123]. The second, more recent, work considering diffusion with textual data defined the key formulation for diffusion on discrete units such as text characters – aptly named multinomial diffusion [80]. While the authors of [80] show considerable performance at unconditional text synthesis, they leave the question open as to how effective such diffusion methods will be when applied to ASR, and how it might be combined with disentangled SSL features.

5.1.1. Connectionist temporal classification

The fine-tuning step of most existing end-to-end ASR systems involves the use of connectionist temporal classification (CTC). CTC is a method to model the probability of one sequence given a different, possibly unaligned sequence [26]. For speech recognition, the sequence of output features produced by the CTC model is used to model the probability of the sequence of characters (target transcript). These sequences are unaligned since each output feature from the model corresponds to a small window of time (e.g. 10 ms), while an English character may correspond to a variable period of time (e.g. 300 ms). Essentially, each item in the output sequence produced by the model parameterizes a categorical distribution over the characters in the alphabet and a special ϵ character. CTC then allows for many-to-one alignments of this output sequence to the ground-truth transcript by collapsing repeated characters and removing ϵ characters. A loss is formed as the negative log likelihood of all possible alignments between the model output and target character sequence, where dynamic programming is used to make the computation tractable [124]. So, the fine-tuning process of the current large self-supervised models such as [13, 22] involves maximizing the likelihood of the ground-truth transcript, given the model's output features.

5.1.2. Multinomial diffusion

Diffusion networks, or DDPMs, explained in Section 2.6, have recently been shown to scale very well to large model sizes and datasets [7, 8], and we hypothesize that it will yield similar beneficial properties when applied to ASR. One issue with the typical diffusion discussed before is that it is formulated in the continuous domain, such as denoising a continuous pixel or audio sample value slightly in each step of the Markov chain. For discrete signals like text, we must use a recent discrete variant of diffusion – multinomial diffusion.

In 2021, Hoogeboom et al. introduced multinomial diffusion [80] to perform diffusion on signals with discrete alphabets. It works as follows: multinomial diffusion defines the input to a diffusion model as a sequence of discrete units (e.g. letters, amino acids) represented as one-hot encoded vectors $\mathbf{x}$. We index the diffusion timestep with $t \in \{0, ..., T-1\}$ and the position in the sequence (the character index in the transcript) with $i \in \{0, ..., N-1\}$ such that $\mathbf{x}_{t,i} \in \{0,1\}^K$ is the one-hot encoding of the character represented at diffusion timestep t and sequence position i using a K-sized alphabet. However, all diffusion operations proposed by [80] are independent across sequence length, thus when we omit the index i it indicates that the statement applies to the entire sequence independent of sequence index. Multinomial diffusion defines the forward noising process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, and the reverse diffusion process $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ for a K-sized alphabet as:

$$\begin{aligned}
q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) &= \mathcal{C}\left(\mathbf{x}_t \,\middle|\, (1-\beta_t)\mathbf{x}_{t-1} + \beta_t/K\right), \\
q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0) &= \mathcal{C}\left(\mathbf{x}_{t-1} \,\middle|\, \frac{1}{A}\left[\alpha_t \mathbf{x}_t + (1-\alpha_t)/K\right] \odot \left[\bar{\alpha}_{t-1}\mathbf{x}_0 + (1-\bar{\alpha}_{t-1})/K\right]\right), \text{ and} \\
p(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t) &= \mathcal{C}\left(\mathbf{x}_{t-1} \,\middle|\, \frac{1}{A}\left[\alpha_t \mathbf{x}_t + (1-\alpha_t)/K\right] \odot \left[\bar{\alpha}_{t-1}\hat{\mathbf{x}}_0 + (1-\bar{\alpha}_{t-1})/K\right]\right),
\end{aligned}$$

where $\mathcal{C}$ denotes a categorical distribution with category probabilities specified after $|$, and $\odot$ denotes pointwise multiplication. β_t is the diffusion noise schedule defined in the original binomial diffusion work [27], and $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{\tau=0}^{t} \alpha_\tau$. The fraction $\frac{1}{A}$ is a normalizing constant to ensure the probabilities sum to one [80]. The final-timestep sequence $\mathbf{x}_0$ is the one-hot encoding derived from the ground-truth text for the posterior, and $\hat{\mathbf{x}}_0$ is the predicted probabilities over the vocabulary for each position in the sequence. This is how the diffusion process is parameterized with a neural network: at each diffusion timestep, the model predicts a distribution over the vocabulary $\hat{\mathbf{x}}_0$ of the *fully denoised transcript* at $t = 0$. Intuitively, the forward process slowly moves probability mass from the ground truth one-hot vector to (at timestep T) a uniform categorical distribution. The reverse process, then, produces probabilities that are interpolated between the current noise level and the model's predicted probabilities for $\mathbf{x}_0$. To get an idea how the reverse process iteratively denoises a sample transcript using multinomial diffusion, see Figure 5.3 (explained later). In this chapter we phrase the task of speech recognition as a speech-feature-guided multinomial diffusion task.

5.2. TransFusion model

Our model is a denoising probabilistic diffusion model [27] that transcribes utterances from a provided sequence of speech features extracted from a self-supervised speech representation model. So, as our model is a <u>trans</u>cribing dif<u>fusion</u> model, we dub it **TransFusion**. Concretely, it adapts classifier-free guidance [125] (i.e. conditioned diffusion) and multinomial diffusion [80] to allow a discrete diffusion model to be conditioned on a sequence of disentangled speech features extracted from the SSL model WavLM [23].

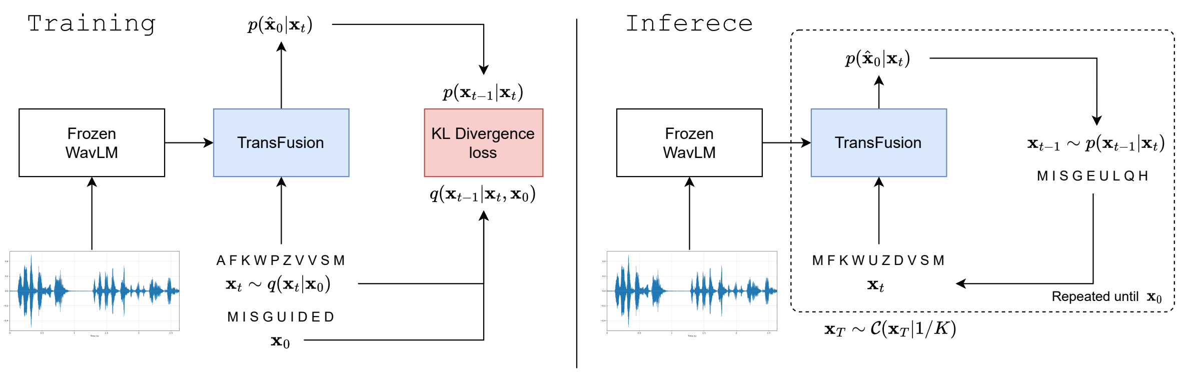

Training
Frozen WavLM
TransFusion
KL Divergence loss
$p(\hat{\mathbf{x}}_0|\mathbf{x}_t)$
$p(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$
A F K W P Z V V S M
$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$
M I S G U I D E D
$\mathbf{x}_0$
Inferece
Frozen WavLM
TransFusion
$p(\hat{\mathbf{x}}_0|\mathbf{x}_t)$
$\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t)$
M I S G E U L Q H
M F K W U Z D V S M
$\mathbf{x}_t$
Repeated until $\mathbf{x}_0$
$\mathbf{x}_T \sim \mathcal{C}(\mathbf{x}_T|1/K)$

Figure 5.1: TransFusion diagram. Speech features from an utterance computed using a frozen WavLM encoder [23] are used to condition TransFusion. During training (left) the model is trained according to multinomial diffusion [80] to minimize the KL divergence between the reverse process $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and posterior process derived from the ground truth utterance $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. During inference (right), a uniformly random sampled transcript $\mathbf{x}_T$ is iteratively denoised using TransFusion until $\mathbf{x}_0$ is the predicted output transcript.

5.2.1. Conditioning diffusion on speech representations

The training and inference setup of TransFusion is shown in Figure 5.1. During training, we have an input utterance waveform and its associated ground-truth transcript denoted $\mathbf{x}_0$. During each training step, a noised version of the transcript is calculated for diffusion timestep t using $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{C}\left(\mathbf{x}_t \,|\, \bar{\alpha}_t \mathbf{x}_0 + (1-\bar{\alpha}_t)/K\right)$ [80], where t is sampled uniformly at random from $\{0, ..., T-1\}$. Intuitively, the input text fed to the model has its characters randomly flipped, with increasing randomness until at the highest timestep $t = T-1$, the transcript fed to the model is entirely random. The waveform is converted into a sequence of disentangled speech features $\mathbf{c}$ using a fixed pretrained SSL model WavLM [23]. This sequence of features $\mathbf{c}$ is then used to condition the main TransFusion model. TransFusion's architecture is that of a transformer variant (discussed later) and maps the noisy input characters to a predicted distribution of output characters for the desired transcript $\mathbf{x}_0$. More formally,

$$p(\hat{\mathbf{x}}_0|\mathbf{x}_t, \mathbf{c}) = \text{TransFusion}(\mathbf{x}_t, \mathbf{c})$$

During inference (Figure 5.1, right), given speech features $\mathbf{c}$ from an utterance with unknown transcript, we sample a random sequence of characters at $\mathbf{x}_T$. We then iteratively denoise the transcript by using TransFusion and the diffusion parameters to compute a distribution for the reverse process $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The slightly denoised transcript is then sampled from this distribution as $\mathbf{x}_{t-1}$ and used as the input to the model in the next iteration. This process continues until $t = 0$ and $\mathbf{x}_0$ is a refined prediction of the transcript of the utterance.

5.2.2. Training task

The loss follows that of multinomial diffusion (Section 5.1.2). Specifically, using the diffusion parameters β, α, $\bar{\alpha}$ at timestep t and the current noised inputs, the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is computed. Likewise, with the model's prediction and the diffusion parameters, the reverse distribution $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ also provides a distribution over $\mathbf{x}_{t-1}$. For TransFusion to accurately undo the noise added between timestep $t-1$ and t, the predicted reverse distribution $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ should be close to the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ (which has access to the ground-truth transcript). So, a training loss is formed as the Kullback–Leibler (KL) divergence:

$$\mathcal{L} = \mathrm{KL}\left(\, q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p(\mathbf{x}_{t-1}|\mathbf{x}_t) \,\right)$$

Furthermore, the theory of multinomial diffusion also requires an additional term be added when $t = 0$ [80]. Namely the cross-entropy between the one-hot ground-truth distribution $\mathbf{x}_0$ and the predicted probabilities from TransFusion $\hat{\mathbf{x}}_0$ is added to the loss when $t = 0$ [80]. Intuitively, this pushes the distribution predicted by the model to be close to the one-hot ground truth targets when $t = 0$. This loss is readily computed and the model can be trained through backpropagating through the predictions $\hat{\mathbf{x}}_0$ used to compute $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

5.2.3. Architecture

TransFusion's architecture is based on a transformer [58] and is depicted in Figure 5.2. It draws on the diffusion conditioning paths proposed for images in [7], and incorporates the relative positional encoding used by wav2vec 2.0 [22]. Concretely, the model consists of 24 transformer layers where the vector sequence used for the self-attention block [58] is carefully crafted to incorporate timestep and conditioning information. For discrete inputs like the character sequence $\mathbf{x}_t$ and the timestep t, we first embed them into a continuous vector space using regular learnt embedding layers (for characters) or fixed sinusoidal embeddings [126] (for the timestep). The timestep embedding (after a few layers as in Figure 5.2) is summed with each character embedding to condition the sequence on the current timestep.

The frozen WavLM model computes the sequence of features $\mathbf{c}$ associated with the utterance, producing a vector for every 20 ms of the utterance. To condition TransFusion on these features $\mathbf{c}$, we adapt the technique proposed for image synthesis in [7] and compute two streams of information as shown in Figure 5.2. First, we sum the vector derived from the mean across the entire sequence $\mathbf{c}$ with each character embedding. And second, we concatenate the entire sequence of vectors $\mathbf{c}$ (after passing it through a few layers) with the sequence of character embeddings, and use this longer sequence as the keys and values for the self-attention block. Intuitively, the mean-pooled vector is meant to provide

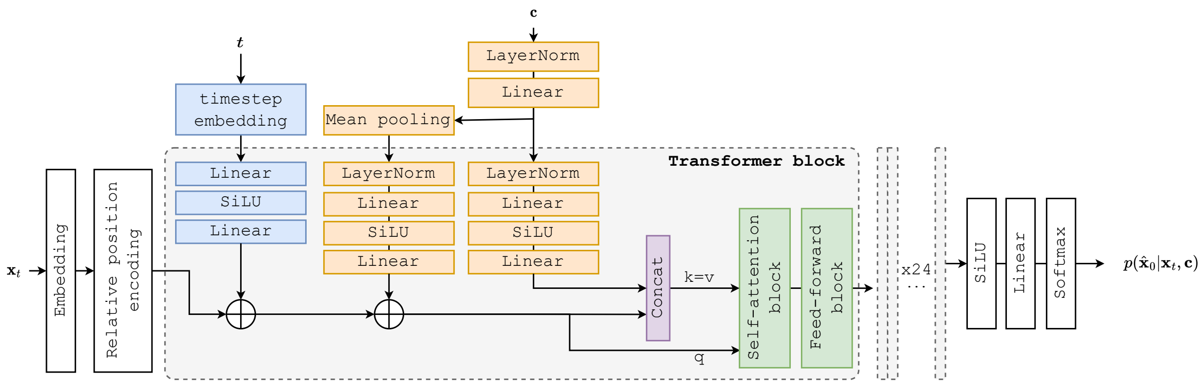


Figure 5.2: TransFusion architecture. The sequence of input characters $\mathbf{x}_t$ is passed through an embedding and positional encoding layer and then into a sequence of 24 transformer blocks before being projected to an output distribution $p(\hat{\mathbf{x}}_0|\mathbf{x}_t, \mathbf{c})$ over the vocabulary for each character in the sequence. In each transformer block, the mean-pooled WavLM features $\mathbf{c}$ and processed timestep embedding are added to each vector in the sequence which acts as the query to a self-attention block [58]. The key and value are formed by concatenating the transformed sequence of WavLM vectors with the main sequence derived from the characters. `SiLU`, `LayerNorm`, `Concat` layers refer to Sigmoid Linear Unit [127], layer normalization [128], and concatenation across sequence length, respectively.

TransFusion with a summary of the entire utterance, while providing the full sequence $\mathbf{c}$ to the attention layer allows for the model to learn more fine-grained spelling as it can attend to specific parts of the utterance for each query vector.

This series of operations is all encompassed in a single transformer block, and the full TransFusion consists of 24 such blocks and a final softmax projection head to yield the final distribution $p(\hat{\mathbf{x}}_0|\mathbf{x}_t, \mathbf{c})$. Finally, since concatenating the entire sequence of WavLM-derived vectors in each transformer block entails a very large memory and compute time cost, we only apply the `Concat` layer in Figure 5.2 in a select few transformer blocks (detailed in Section 5.4).

5.3. Diffusion decoding

To perform ASR with TransFusion, we must define the process of decoding a new utterance input through the diffusion model. The simplest form of the diffusion decoding process, described briefly in Section 5.2.1 and shown in Figure 5.1, iteratively computes $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$. An example of this decoding is given in Figure 5.3, where the speech features $\mathbf{c}$ contain the linguistic content "`MISTER QUILTER`". Starting at a random sequence of characters at $t = T = 200$, the model iteratively denoises until the final transcript is reached at $t = 0$. This approach, while effective, often results in errors related to the position of words in the overall sequence. If the model mistakes word placement early on, it is unable to make corrections because it is unable to easily shift characters. To this end, we propose new techniques for effectively decoding multinomial diffusion models.

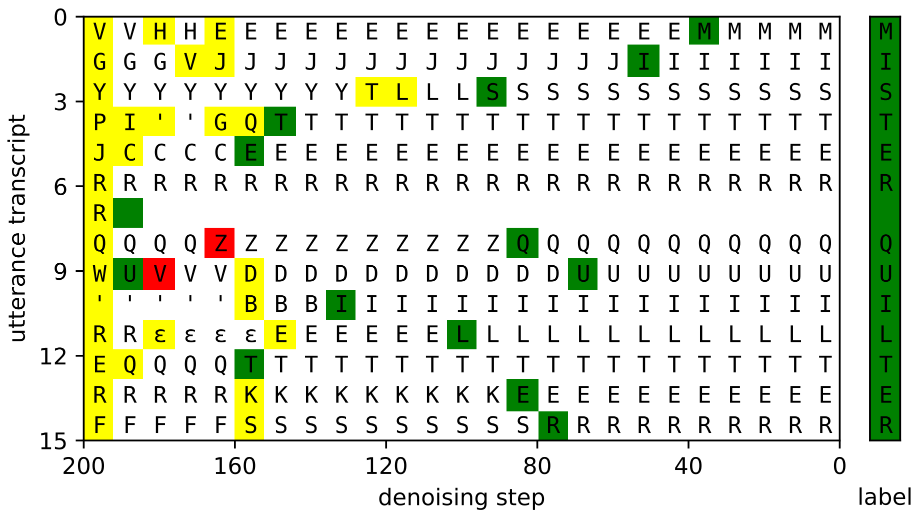


Figure 5.3: Example denoising process of the TransFusion model, starting from a random sequence $\mathbf{x}_T$ (leftmost column) and denoising until $\mathbf{x}_0$ (rightmost column, excluding label). Green blocks indicate transitions to the ground truth target transcription. Red and yellow blocks indicate transitions from a right character to a wrong character, and a wrong character to another wrong character respectively.

5.3.1. Resampling

RePaint [28] introduced the idea of resampling for the reverse diffusion process to improve inpainting performance during image synthesis. With RePaint, instead of linearly denoising from $t = T - 1$ to $t = 0$, they jump back and forth applying both forward and reverse diffusion on a set schedule. They found that repeatedly denoising and adding noise (i.e. diffusing) an image improved image generation quality, allowing the model to improve local coherency. This resampling schedule can be used directly in our decoding. Concretely, we define two constants, jump length L and number of jumps J. During decoding, we alternate between denoising (applying the reverse function $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$) and diffusing (applying the forward noising process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$), each lasting L timesteps and repeated J times. This is repeated until $\frac{T}{L} - 1$ times linearly along t. See [28] for precise details.

5.3.2. Sequentially progressive diffusion

Another method introduced in [28] is inpainting or known region conditioning. The idea of inpainting is to predict the missing pixels within a masked region of an image. Influenced by this, we considered using a similar approach by decoding word-by-word from the beginning of the transcript. The predicted words would be then treated as the unmasked region and the model then must predict the rest of the transcript. While effective, this approach is computationally expensive, as the entire denoising process must be repeated for every word in the transcript.

Instead, we develop a new tractable method to implement alongside resampling. We

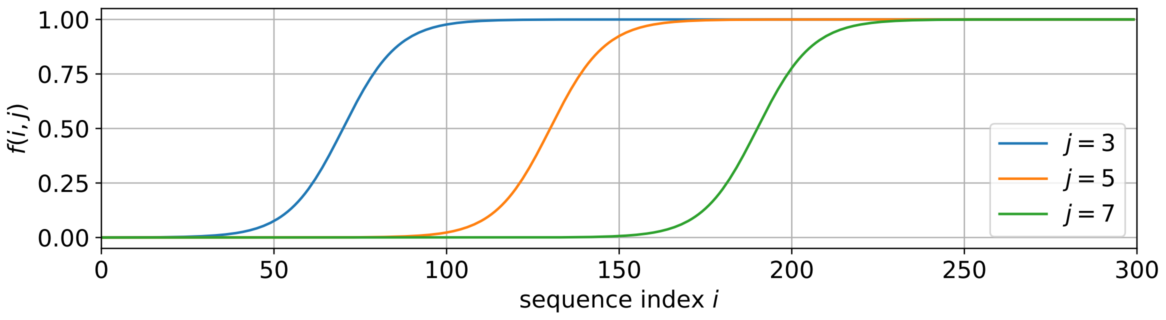


Figure 5.4: Example of sequentially progressive diffusion scaling function f for $J = 10$, shown over the character sequence for $j \in \{3, 5, 7\}$. This function modifies the variance scale β based on both resampling jump and sequence index. The actual function used is $f(i, j) = \sigma((i - \frac{jN}{J} + 2J)/8)$. The constant offset $2J$ is added to ensure the full sequence is diffused at $j = 0$, and 8 was analytically chosen to ensure a smooth overlap between sequential jumps j.

call this sequentially progressive diffusion: instead of applying forward diffusion uniformly along the sequence length, we now scale β along the length of the transcript. This allows us to retain earlier parts of the utterance while diffusing/noising the later parts. To scale diffusion noise based on sequence position, we define the scaled diffusion noise schedule $\beta'_{t,i}$ for each character position i:

$$\beta'_{t,i} = \beta_t \cdot f(i, j)$$

where f defines the scaling function and $j \in \{0, 1, \ldots, J - 1\}$ is the current resample jump. f is chosen such that the diffusion is applied uniformly at $j = 0$, and slides linearly along the sequence length, only diffusing the end of the transcript at $j = J - 1$. We implemented f as a shifted sigmoid as shown in Figure 5.4 to make a smooth transition between the retained and diffused regions.

5.3.3. Classifier-free guidance

As with other conditional diffusion efforts in the image synthesis domain [7, 8], we utilize classifier-free guidance [125] to improve the alignment between the output transcript and speech features $\mathbf{c}$. In our context, classifier-free guidance attempts to force TransFusion to learn both a conditional TransFusion($\mathbf{x}_t, \mathbf{c}$) and unconditional TransFusion($\mathbf{x}_t$) reverse diffusion process. This is achieved by randomly dropping out the conditioning information $\mathbf{c}$ with some small probability (in our case 0.1 – the same as those found to work well in [7, 125]). In this way, TransFusion learns to both generate unconditional realistic text and learns to generate text aligned with a transcript. Then during ASR inference, at each diffusion step, we update the output $p(\hat{\mathbf{x}}_{t-1} \,|\, \mathbf{x}_t, \mathbf{c})$ to move in the direction *from* the unconditional output *to* the conditional output with a 'guidance weight' [125]. More formally, during inference we set:

$$p(\hat{\mathbf{x}}_0|\mathbf{x}_t, \mathbf{c}) = w\,\text{TransFusion}(\mathbf{x}_t, \mathbf{c}) + (1 - w)\,\text{TransFusion}(\mathbf{x}_t)$$

Relating this to our discussion of disentangled latent spaces in Section 2.1.3, this idea works because the raw probability distributions from the model for $\hat{\mathbf{x}}_0$ are continuous and smoothly change from random guesses to the correct one-hot probabilities. So, performing a linear interpolation/extrapolation in the logit latent space of the TransFusion works, similar to how we extrapolated in the latent space of ASGAN in Chapter 3 to strengthen denoising action.

The reasoning here is as follows: if the model output (logits of $\hat{\mathbf{x}}_0$) *conditioned* on information about the utterance is TransFusion($\mathbf{x}_t, \mathbf{c}$) and *not conditioned* on the utterance information is TransFusion($\mathbf{x}_t$), then intuitively the linear direction from TransFusion($\mathbf{x}_t$) to TransFusion($\mathbf{x}_t, \mathbf{c}$) corresponds to the direction of increasing conditioning information. We can then improve the strength of the conditioning – in our case, to improve alignment of output transcript with the utterance – by linearly adjusting the conditional output in this direction [125]. Note that we apply this linear combination before the output `softmax` of the model in Figure 5.2 to ensure the adjusted output still is a valid probability distribution and that we are still operating in the somewhat disentangled latent space learnt by TransFusion (as opposed to a probability space after the softmax). With $w = 1$ in the above equation there is no guidance while increasing values $w > 1$ strengthens the guidance effect. We found $w = 1.5$ to yield the best results based on decoding ablations on our validation set (following the same ASR setup as described in Sec. 5.4). We use this setting in all our evaluations.

5.3.4. Full inference process

The full decoding process combines the methods defined above to perform ASR on an utterance from speech features $\mathbf{c}$. The core of the inference is resampling with sequentially progressive noise scaling in the forward diffusion steps. From the same validation decoding ablation experiments, we found resampling worked best with $J = 10$ and $L = 10$, which we use for our final resampling decoding in the next section. The reverse diffusion step utilizes classifier-free guidance to improve the alignment of the output transcript. We also note that we can use arbitrary sequence lengths at inference due to the model using relative positional encoding. In our final inference we use a sequence length $N = 400$ to ensure we cover all transcripts in the LibriSpeech test and dev datasets described in Section 5.4.1 ($> 99\%$ of LibriSpeech transcripts are shorter than 400 characters). In all our diffusion training and evaluation experiments we set β_t according to the cosine noise schedule from [126] using the recommended value of $s = 0.008$.

5.4. Experimental setup

We compare our model against other common ASR models on a standard speech recognition benchmark dataset: LibriSpeech [74]. Namely, we compare against the high-performing self-supervised speech representations models wav2vec 2.0 [22], Conformer [129], and w2v-BERT models [122].

5.4.1. Dataset and metrics

We perform our experiments on the LibriSpeech dataset [74]. It consists of ∼1000 hours of spoken audiobooks by multiple speakers with varying amounts of noise and audio quality. We train our model on the full 960 h `train` split of LibriSpeech and evaluate it on the official `dev` and `test`-splits. For the frozen WavLM model, we use the `WavLM-Large` pretrained model from the original authors [23]. Note that this model has only been pretrained with a masked prediction task on raw audio (Section 2.5.1), and has not been fine-tuned to perform ASR (i.e. it has never been exposed to transcripts of utterances). To evaluate our model we use the standard ASR metrics of word error rate (WER) and character error rate (CER), however ASR papers typically focus on the WER metric [13,23] so we focus on WER for comparison. We compute the mean WER for our model sampled using various decoding strategies and compare it against the baseline models (described next) using their best results reported by the original authors.

5.4.2. Baseline models

We compare against three high-performing models for ASR: wav2vec 2.0 [22], Conformer [129], and w2v-BERT [122]. The first two are large transformers trained in two phases. First, they are trained in a self-supervised fashion using large amounts of unlabeled audio on a masked token prediction task, and then they are fine-tuned with a CTC objective on the LibriSpeech dataset of labeled (i.e. transcribed) audio. The w2v-BERT model also follows this pretraining-finetuning setup, but also incorporates additional tricks to improve performance such as self-training with noisy student training [130]. These practical techniques to squeeze out additional performance from ASR models have been developed primarily for CTC-type ASR models and are largely undeveloped for diffusion-type ASR models. This is because CTC-type models (including all the baseline models) ultimately produce a probability distribution for a character or phoneme being present at a **certain time** in the utterance. Meanwhile, our diffusion-type model produces a probability distribution for a character being present at a **certain position** in the transcript.

Furthermore, decoding an acoustic model with a typical language model and lexicon has also been developed with these CTC-type models in mind, making them not very effective when applied directly to our model which predicts characters at fixed positions in the transcript. The primary reason for this is that any insertion or deletion of characters early on in the transcript will cause a substantial change in the predicted likelihood of the rest of the characters in the utterance. For example, if the modal model prediction for the first two words is "`SQUEZE IT`", and the lexicon is naively used to decode the first word to "`SQUEEZE `", then the diffusion acoustic model will have a very high likelihood of "`SQUEEZE T`" and a very low likelihood for the desired correction "`SQUEEZE IT`".

Again, this problem stems from our different way of phrasing the ASR problem as predicting characters at fixed positions in a transcript. So, if we insert a character early on in the decoding process against the acoustic model's recommendation (i.e. not using its modal prediction), the acoustic likelihood will incentivise the dropping of a character elsewhere in the transcript to retain a high likelihood score for the ground-truth transcript. Because decent techniques for language model decoding have not yet been developed for diffusion-type models, all experiments in this paper do not use a lexicon or language model—the models that we compare to are also used without a language model or lexicon.

5.4.3. TransFusion implementation

Layers

For the relative positional encoding layer at the input to the transformer, we use the formulation provided in [22]. As mentioned in Section 5.2, we do not apply the `Concat` layer in every transformer block. To save compute resources and to allow for an improved attention weighting (discussed next), we only apply the `Concat` operation in every 4th block. So, with 24 transformer blocks, the `Concat` layer is present in layers $1, 5, 9, 13, 17, 21$. The self-attention and feed-forward blocks follow the same architecture as in the original attention article [58].

Model hyperparameters

Our model uses a 29-sized character alphabet (letters and special padding/punctuation characters) and contains 24 transformer blocks. The output dimension of all embedding, linear, and attention layers is 768. The feed-forward blocks have a dimension of $4 \times 768 = 3072$ and each self-attention operation uses 8 attention heads. Transformer attention and feed-forward blocks use a dropout of 0.1, and we also completely dropout all conditioning information with probability 0.1 (in line with classifier-free diffusion guidance [125]). For the relative positional encoding [22], we use a 256-size convolution kernel with 32 convolution groups. The conditioning sequence $\mathbf{c}$ from `WavLM-Large` model is defined as the average of the activations of the last 9 layers from the model pretrained on LibriLight [131], since [23] found these last layers to be most important for representing linguistic information.

Table 5.1: WER results on the LibriSpeech dev and test splits for ASR models trained on the full 960 h LibriSpeech training set. Decoding for prior models is done with lexicon-free CTC-decoding [26], while several decoding strategies are applied to our diffusion model (Section 5.3). No language models are used in decoding, and all models are fine-tuned on the full LibriSpeech 960 h training data. The unlabeled data used to pretrain each model is specified as the LibriSpeech 960h dataset (LS-960h) [74] or LibriLight 60 000 hour dataset (LV-60kh) [131].

			dev set		test set	
Model	Pretraining	Params (M)	clean	other	clean	other
Including pretraining						
wav2vec 2.0 `Base` [22]	LS-960h	95	3.2	8.9	3.4	8.5
wav2vec 2.0 `Large` [22]	LS-960h	317	2.6	6.5	2.8	6.3
wav2vec 2.0 `Large` [22]	LV-60kh	317	2.1	4.5	2.2	4.5
Conformer XXL [129]	LV-60kh	1000	1.6	3.2	1.6	3.3
w2v-BERT XXL [122]	LV-60kh	1000	**1.5**	**2.7**	**1.5**	**2.8**
No pretraining						
wav2vec 2.0 `Large` [22]	None	317	2.8	7.6	3.0	8.5
Conformer L [129]	None	103	**1.9**	**4.4**	**2.1**	**4.3**
TransFusion (ours, 350k updates)	None	253				
basic decoding (Sec. 5.2.1)			9.6	12.1	10.5	12.5
+ classifier free guidance			9.4	11.7	10.2	12.2
+ resampling			8.4	10.7	9.0	11.0
+ sequentially progressive diffusion			8.1	10.5	8.9	10.8
+ further trained to 462k updates			**6.1**	**8.3**	**6.7**	**8.8**

Optimization

We train TransFusion on the full LibriSpeech 960 h training subset for 350k updates using a batch size of 720 with Adam optimization [93] with $\beta = (0.9, 0.999)$. Further, we use a constant learning rate of 3×10^{-5} with a linear warmup of 10k updates and clip the global gradient norm at 10. As this is an initial foray into ASR with diffusion, we do not use any data augmentation such as SpecAugment. This differs from the baselines, all of which have been trained with substantial data augmentation to further improve performance [22, 122, 129]. Even without augmentations, the model's validation performance (WER on internal validation split) was still improving at the end of training – we hypothesize that training a larger model for longer with more compute resources and all the typical data augmentation techniques used for other CTC-type models will yield further improvements to results listed in the next section. To demonstrate the level of improvement gained from longer training, we continue training our model up to 462k updates and show its performance compared to the base 350k update model in the next section. All training is done on three NVIDIA Quadro RTX 6000 devices with mixed FP16/FP32 precision.

5.5. Results

The ASR results are given in Table 5.1. First we observe that our new model using our best decoding strategy does not beat the current state-of-the-art CTC-type models such as Conformer L or w2v-BERT. However, the performance is still considerable given that this is a first investigation into an entirely new approach to ASR using discrete diffusion. Concretely, we achieve a `test-clean` and `test-other` WER of 6.1% and 8.8%, respectively. The nature of many of our model's mistakes follows the issue outlined in Section 5.4.2. Specifically, often if the model decodes a character by an erroneous insertion/deletion early on in the transcript, it will attempt to drop or insert another erroneous character later on to ensure that the alignment for the rest of the characters is still correct. While this worsens the WER, the effect on CER is less impactful, where TransFusion achieves a CER of 3.2% and 3.6% on `test-clean` and `test-other`, respectively.

Furthermore, we observe from Table 5.1 that each of our decoding techniques cumulatively improves the results, with our final addition of sequentially progressive diffusion yielding a more than 1.5% absolute WER improvement over basic decoding. This demonstrates the effectiveness of our initial decoding methods and suggests that even further improved performance may be achievable given better decoding approaches. Further training our model to 462k updates also substantially improves performance (last row of Table 5.1), bringing the `dev-other` subset results above that of wav2vec 2.0 `Base`. This suggests that – in line with our motivation about diffusion scaling in Section 5.1 – even greater performance is likely achievable with increased compute and model sizes.

It is also interesting to observe that the difference in WER for TransFusion between the less noisy `dev-/test-clean` sets and the more noisy `dev-/test-other` sets is much less than all the CTC-type models. For the CTC models, WER on the clean subsets are often half of the result on the noisy subset, while with the diffusion model the difference is much smaller. While we are not sure of the precise reason for this, we speculate that our decoding method for diffusion-type acoustic models does not draw out the full performance possible from TransFusion, unlike the well-explored CTC decoding techniques possible with CTC-type models. In other words, if the performance trend of the CTC-type models is representative of the difficulty difference between `clean` and `other` subsets, then we should be able to achieve a better clean performance once more optimal decoding methods are developed for diffusion-type models. Finally, while TransFusion does use features from a pretrained WavLM model, it does not fine-tune the WavLM encoder which provides these features unlike the other methods considered, hence we denote the trained weights of TransFusion as not including any pretraining in Table 5.1.

5.6. Summary and conclusion

This chapter investigated our third research question: can we apply implicitly disentangled SSL features and continuous modelling techniques to a discrete speech processing task and see similar benefits as in the prior two chapters? We proposed TransFusion – a model that utilizes multinomial diffusion to phrase the task of speech recognition as a conditional discrete diffusion task. TransFusion iteratively denoises an arbitrarily noised transcript until it resembles coherent text corresponding to the transcript of a provided utterance. This is done by providing disentangled SSL speech features associated with the utterance to condition a large transformer model predicting a categorical distribution over a character alphabet. Since we are the first to phrase ASR in this way, we proposed new methods to decode such diffusion-type ASR models during inference.

With the model designed and trained, we went on to assess how it compares against the current status quo for ASR. We evaluated TransFusion's performance on the LibriSpeech dataset and compared it to existing state-of-the-art CTC-type models, demonstrating comparable performance. While we do not outperform the best large CTC-type models we compare to, we achieve a 8.8% WER / 3.6% CER on the LibriSpeech `test-other` set. This is noteworthy given the completely new method for ASR proposed here, considering how much effort has gone into optimizing and perfecting CTC-type ASR models.

Given these results, we can draw two main conclusions: DDPM scaling properties seen in continuous domains are also present in ASR, confirming the benefit of applying diffusion (a typically continuous method) to this discrete task. Second, using continuous disentangled SSL features as conditioning does allow our new ASR formulation to compete with the highly developed CTC models. A key limitation in this chapter is that we do not yet outperform current state-of-the-art CTC models. This is to be expected, however, given this is the first work attempting to approach ASR in this new way. I.e. there are likely better decoding methods possible, and it remains an open question how to combine these discrete diffusion acoustic models with language models – avenues for future work.

In this and the previous chapter, we looked at how continuous, disentangled features can be applied to the generic, well-explored tasks of voice conversion (Chapter 4) and ASR (this chapter). With the success and applicability in these settings, we want to know if our motivation around disentanglement holds in less well-explored, practical situations. So, in the following chapter, we investigate whether these benefits of 'designing around using continuous disentangled features' hold for several downstream tasks.

Chapter 6

Downstream applications of disentanglement

Publications

This chapter is adapted from our publications:

Published conference paper: M. Baas, H. Kamper, "Voice Conversion Can Improve ASR in Very Low-Resource Settings," in *Proceedings of Interspeech*, 2022.

Published journal article: M. Baas, H. Kamper, "Voice Conversion for Stuttered Speech, Instruments, Unseen Languages and Textually Described Voices," *Springer Communications in Computer and Information Science*, vol. 1976, pp. 136-150, 2023.

In the investigations up to this point we have been intrinsically linked to disentanglement. The unconditional speech synthesis of Chapter 3 looked at learning good disentangled spaces. The voice conversion model of Chapter 4 looked at how to incorporate disentangled features to improve a more general speech processing task. Then, the ASR investigation of Chapter 5 looked at whether disentangled latent spaces can be applied to discrete-output tasks like speech recognition. But, a key part of research is *practical usability* – how can we translate our efforts to useful contributions in *downstream* applications? This chapter aims to do just that, by applying the lessons learned in the previous chapters to several downstream tasks corresponding to practical use cases of speech processing technology.

We split this chapter into two broad parts. In the first, we try combine the lessons of Chapter 3 and 4 to build a voice conversion model that learns to implicitly disentangle speaker identity in a single end-to-end system (i.e *without* large SSL models trained on massive datasets). Namely, Chapter 3 showed how to learn a disentangled latent space in a restricted setting, and Chapter 4 showed how we can apply existing latent spaces to voice conversion. Combining these two, we ask – can we learn a disentangled latent space jointly with a voice conversion model? And do so in a practical downstream application?

We attempt to answer both these questions in the affirmative by proposing an end-to-end, any-to-any VC model based on hierarchical global style tokens (HGSTs, explained later) and test its generalization in a very practical use case: data augmentation for improving ASR in very low-resource settings. The idea is to apply voice conversion to improve speech recognition systems in low-resource languages by using it to augment limited training data by increasing speaker diversity. We evaluate our method as an ASR augmentation technique against existing techniques on four low-resource South African languages.

In the second part of this chapter we want to know if the models we have developed so far can truly generalize to uncommon, downstream speech processing applications. ASGAN (Chapter 3), like other unconditional speech synthesis systems, is still limited to restricted settings which makes them not particularly useful in practical scenarios, and TransFusion (Chapter 5) is an ASR model, where ASR is already a well-recognized useful and practical task. So, we focus on applying kNN-VC (Chapter 4) to more out-of-domain downstream tasks. We investigate its application to stuttered speech correction, cross-lingual conversion, musical instrument conversion, and even textually described voice conversion (where the target speaker is specified by a text prompt).

Understanding the extent to which these models are useful in practical situations will be the final validation of our motivation and central thesis defined in Chapter 1 – that methods based around implicit disentanglements of speech yield improved generalization and control compared to explicit disentanglement methods.

6.1. Voice conversion for ASR data augmentation

The first part of this chapter focuses on developing a voice conversion model that can learn an implicit disentanglement for speaker identity, and apply that idea to a downstream task: improving ASR in low-resource settings by using it as a data augmentation technique. If we can show that learning a disentangled latent space does yield benefits in this task, then we will have evidence to answer our last research question of Section 1.3 in the affirmative.

While ASR performance has greatly improved over the last few years [13, 22, 129], they still struggle in very low-resource settings where training data is minimal and a high-quality language model (LM) is not available [132]. Concretely, even state-of-the-art ASR performance is limited for languages where we only have a few minutes of transcribed audio from a handful of speakers [133], e.g. when building systems for regional dialects or code-switched speech.

One possible solution is to use data augmentation: available data is processed so that it has different characteristics from the original while preserving the content (words spoken), effectively giving more training data. This idea has a long history [134, 135], with more recent methods like SpecAugment [136] improving ASR performance and robustness. While these signal processing approaches have proven effective, they are mostly constrained to direct general-purpose modifications of the audio waveform – e.g. pitch shifting, time stretching, and frequency masking – ignoring some of the unique aspects of human speech. Using voice conversion to augment data is one way to address this, which we attempt here.

To use voice conversion for data augmentation, ASR training utterances are fed into a voice conversion model and converted to arbitrary target speakers, thereby creating additional synthetic training utterances with a known transcript. It has been hypothesized that this would improve ASR performance in low-resource settings [137], although initial attempts (discussed later) have had limited success. Moreover, for a voice conversion model to be practically useful for ASR, it should be applicable to low-resource languages which have not been seen during training. To address this, we propose a voice conversion system trained on English that is practical for cross-lingual data augmentation: running faster than real time on unseen speakers and unseen languages. We then apply it to create synthetic data for several low-resource settings in English, Afrikaans, Sepedi, isiXhosa, and Setswana. We then compare the performance of ASR systems trained with and without generated voice-converted data, and also compare to a well-known augmentation method SpecAugment [136]. With limited training data (10 min) we find that our voice conversion approach is complementary to SpecAugment on English, while it is superior for data augmentation on the low-resource languages. To the best of our knowledge, our work in this chapter is the first to use cross-lingual voice conversion for data augmentation, where voice conversion is applied to unseen languages and speakers.

6.1.1. Related work

There have been several prior attempts to augment data using neural network techniques [138]. Laptev et al. [139] explored using text-to-speech (TTS) methods to perform data augmentation, obtaining improvements on medium-resource ASR. Since terms about resources are contested in the literature [140], for this chapter we define *very low-resource*, *low-resource*, and *medium resource* as settings with roughly 10 min, 1 h, and 10 h to 50 h of labelled audio, respectively. For very low-resource applications like the ones explored in this paper, a good TTS model or LM is usually unavailable.

Cui et al. [141] made initial attempts to improve medium-resource ASR using voice conversion, showing moderate improvements despite limitations of techniques available at the time. Other studies [142, 143] showed that using more recent deep learning-based voice conversion methods can improve TTS systems where the desired speaker is low-resource. However, these studies use a target speaker from a language seen during training, leaving the question open as to whether ASR can be improved in unseen low-resource languages.

Zhao et al. [144] made an attempt to use cross-lingual voice conversion to improve code-switched TTS models. However, their conversion models are trained separately for each speaker considered, making their system unsuitable for use on new unseen low-resource languages with many speakers. Finally, [145] found that applying a deep-learning-based voice conversion approach failed to noticeably improve medium-resource ASR models when all models are trained and evaluated on English. They were, however, limited by practical problems associated with voice conversion techniques available at the time, such as the fast but low-quality Griffin-Lim vocoder.

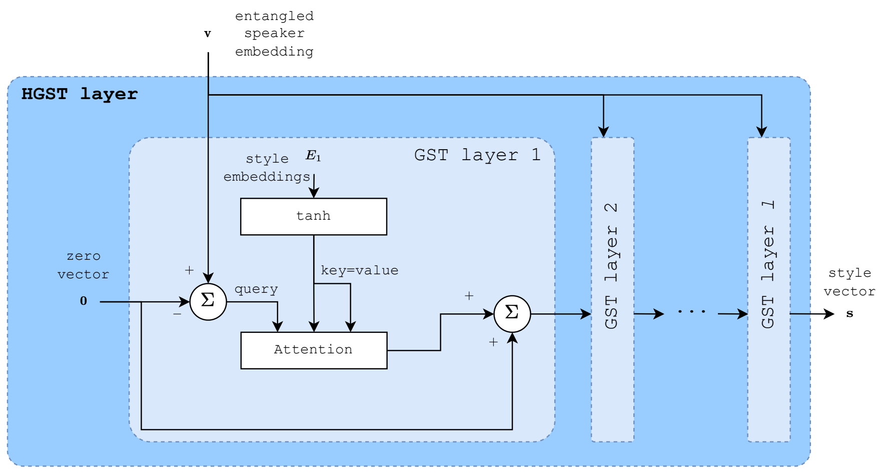


Figure 6.1: Hierarchical global style token (HGST) layer diagram. The layer consists of l global style token (GST) sublayers, each of which has a series of h trainable style embeddings E which, after a `tanh` activation, act as the keys and values for an attention operation [58]. The attention query is the input vector to the later $\mathbf{v}$ less a rolling residual vector (initialized as zeros).

6.1.2. Hierarchical global style tokens

Recall in the introduction for this chapter we make a simplifying assumption that we are only interested in learning a disentangled latent space for a single aspect of speech: speaker identity. Because of this decision, we can use hierarchical global style tokens (HGSTs) as the key to learning a disentangled space corresponding to speaker identity. HGST is a technique introduced in [29] to model different levels of abstract concepts of speaker identity with several continuous latent spaces. Intuitively, the HGST is a model layer that accepts a single vector as input and produces a single vector as output, with the aim to disentangle speaker information contained in the input vector such that the linear directions in the vector space spanned by its output vectors are associated with common factors of speaker identity variation. E.g. the input speaker vector may be highly entangled (being difficult to separate out speaker gender, speaking rate, speaker loudness, etc...), and the output vector should ideally be disentangled (having linear directions/clustering corresponding to gender, rate, loudness, etc...)

The concrete specification of the HGST layer is given in Figure 6.1. The layer consists of l global style token (GST) sublayers [146], each of which has a trainable matrix of h style vectors. Intuitively, the first GST layer attempts to represent the entangled input vector $\mathbf{v}$ containing speaker information as a convex combination of the first GST layer's style embeddings E_1. This forces the rank of the output vector of the first GST layer to be at most h (since it is expressed as a linear combination of h style vectors), enforcing a low-dimensional subspace on the output. *Then*, the residual difference between the original vector $\mathbf{v}$ and the approximation produced by the first GST layer is then fed to a second GST layer for another round of approximation. This process continues in this way until the last sublayer l, where the vector $\mathbf{v}$ has been recursively approximated by low-rank subspaces.

If the latent space in which final output style vector $\mathbf{s}$ exists is to be disentangled, then the residual approximated by each of these GST layers – acting as information bottlenecks – must encode common factors of speech variation in a hierarchical fashion, hence the name HGST. An important question to ask is: why must the embeddings of each GST layer *approximate* the input vector? While tricky to observe at first glance, if one looks closely at the form of each GST layer, the output of the attention layer (i.e. a convex combination of the `value` and thus embedding vectors) is summed with the residual. For later GST layers to effectively learn from this summed output, they must encode similar factors of speech variation.

More technically, during model design, if we enforce the design constraint that speaker information can *only* be obtained through this final style vector $\mathbf{s}$, and the model is trained in some way to match the synthesized speech with a provided ground-truth sample, then we can show that each GST sublayer learns to recursively approximate the entangled input vector $\mathbf{v}$ [29].

In this way, a single vector $\mathbf{v}$ is expressed as the sum of l convex combinations. The embeddings and residuals for each sublayer in the hierarchy encode different aspects of speech. For example, for $l = 3$ sublayers, the first layer encodes speaker gender and broad speaker characteristics, while the second encodes more fine-grained speaker identity, and the last sublayer encodes details such as speaker microphone quality [29]. This ability to learn low-rank manifolds corresponding to common speaker characteristics makes this a key technique for designing a model to learn a disentangled latent space over speaker identity for speech synthesis.

6.1.3. Model design

Our goal is to use voice conversion for low-resource ASR data augmentation. To that end, we opt to use these HGSTs to learn a disentangled latent space corresponding to speaker identity.[1] To be practically useful for ASR augmentation, the model must allow for effective cross-lingual voice conversion to unseen languages, in addition to being computationally efficient, and any-to-any in nature (Section 2.4.1). And remember, our goal here is not to build the best voice conversion system (as we did in Chapter 4), but rather to learn a disentangled latent space together with speech synthesis in a single model, while scaling sufficiently well to be useful in a practical task.

[1]We note that the voice conversion system introduced here is more complex than that proposed in Chapter 4. This is because, chronologically, the design and development of the model in this section happened in 2022, before that of kNN-VC in 2023.

Overview

Our approach is shown in Figure 6.2. We want to convert an input waveform into a sequence of feature vectors and then separate the *linguistic content* (the words being spoken) from the *speaker information.* We do this with specially designed style and content encoders, denoted with dashed boxes. The style encoder produces a single *style vector* $\mathbf{s}$ (the output of the HGST layer), while the content encoder produces a sequence of *content embedding vectors.* During training (Figure 6.2a) the style vector $\mathbf{s} = \mathbf{s}_{\text{src}}$ and content embeddings are obtained from the same input utterance. The model is trained to reconstruct the input by summing $\mathbf{s}_{\text{src}}$ with each content embedding, passing the resulting vectors into a decoder module. At test time (Figure 6.2b) the content and speaker encoders receive speech from different utterances. This means that the source or reference utterance (or both) may be from languages and speakers unseen during training. If the content and speaker information are appropriately separated, then the source utterance will be produced in the voice of the speaker supplying the reference utterance. Details on each component follows.

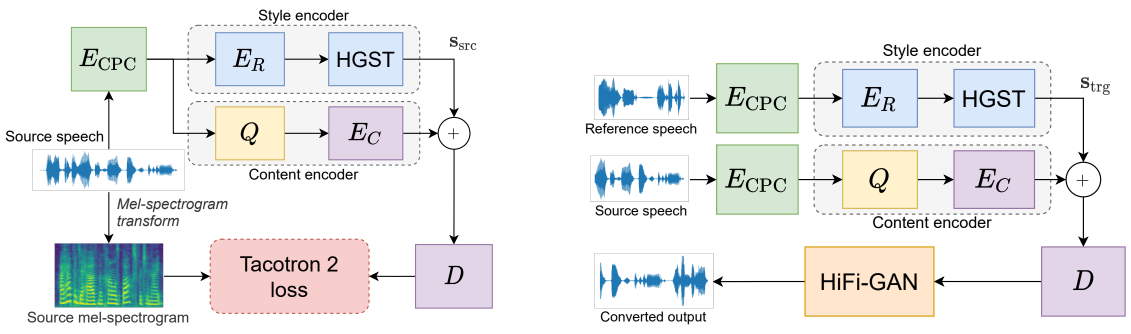
Speech encoder

Instead of using a spectrogram to represent the reference or source utterances, we use a pretrained speech encoder. Specifically, we use the contrastive predictive coding (CPC) model from [71]. This model, E_{CPC} in Figure 6.2, produces a 512-dimensional vector for every 10 ms of 16 kHz-sampled audio. This model transforms the input waveform into representations that make it easier to disentangle content and speaker information for the downstream style and content encoders [70]. The CPC model is an early variant of SSL speech representation models (Section 2.5.1) trained only on LibriSpeech (960 h of English speech). Compared to large-scale recent SSL models (e.g. WavLM [23], data2vec 2 [147]), this CPC model is tiny (in terms of parameter count) and trained on English-only data[2]. This makes it efficient at inference (important if we wish to generate many augmentations of a whole ASR dataset), and a good test case to check whether we can learn a disentangled latent space for speaker identity that generalizes to new languages entirely unseen by any component of the system.

Style encoder

The style encoder consists of a reference encoder E_R and a HGST layer [65]. The reference encoder E_R consists of several 1-D convolutional layers followed by an LSTM [148]. The final hidden state of the LSTM is linearly transformed and passed to the HGST layer. The HGST layer acts as a form of information bottleneck; the idea is that the network

[2]WavLM, data2vec 2, and other recent top-performing SSL models are typically trained on over 60 kh of speech data, sometimes covering several languages [23, 147]



(a) Training setup **(b)** Inference setup

Figure 6.2: Our cross-lingual voice conversion model. (a) During training, the same utterance is fed through a CPC encoder into both the style and content encoders; the decoder tries to reconstruct the input utterance's spectrogram. (b) At inference, separate input and reference utterances are fed into the style and content encoders, respectively, before vocoding the spectrogram using HiFi-GAN [53].

learns to *only* include speaker information in $\mathbf{s}$, since the content information can be obtained from the content encoder output. Recalling Section 6.1.2, the HGST layer is parameterized by l sublayers and h trainable vectors per sublayer. The first sublayer attempts to represent the last output vector from E_R as a convex combination of its h vectors. The difference between this combination and the input vector is then used as the input to the next sublayer, so each layer in the hierarchy models the residue from the previous approximation [65]. With small h and l, the HGST layer forces the final style vector $\mathbf{s}$ to only retain information about the global speaking style – i.e. that information which is constant throughout an utterance.

Content encoder

The content encoder attempts to remove speaker information from the CPC features so as to only retain linguistic content. The quantization block Q in Figure 6.2 first performs speaker normalization using means and variances computed for features from the pretrained E_{CPC} on a subset of the training data. This was shown to improve speaker invariance [70] (of which we are a co-author). During inference with unseen source speakers, we compute the mean and variance on the features of the single source utterance. The normalized embeddings are quantized by representing them by the nearest centroid vector in a K-means clustering model with $K = 100$. This quantized sequence is then passed through a content encoder module E_C with the same architecture as E_R, except returning all LSTM states – one content vector every 10 ms. The K-means and speaker normalization parameters are precomputed before the rest of the network is trained. One may ask, "if we are trying to design for disentanglement, why is the model performing quantization? Does that not reinforce the tension of Section 2.2.1?" While it does reinforce the tension, this is only the case if we want to disentangle all aspects of speech. But, remember for this investigation, we are only interested in learning a disentangled space for *speaker identity* – so only the speaker encoder block (top path of Figure 6.2) must be free of quantization, as it is.

Decoding and vocoding

Each content vector is summed with the style vector $\mathbf{s}$. The resulting sequence is then passed into a Tacotron 2 decoder network D to produce an output mel-spectrogram [2]. We use the same Tacotron 2 decoder architecture as in [2]. During training (Figure 6.2a), the loss is formulated the same as for the original Tacotron 2: an L_2 loss between the predicted spectrogram and the spectrogram of the source utterance with an additional loss associated with the length of the output spectrogram [2]. During inference (Figure 6.2b), the output spectrogram is converted to the time-domain using the spectrogram parameters and pretrained HiFi-GAN model from [53], due to its high performance and fast inference speed. With these choices for the different modules, our overall approach is faster than real-time on a modern GPU.

6.1.4. Experimental setup

Data

We want to know whether a voice conversion model trained on a well-resourced language can be applied to an unseen low-resource language to improve ASR. English is our well-resourced language; we use data from LibriSpeech [74], which consists of $\sim$960 h of read English. We consider four low-resource languages: Afrikaans, Setswana, isiXhosa, and Sepedi. Data for the first three are obtained from [149] (roughly 3.2 h per language), while Sepedi data is obtained from [150] (10.8 h).

VC model

For the speech encoder E_{CPC}, we use the pretrained SSL model `CPC-big` from [71] (it is trained on a small 6 kh subset of the Libri-Light dataset [131]). The E_R and E_C modules use exactly the same architecture as in [2], except that the convolutional kernel sizes are 9 instead of 5 and E_R has an additional 512-dimensional linear output layer. For the HGST layer, we use $l = 3$ sublayers with $h = 5$ vectors each, outputting a 512-dimensional style vector $\mathbf{s}$. With the pretrained E_{CPC} and quantization module Q fixed, the weights of E_R, E_C, D, and HGST are trained on 275k utterances from the LibriSpeech training set and validated on the remaining utterances in the set. We use Adam optimization [93], gradient norm clip of 1, a maximum learning rate of $6 \cdot 10^{-4}$ with the scheduler from [151], and train for 50 epochs with a batch size of 32. To stabilize training, we further follow [39]: utterances in a batch are randomly cropped to a fixed length; the base crop length is linearly increased from 0.6 s to 10.2 s over the training iterations, with the actual crop length uniformly sampled from ±20% of the current base value.

Voice conversion inference

In all experiments, the reference utterance is randomly selected from the LibriSpeech training set (recall we can do this because we are purely interested in increasing the diversity of the ASR dataset, so ASR training utterances to speakers known to the voice conversion system is acceptable). For Afrikaans, isiXhosa, and Setswana, we only consider reference utterances from female target speakers since these datasets only contain speech from females. For English and Sepedi, we consider reference utterances from both male and female speakers. At inference time, we convert arbitrary length utterances into 7 s chunks, stitching the results together to produce a final output.

SpecAugment

We use the SpecAugment settings exactly as in [136]. We also consider whether voice conversion and SpecAugment are complementary: in these experiments we *first* perform voice conversion and *then* further augment the result with SpecAugment. We denote this chained augmentation as VC→SpecAug.

Automatic speech recognition

For doing ASR in each of our low-resource languages, we try mimic the approach likely to be taken when first attempting to build an ASR system on a very low-resource language. Specifically, we fine-tune the pretrained `XLSR-53` wav2vec 2.0 model [133] with a character-level CTC loss [26]. For the initial validation experiments on English, however, we use the pretrained wav2vec 2.0 English models from [22]. We train the ASR models using `fairseq` [92] with its 10 min fine-tuning configuration unless otherwise stated. Since we are interested in demonstrating the potential *change* in ASR performance, we opt to use greedy CTC decoding without an LM to isolate the effect of augmentation on the acoustic model. In many real low-resource settings, it might also not be possible to obtain a good LM. However, to confirm the trends for cases where an LM is available, we also include English results with the official 4-gram LibriSpeech LM [74].

6.2. ASR augmentation experiments

Given the model design in the previous section, we now want to know if our design motivation around HGSTs is valid, and whether the model is practically useful for ASR data augmentation. To answer this, we perform three sets of experiments using the trained voice conversion model:

1. Can it perform voice conversion well in the easiest case (**seen** speakers, **seen** language)? In other words, does it disentangle speaker information well from the rest of the information in speech? Can it even scale more than the pure unconditional models investigated in Chapter 3?

2. Can it improve ASR on English (**unseen** speakers, **seen** language)? I.e. is it performant enough to be useful for augmentation at all?

3. Can it improve ASR on Afrikaans, Setswana, isiXhosa, and Sepedi (**unseen** speakers, **unseen** languages)? I.e. is the disentanglement strong enough to generalize to new speakers and languages?

These experiments have associated audio samples, and we encourage the reader to listen to the generated samples for each experiment: https://rf5.github.io/interspeech2022/

6.2.1. Experiment 1: Validating voice conversion quality

We first perform validation experiments to confirm that our voice conversion model performs adequately both on English (its training language) and unseen languages. For the latter, we focus here on Afrikaans and Sepedi, treating the other two languages as completely unseen test languages for our final experiment (Section 6.2.3).

To assess intrinsic conversion quality, we use two tests similar to those used to assess kNN-VC in Section 4.4. The first checks whether linguistic content is retained through a re-synthesis experiment [39]. Each test utterance from each language is converted to a target speaker as per Section 6.1.4. ASR is then performed on both the input and converted utterances. If the drop in ASR scores after conversion is small in terms of word/character error rate (W/CER), this indicates that voice conversion did not significantly reduce intelligibility. Table 6.1 shows the results. As expected, the converted output is less intelligible – higher W/CER for the full model – compared to the original audio, with a larger drop in scores on the languages that have not been seen during model training (Afrikaans and Sepedi). However, the drop is small enough that the output utterance can still be largely understood.

The second intrinsic test assesses how well speaker information is disentangled from content information **and** the extent of learnt disentanglement within the speaker embedding latent space. This is done by measuring how well a converted utterance matches the target speaker instead of the original source speaker. Concretely, we use our pretrained speaker verification model[3] following the [34] speaker verification task to check whether the speaker embedding associated with the converted speech is closer to that of the reference or source utterance in terms of their cosine similarity. We report an error rate: the proportion of converted test-set utterances that are incorrectly closer to the source utterance rather than

[3] https://github.com/RF5/simple-speaker-embedding v0.1

Table 6.1: Re-synthesis results in terms of ASR performance on original and converted data. The English-trained voice conversion system is applied to English, Sepedi, and Afrikaans evaluation data. Lower W/CER (%) is better (higher intelligibility).

	English		Afrikaans		Sepedi	
VC Model	WER	CER	WER	CER	WER	CER
Original data	5.7	1.9	6.3	4.3	2.1	0.9
Full model	20.6	9.6	32.5	11.0	20.4	9.5
Sans HGST	21.3	9.9	34.0	11.9	21.1	9.9
Sans Q	**7.1**	**2.6**	**17.0**	**4.6**	**3.7**	**1.6**

Table 6.2: Speaker similarity error rates (%). Lower scores indicate that converted utterances are more often closer to the reference speaker than the source speaker (better voice conversion).

VC Model	English	Afrikaans	Sepedi
Full model	8.7	22.0	58.3
Sans HGST	**2.3**	**6.9**	**34.4**
Sans Q	99.7	99.8	99.9

the reference. The higher this error rate, the *poorer the disentanglement*, as it means it is easier to identify the output utterance as containing stronger traces of speaker information associated with the original speaker. Note this is different from the EER used to assess kNN-VC (Section 4.4), where higher EER scores were better – in this investigation EER scores are less useful due to the very low-resource nature of some of the ASR languages (where some speakers only having a single utterance, preventing EER computation).

Table 6.2 shows the results. We again see a drop in performance from seen to unseen languages for the full model, where the converted Afrikaans and Sepedi data lead to more errors in capturing the reference speaker identity. Recall, though, that our goal of data augmentation is to ensure that the converted utterance sounds sufficiently different to the input utterance to increase speaker variety in a limited dataset while still being intelligible so as to aid in ASR training.

The last two rows in Tables 6.1 and 6.2 show the effect when we remove either the HGST or the normalization-quantization block Q from the full model. Without the HGST bottleneck, speaker conversion performance improves (Table 6.2), but at the cost of intelligibility (Table 6.1). This makes sense as the HGST acts as an information bottleneck (Section 6.1.2). So, without the HGST layer, the style vector $\mathbf{s}$ captures more granular reference speaker information (better conversion) as there is no information bottleneck. However, this also causes it to contain more linguistic/content information from the reference utterance (less intelligible) – meaning that the linguistic and speaker information are more entangled if we remove the HGST layer!

The opposite happens when we remove the bottleneck Q from the content encoder, where substantially more speaker information leaks through (poorer speaker conversion), but more granular content information is present (more intelligible) – again increasing the entanglement between speaker information and content information. Including both blocks therefore reinforces the disentanglement between speaker and content information and gives a balance between intelligibility and conversion quality, so we use the full model in the remaining experiments.

6.2.2. Experiment 2: ASR augmentation on English

Before applying voice conversion cross-lingually (Section 6.2.3), we first want to test what level of augmentation is beneficial on the seen language (English). We want to specifically know: How much and in which low-resource settings does real vs augmented data help? And how does it compare or synergize with SpecAugment (a well-known spectrogram warping augmentation technique)? To answer these questions, we fine-tune the `Base` wav2vec 2.0 English ASR models on 10 min, 1 h, and 10 h of real data from Libri-Light's fine-tuning dataset [131] as simulated low-resource settings. In each setting we train several models: we compare VC (just applying voice conversion), SpecAug, and VC→SpecAug augmentation (with 100% and 500% of additional generated data) to an ASR model trained without any augmented data.

The results are given in Table 6.3. Looking at the 10 min setting without an LM, we can make a few observations: First, using more augmented data appears to improve performance, with the lowest WER (a 5.3% absolute improvement over no augmentation) achieved with 500% additional data generated by chaining voice conversion and SpecAugment. Second, voice conversion augmentation appears to give similar improvements to SpecAugment when applied independently. Third, applying both voice conversion and SpecAugment improves performance more than either in isolation. This shows that the two approaches are complementary on English.

Does this improvement from voice conversion hold in all low-resource settings? No. With more real data (1 or 10 h), adding more voice conversion augmented data appears to hinder ASR performance. Voice conversion augmentation gives more speaker variability but it does so at the cost of intelligibility (Section 6.2.1); we suspect higher-resourced settings already have sufficient speaker variability, so the additional (potentially less intelligible) generated data contributes little. When using an LM for decoding (last 3 columns), the benefits of any augmentation appears greatly reduced. In very low-resource settings, however, an LM is often not available and voice conversion still gives a small improvement even though an LM is used.

We can now answer a part of our question posed at the start of this section: using voice conversion for data augmentation does help in very low-resource contexts on a *seen*

Table 6.3: WERs (%) on LibriSpeech test data for ASR models trained with increasing amounts of voice conversion (VC)- and SpecAug-augmented data, with and without the use of a 4-gram LM during ASR decoding.

		No LM			LM decoded		
Augmentation	Amount	10 min	1 h	10 h	10 min	1 h	10 h
None	0%	47.7	**30.4**	13.4	17.4	**10.6**	**7.5**
VC	100%	43.8	32.7	13.5	**17.2**	11.4	7.6
VC	500%	43.5	34.4	14.4	17.9	11.9	8.1
SpecAug	100%	44.3	31.8	**13.1**	18.8	11.2	7.6
SpecAug	500%	43.1	34.4	13.3	17.7	12.1	7.7
VC→SpecAug	100%	42.5	31.3	13.2	18.5	11.2	7.6
VC→SpecAug	500%	**42.4**	35.0	14.2	18.4	12.5	8.1

language. We now turn to the question of whether this holds for languages *unseen* during voice conversion training. As a reminder, our goal is to determine whether voice conversion can improve low-resource ASR and to validate its disentanglement properties, not whether it is the best ASR augmentation method.

6.2.3. Experiment 3: ASR augmentation with unseen languages

In this final experiment we use our English-trained voice conversion system to augment data from four unseen low-resource languages: Afrikaans, Setswana, isiXhosa, and Sepedi. The degree to which it improves ASR performance will give an indication of whether speaker identity is so strongly disentangled from content information that the content information can be swapped out with that from an unseen language, and the system should still produce reasonable quality speech when recombining the speaker and content information. For each language we use a 10 min training set with roughly equally sized validation and test sets covering the rest of the data in each respective language. There is no speaker overlap between any of the sets. For each language, we fine-tune `XLSR-53` [133] on the original 10 min of data, and compare this to training with varying amounts of additional voice conversion and VC→SpecAug generated data.

For all four unseen languages, Table 6.4 shows an improvement in ASR performance when 10 min and 50 min of additional data is generated using the English-trained voice conversion system. For isiXhosa, adding the augmented data gives a 7% absolute improvement in WER. Chaining voice conversion with SpecAugment, on the other hand, worsens performance. Additional experiments were done with 500% of additional voice conversion and VC→SpecAug data to see if trends followed similar to Section 6.2.2. However, W/CER got worse in all cases compared to training with just the original data. We also applied SpecAugment in isolation on Afrikaans, which again gave worse performance compared to doing no augmentation.

Table 6.4: ASR results (%) on test data of four low-resource languages when trained on 10 min of real audio data and different amounts of additional voice conversion (VC)- and VC→SpecAug augmented data. Sepedi* uses a non-default training procedure.

Language	Augmentation	Amount	WER	CER
Afrikaans	None	0%	52.3	15.9
	VC	100%	**48.9**	**15.0**
	VC	500%	61.5	19.5
	VC→SpecAug	100%	53.5	16.5
	VC→SpecAug	500%	83.9	33.1
Setswana	None	0%	68.9	26.1
	VC	100%	**65.9**	**25.1**
	VC	500%	78.2	31.1
	VC→SpecAug	100%	69.3	26.8
	VC→SpecAug	500%	87.0	40.3
isiXhosa	None	0%	63.2	15.5
	VC	100%	**56.5**	**13.8**
	VC	500%	65.0	16.4
	VC→SpecAug	100%	69.3	26.8
	VC→SpecAug	500%	89.4	28.2
Sepedi*	None	0%	92.6	50.7
	VC	100%	**52.8**	**19.9**
	VC	500%	56.4	21.5
	VC→SpecAug	100%	97.8	69.1
	VC→SpecAug	500%	98.4	73.7

Some of Table 6.4's results therefore contradict the results on English in Section 6.2.2: in Table 6.3 SpecAugment and voice conversion gave decent gains when applied in isolation and even more so when chained, while here on unseen languages SpecAugment hampers performance in all cases. This requires further investigation. Nevertheless, we see on all four low-resource languages that cross-lingual voice conversion augmentation *does* improve ASR performance.

It is worth briefly commenting on the results for Sepedi in Table 6.4, where performance is very poor without augmentation. We used the same 10 min training configuration for all languages, but on Sepedi, this configuration failed (WER $> 98\%$) regardless of whether data augmentation was used or not. We therefore did validation experiments and made minimal changes to the default configuration, only for Sepedi: we use 1k warmup updates and train for a total of 4k updates. Using this setting enabled the ASR model to learn, achieving the test performance in Table 6.4.

Taken together, the results in Table 6.4 confirm our hypothesis: a voice conversion system trained on one well-resourced language (English) can be applied cross-lingually to generate additional data for an unseen low-resource language, improving ASR performance. This also demonstrates that the voice conversion model proposed in this chapter strongly disentangles speaker identity from speech content, validation our design motivation around utilizing HGST (motivated in Section 6.1.2) and the quantization block Q (motivated in Section 2.5.1). Our efforts here seem to conflict with [145], where a similar approach was used but the voice conversion and ASR models were both trained and evaluated on English (Section 6.1.1). Our model, however, is different in that we do cross-lingual conversion, and also use much less data to fine-tune the ASR models – since our focus is on very low-resource ASR.

Relating this back to the high-level goal of this chapter, the experiments performed in this first part have shown that we *can* learn a disentangled latent space for speaker identity while also generalizing to unseen languages. This was done by focusing our design to disentangle (i.e. construct low-rank subspaces spanning common factors of speech variation) a latent space corresponding to speaker identity, with the help of HGSTs. Once trained on English, our model is performant enough to be useful in the practical task of improving speech recognition with unseen languages in very low-resource settings – a practical benefit indeed!

6.3. Non-standard downstream voice conversion tasks

In this second part, we wish to see whether the kNN-VC model proposed in Chapter 4 is useful in more out-of-domain downstream tasks. Recall this is part of our final research question (Section 1.3), where we want to know whether the techniques we have developed can be applied to practical and difficult speech processing tasks. Our investigations into unconditional speech synthesis are too restricted to yield practical improvements in hard downstream tasks, and the investigation into ASR is already a widely recognized practical task. So, here, we apply kNN-VC to four less-common, practical tasks to determine if our designs and motivation around using the disentangled space of SSL features leads to benefits in a variety of downstream synthesis tasks. Each of these tasks has an associated demo and audio samples, and we encourage the reader to listen along as we investigate each task in the rest of this chapter: https://rf5.github.io/sacair2023-knnvc-demo/.

How do we go about identifying hard downstream tasks? Recalling Section 2.4.1, older voice conversion models could only convert to and from speakers seen during training, while more recent models can also handle unseen speakers [110]. Further, there are early indications that recent models can even deal with non-speech inputs, e.g., using animal sounds as the target audio [152]. Looking at our kNN-VC model from Section 4.2, no part of the model's design explicitly assumes that the source or target data is regular human speech, and the SSL latent space it operates on is largely disentangled. So, pushing the limits of our motivations around disentanglement, we believe that kNN-VC would be particularly well-suited to non-standard downstream voice conversion tasks where the inputs/outputs are not regular human speech.

We look at four tasks. The first is stuttered voice conversion, where the target speech comes from a speaker with a stuttering speech impairment [153]. Second, we use kNN-VC for cross-lingual voice conversion where the source and target speakers use different languages [32]. Both the input and output languages are also unseen during training (this is similar to the previous investigation in this chapter, but here we stick to higher resource languages and settings). Thirdly – going even further out-of-domain – we consider completely non-speech input: we apply kNN-VC to musical instrument conversion, where the source and target "utterances" come from different musical instruments. Fourthly, we use kNN-VC to perform text-to-voice conversion where, instead of using a reference recording, the target voice is specified through a user-provided textual description, e.g., "an elderly woman with a velvety and resonant low voice". This last task is performed by training a small multi-layer perceptron to map a vector representation of the target speaker description to a distribution over known speakers for kNN-VC inference.

To understand whether kNN-VC translates to a *benefit* when applied to these tasks, we compare it to the prior state-of-the-art voice conversion model FreeVC [30], as we did for our standard voice conversion comparisons in Section 4.4. We find that kNN-VC retains high performance in stuttered and cross-lingual conversion, outperforming the baseline in terms of similarity to the target speaker. But quality is poorer in instrument and text-to-voice conversion, where kNN-VC doesn't consistently outperform the baselines; despite the degraded performance, the conversions are still compelling considering how far out-of-domain music and text are for a model that has only seen speech during training. Taken together, our experiments indicate that kNN-VC – based around utilizing SSL features – is largely applicable to non-standard downstream tasks, with a single model able to perform a range of tasks that it wasn't trained for, yielding improvements over prior models in some of the downstream tasks. However, there are still limitations in generalization when the model is presented with data very far from its training distribution.

6.4. Background of downstream conversion tasks

Summarizing how kNN-VC functions from Section 4.2, it is based on linear distance comparisons on disentangled features from the WavLM SSL model. The key to its success is that the SSL features are linearly predictive of various aspects of speech. kNN-VC uses the activations from layer six of WavLM, where linear distance comparisons are indicative of phonetic (i.e. linguistic content) differences [154]. Voice conversion is performed in kNN-VC by mapping each WavLM vector from the source utterance to the mean of its k nearest vectors from the reference utterance. The resulting vector sequence is then converted back into a waveform using a HiFi-GAN vocoder. If there are multiple reference utterances for the target speaker, the features from each are simply collated together into a pool of features known as the matching set. While the WavLM encoder and HiFi-GAN vocoder have only seen speech during training, kNN-VC's design does not make any specific design assumptions restricting it to speech. This, combined with the generality of the non-parametric kNN algorithm, makes kNN-VC a good choice for testing it on non-standard conversion tasks.

Table 6.5: Voice conversion setup for the four non-standard downstream tasks.

Task	Source	Target reference	Output
Stuttered conversion	Non-stuttered speech	Stuttered speech from desired speaker	Non-stuttered speech in desired voice
Cross-lingual conversion	Language A utterance	Language B utterance	Language A utterance in voice B
Instrument conversion	Music with instrument A	Music with instrument B	"Content" from A using instrument B
Text-to-voice conversion	Source utterance A	Text description of a speaker B	Utterance A in voice B

Given this setup, for each of the four tasks we explore, the conversion process can be formulated as a specific setting of a conversion model's source and reference data. Concretely, the specification for the source, reference and output of a voice conversion model for each task is given in Table 6.5. Here we give more background on each of the four tasks.

Stuttered voice conversion

In our first task, we want to convert input speech so that it inherits the voice of a speaker having a stuttering impairment (as shown in Table 6.5). Imagine that a student needs to create a presentation for their thesis, but the student suffers from a speech impairment. If we could solve the stuttered speech conversion task, then someone else could read a prepared script which would then be converted to the student's voice. This would result in an output resembling the student's voice but without stutters (since the source was read by a fluent speaker). Prior efforts to correct stuttered speech have used energy correlations of speech segments to attempt to remove repetitions from stuttered speech [155]. However, this approach attempts to directly edit stuttered speech into a non-stuttered form, while we focus on stuttered voice conversion which would enable a stuttering speaker to map their voice onto an existing fluent utterance. Related efforts in [153] attempt to train a generative adversarial network to generate fluent speech for a given set of speakers with a range of speech impairments. But the approach struggles with stuttered speech because the model fails to appropriately change the prosody. A voice conversion approach can address this since prosody is inherited from the source utterance (and not the stuttering reference).

Cross-lingual voice conversion.

Here the source and reference utterances come from different, potentially unseen, languages (Table 6.5). This is the most well-understood of the four voice conversion tasks, with the Voice Conversion Competition in 2020 [32] having a dedicated track for cross-lingual conversion. Converting speech between languages unseen during training would improve access to speech technologies by speakers of low-resource languages, as we saw in the first part of this chapter. Newer any-to-any conversion models like FreeVC [30] and kNN-VC can convert to unseen languages, but still struggle with completely unseen languages (and typically perform very badly in very low-resource unseen languages, hence our exclusion of their use in the first part of this chapter). Here we are particularly interested in seeing whether existing voice conversion models can generalize to higher resource unseen languages. In our experiments here, we therefore use kNN-VC and FreeVC (both trained exclusively on English) to convert utterances between several non-English European language pairs.

Musical instrument conversion.

In this difficult task, we apply voice conversion to convert music played by one instrument to sound as though it is played by another. In this case, the source and reference are both non-speech pieces of music from a single instrument (Table 6.5). Most prior work in this area – unsurprisingly – uses models trained on music. The goal is typically to give a user control over the style of the synthesized audio. E.g. [156] uses a variational autoencoder to model music style, while more recent studies like [157] use differentiable digital signal processing methods to allow a user to modify the output audio. Our goal is not to achieve state-of-the-art music synthesis, but rather to assess how well a voice conversion approach based on disentangled SSL features (and trained on speech audio, not music) is able to generalize to hard out-of-domain tasks. So, for instrument conversion, we still use kNN-VC and compare it to FreeVC – both models having only seen speech data.

Text-to-voice conversion

The last task is one where, instead of using a reference utterance to specify the target speaker, a textual description is provided, e.g. "an older man with a British accent and a pleasing, deep voice". This is known as text-to-voice conversion or prompt-to-voice conversion [33]. This task is much less explored than the others. It was first introduced by Gölge and Davis [33] in 2022, who trained a specialized prompt-to-voice model. Their work is proprietary and they also specifically considered a text-to-speech task where the target voice is provided as a text description rather than a reference utterance. This motivates our introduction of a simple extension to kNN-VC to allow for textually described target voices in a voice conversion setting. This can be used to e.g. enhance privacy by ensuring that we are not cloning a real person's voice, but the voice associated with an arbitrary text prompt (which has substantially fewer privacy concerns).

6.5. Downstream task experiments

6.5.1. Stuttering experiments

For stuttered voice conversion, we perform two experiments: a comprehensive large-scale evaluation and a smaller-scale practical conversion of monologue recordings. The former allows us to quantify kNN-VC's performance at stuttered conversion using the standard objective metrics of voice conversion, while the latter shows that it can be used in a real-world practical setting.

Performance metrics

When evaluating voice conversion performance, two categories of metrics are used as in Section 4.3.3: ones which measure how *intelligible* the output is (i.e. how much of the linguistic content it retains from the source), and ones which measure the *speaker similarity* to the target speaker [32].

Summarizing Section 4.3.3, to measure intelligibility we compare the transcript of the source utterance to the transcript of the converted output (obtained using an existing high-quality ASR system). A word/character error rate (W/CER) is then computed between the transcript of the converted output and the source: the lower the error rate, the better the conversion (i.e., more intelligible). The automatic speech recognition system we use is the pretrained Whisper `base` model using a decoding beam width of 5 [59].

For speaker similarity, we compute an equal error rate (EER) as in Chapter 4. It is computed by calculating speaker similarity scores between pairs of real/real and real/generated utterances. The real/real pairs are assigned a label of 1, and real/generated pairs a label of 0. An equal error rate is then computed using the similarity scores and the labels. The speaker similarity score between two utterances is defined as the cosine distance between speaker embeddings computed using a pretrained speaker embedding model [119] (same as our metrics computed in Chapter 4). The higher the EER, the less able we are to distinguish between the target speaker and converted output (i.e., better speaker similarity), up to a theoretical maximum of 50%.

Stuttering events in podcasts

We evaluate kNN-VC and FreeVC on a combination the LibriSpeech's `dev-clean` dataset [74] and the Stuttering Events in Podcasts dataset [158]. LibriSpeech consists of hundreds of hours of read books in English by speakers without any speech impairment, while the Stuttering Events dataset consists of data from podcasts where stuttering events have been annotated. Both datasets contain speech from many speakers, none of which are seen by kNN-VC or FreeVC. Using the annotations in the Stuttering Events dataset, we trim each utterance to only those segments where there is a sound or word repetition, i.e., we filter out all non-stuttered speech and only use stuttered instances as reference audio.

As explained in Section 6.4, we want the generated speech to match the identity of a reference speaker that stutters, but the content should match a fluent source utterance. So, for our evaluation, we convert each of the 200 utterances from LibriSpeech's `dev-clean` subset from the kNN-VC paper [154] to ten random speakers from the Stuttering Events dataset. Since each reference speaker has several short clips with stuttered speech, we sample up to 30 stuttered reference clips for each target speaker[4]. These clips then serve as the matching set for conditioning kNN-VC when it is presented with a source utterance from LibriSpeech (see Section 6.4). For the FreeVC [30] baseline, we use the same input utterances but compute the mean speaker embedding

[4]Note the many utterances for each speaker here, this is what allows us to use kNN-VC, FreeVC, and the EER measure as from Chapter 4. If we only had 1 utterance for some speakers, we would be in the very low-resource setting like the last ASR experiment and would need to resort to the voice conversion model developed earlier in this chapter and its associated speaker similarity metric.

Table 6.6: Stuttered voice conversion performance, where the source is fluent speech and the target reference is from a speaker with a stuttering impairment.

Method	WER ↓	CER ↓	EER (%) ↑
dev-clean topline	4.63	1.60	—
FreeVC	**6.59**	**2.57**	25.6
kNN-VC	14.61	7.16	**46.7**

of the stuttering clips to serve as the target. This is done because FreeVC's target conditioning mechanism doesn't use a matching set; instead, it uses a trained speaker encoder which maps a reference utterance to a single speaker embedding [30].

The result of this evaluation is presented in Table 6.6. The topline is an approach where we perform no conversion and simply evaluate the baseline transcription error rates of the Whisper ASR model on the ground truth source utterances. We make two key observations from the results. First, voice conversion can be used effectively for this task, with high EER (good speaker similarity) and low W/CER scores (good intelligibility). Second, there is a tradeoff between kNN-VC (designed around using implicitly disentangled SSL features) and FreeVC (designed around explicit techniques to disentangle speaker identity): FreeVC retains more of the desired linguistic content, with W/CERs that are less than half of the rates achieved by kNN-VC. Conversely, kNN-VC is substantially better in matching the target speaker, with a much higher EER. We suspect that the better speaker similarity of kNN-VC is due to the matching set mechanism that is more robust to non-standard speech compared to the speaker embedding approach of FreeVC. However, the matching approach of kNN-VC might also be what causes poorer intelligibility because, recalling the logic of kNN-VC in Section 4.2, the matching set here contains several non-fluent phones and biphones.

Practical evaluation

Having objectively evaluated kNN-VC's performance, we now apply it in two practical scenarios in a qualitative evaluation. First, we apply it to the example already mentioned in Section 6.4, where we want to generate the audio of a thesis presentation for a student who suffers from a stuttering impairment. The source to kNN-VC is provided by a fluent speaker reading a script provided by the student. We then perform a few conversions. In the first, we used the reference data provided by the student, but the parts of the reference where stuttering occurs were removed by hand. For the second and third samples, the reference waveform is simply the raw stuttered recordings with heavy stuttering and silences. This makes the task slightly harder for the model since the reference speech is further out-of-domain. We also apply kNN-VC using stuttered speech from an actor as reference, specifically the actor Colin Firth acting as King George VI in the movie "The King's Speech". In both cases, to further improve output quality, we also apply the `voicefixer` upsampling model to remove some conversion artifacts [159]. We encourage the reader to listen to conversion samples (with and without voicefixer) at https://rf5.github.io/sacair2023-knnvc-demo/. The samples provide further evidence that our kNN-VC method can generalize to this

Table 6.7: Cross-lingual voice conversion performance on the Multilingual Librispeech test dataset.

Method	WER ↓	CER ↓	EER (%) ↑
test set topline	21.5	7.1	—
FreeVC	34.1	13.5	7.7
kNN-VC	**33.9**	**13.1**	**25.0**

practical out-of-domain use case, and the disentangled nature of the SSL features used does allow the model to accurately retain the target speaker's identity despite it being different from regular speech.

6.5.2. Cross-lingual voice conversion

In cross-lingual voice conversion, we want to convert to a target speaker talking a different language from the source. The languages and identities of the source and target speakers are unseen during training. We again apply kNN-VC as our main model and use FreeVC as a baseline. Because the inputs and outputs in this task are still human speech, we use the same intelligibility and speaker similarity metrics as in Section 6.5.1.

Experimental setup

We conduct a large-scale evaluation on the Multilingual LibriSpeech dataset [160]. The dataset consists of thousands of hours of audiobook recordings across eight European languages. For evaluation, we sample 16 utterances from three random speakers from the test subset of each language, yielding 384 evaluation utterances. Evaluation involves converting each of these utterances to every other speaker in the evaluation set, giving a total of just under 9000 unique source-target evaluation pairs covering all possible language combinations.

For measuring intelligibility as described in Section 6.5.1, we can still use Whisper `base` to transcribe the original and converted utterances because Whisper is a multilingual speech recognition model that can be presented with input in any of the European languages considered here [59]. Similarly, speaker similarity is evaluated using the same speaker embedding model, since these models are trained to ignore linguistic content and only extract speaker information. We can therefore calculate an EER exactly as in Section 6.5.1 and 4.3.3.

Results

The cross-lingual voice conversion results on the Multilingual LibriSpeech dataset are presented in Table 6.7. Unlike the stuttered conversion experiments, we see that kNN-VC is consistently more intelligible and more similar to the target speaker than FreeVC.

How do cross-lingual conversions compare to converting between speakers in English (the language on which both FreeVC and kNN-VC are trained)? If we compare the results in Table 6.7 to the monolingual scores reported in Section 4.4, we see that both the intelligibility and speaker similarity are lower when performing cross-lingual voice conversion compared to monolingual conversion. This degradation is to be expected, given that these voice conversion models might not have seen all the phones necessary for converting between unseen language pairs. E.g., when converting Polish to Spanish with kNN-VC, the Spanish matching set might not have all the phones necessary to represent the source Polish utterance.

Nevertheless, for kNN-VC in particular, the cross-lingual performance is still compelling, with the CER and EER scores in Table 6.7 coming close to FreeVC's performance on English [30]. This shows that voice conversion models trained only on English can be applied successfully to unseen languages. Critically for our research question for this chapter, the table provides good evidence that for prior models like FreeVC, linguistic content and speaker identity is still largely entangled. Meanwhile, kNN-VC can perform the conversion to a much better degree, indicating a superior disentanglement of speaker identity and linguistic content. The key difference between kNN-VC and FreeVC is, as reasoned in Chapter 4, that of implicit disentanglement (used by kNN-VC with SSL features) and explicit disentanglement (used by FreeVC) of speaker identity. The results of Table 6.7 provide positive evidence that designing around using learnt disentanglements *does* provide greater generalization capability – in this case to unseen languages. We encourage the reader to listen to a selection of conversion samples on the demo website.

6.5.3. Musical instrument conversion

Moving on to harder tasks, we now consider a conversion problem involving non-speech audio. Specifically, we want to see whether we can use kNN-VC to convert a piece of music played with one instrument to sound as though it is played with another – **without fine-tuning or adapting the model**. To this end, we use a dataset consisting of several music pieces, each played on a single instrument. We then attempt to convert a song from one instrument to another by using all the audio from the target instrument as our matching set in kNN-VC. This effectively treats the recordings from the target instrument as "utterances" from a target "speaker".

Experimental setup

We use the Kaggle Musical Instrument's Sound Dataset [161]. The dataset consists of short music pieces played with one of four instruments: drums, violin, piano, or guitar. For our music conversion evaluation set, we only consider recordings that are between 10 and 90 seconds long. Evaluation involves converting each recording in the test set to every other instrument, resulting in 153 synthesized output audio recordings that are evaluated. For kNN-VC, all the recordings from the target instrument are used as the matching set. For FreeVC, a mean "speaker" embedding is obtained from the target instrument audio.

The intelligibility and speaker similarity metrics from Section 6.5.1 are not applicable to instrument conversion. We therefore use a metric from the music synthesis domain: Fréchet audio distance (FAD) [162]. This metric uses a music classification model (trained to classify the tags of scraped YouTube videos) to compare two recordings, giving a single number which roughly indicates how similar the music in the recordings is [162]. This is achieved by computing

Table 6.8: Musical instrument conversion performance on the Musical Instrument's Sound Dataset, measured in FAD [162] (lower is better).

	Target instrument FAD ↓			
Method	Drums	Violin	Piano	Guitar
FreeVC	20.11	**21.07**	**20.03**	23.22
kNN-VC	**9.81**	22.44	23.95	**18.01**

an "Inception score" between the intermediary features produced by the classifier for two sets of audio inputs. In our case we measure the FAD between the converted outputs and the ground truth recordings from the target instrument. E.g., to measure FAD for drums, we compare all the outputs where drums are the target instrument to several real recordings of drums. We use an open-source implementation of FAD[5].

Results

The musical conversion performance for each instrument is presented in Table 6.8. FAD values greater than 10 are considered to be very poor [162]. So we can see that FreeVC performs very poorly, regardless of the instrument. kNN-VC also performs poorly for most target instruments, except when converting to drums where the best overall FAD is achieved. While a FAD of 9.8 is still poor compared to typical scores in music enhancement [162], qualitatively kNN-VC's conversion to drums sounds much closer to the target than any of the other conversions.

We are still unsure why kNN-VC performs so much better when converting to drums. We speculate that this might be because drum audio has lower frequency content that might be in a similar range to that of human speech – at least compared to the high-frequency content from guitars, pianos, and violins. The encoder and vocoder of kNN-VC have only seen speech, and it is therefore likely that the WavLM features would be better at capturing instrument information with a similar frequency range to speech.

Overall, the results indicate that the kNN-VC and FreeVC **cannot** effectively perform instrument conversion, except in specific instances. This therefore remains an open question for future work; one idea would be to incorporate some of the techniques used in music enhancement and synthesis [156,157]. However, remember our goal is to assess whether we can apply our model based around continuous disentangled features (kNN-VC) to hard unseen tasks (it can) and to test whether it yields benefits over existing voice conversion models on these tasks (comparing to the baseline in Table 6.8, it does).

6.5.4. Text-to-voice conversion

The last task we look at is text-to-voice conversion, where the target speaker is specified using a textual description instead of a reference utterance. Unlike the previous tasks, kNN-VC cannot be directly used with text inputs. So we propose a small extension.

[5]https://github.com/gudgud96/frechet-audio-distance

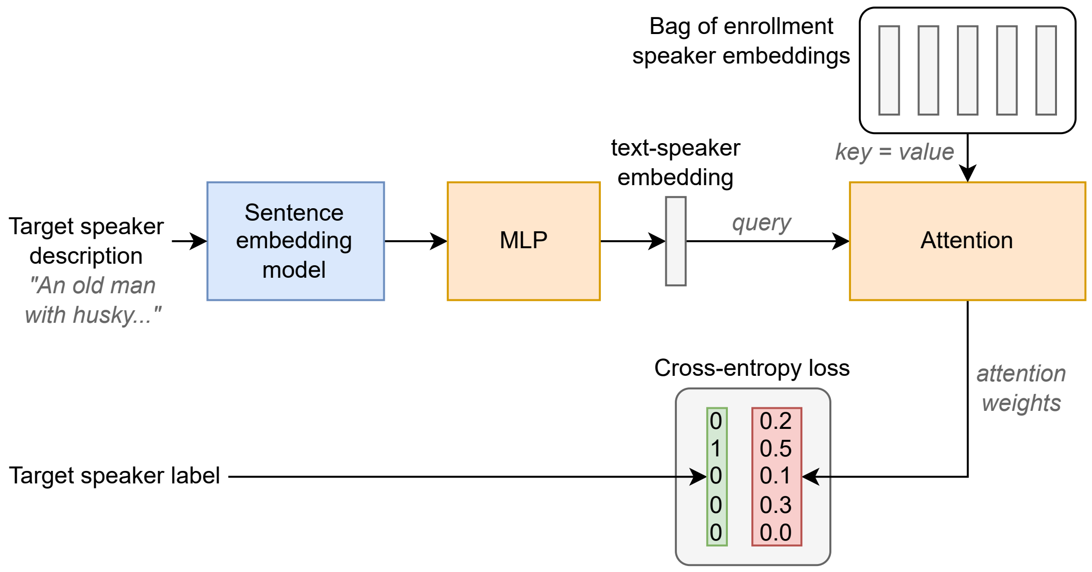


Figure 6.3: For text-to-voice conversion, we use an approach where a text description of a voice is converted to a distribution over enrollment speakers. Only the multi-layer perceptron (MLP) and attention layer (Section 2.3.5) are trained; the sentence embedding model and speaker embeddings are fixed. The parameters are optimized to predict the target speaker label from the provided textual description.

Extending kNN-VC for textual voice descriptions

We extend kNN-VC to work with text descriptions using an insight from Cheng [163]: during synthesis with kNN-VC, we can construct combinations of voices by linearly interpolating the converted WavLM features between those from different target speakers, before vocoding. We therefore rephrase the text-to-voice conversion task as *finding a probability distribution over a set of enrollment speakers that best matches the provided text description.* If we knew this distribution, we could use the probabilities as weighting coefficients for interpolating the converted output to best match the target speaker description.

Figure 6.3 illustrates our approach for predicting the distribution over speakers from a target speaker description. We construct a small text-to-voice matching network that maps a text description to a distribution over a set of enrollment speakers. The network consists of three components:

1. A sentence embedding model [164], which converts a text description into a single vector capturing the target speaker identity. We use the pretrained model from [164] directly.

2. A bag of speaker embeddings computed using a pretrained speaker embedding model [34]; a single speaker embedding is obtained for each enrollment speaker by averaging the embeddings from that speaker's utterances. We use an open-source pretrained speaker embedding model.[6]

3. A multi-layer perceptron (MLP) with three layers and an attention [58] head. The MLP maps the sentence embedding [164] to a "text-speaker embedding", which is used as the

[6]https://github.com/RF5/simple-speaker-embedding

query over the bag of enrollment embeddings (acting as keys and values in the attention operation, see Section 2.3.5 for details of the attention mechanism).

The output of this small matching network is the attention weights over the keys/values, defining a distribution over the enrollment speakers. During training, the model is optimized using a cross-entropy loss between the one-hot target speaker label and the predicted distribution over speaker embeddings. During inference, the distribution is used as the weighting for interpolating the converted WavLM output features of kNN-VC before vocoding.

Experimental setup

We use the LibriSpeech [74] `train-clean-100` subset for this experiment. We manually labelled 100 speakers with a textual description of the speaker's voice, e.g., "a young woman with a squeaky and animated voice, speaking in a rapid and highly expressive manner" [7]. We split this set into 90 training and 10 test speakers. The MLP and attention layer from Figure 6.3 is then trained for 300 steps using AdamW [165] with a learning rate of $5 \cdot 10^{-5}$ and weight decay of $3 \cdot 10^{-3}$. The MLP consists of three layers with LeakyReLU activations and layer normalization [128] between each layer. Since the sentence and speaker embeddings respectively have 384 and 256 dimensions, the output dimensions from the three MLP layers are 1024, 768, and 256. The attention is a standard scaled dot-product multi-head attention with 16 heads [58].

The evaluation task involves converting the same set of `test-clean` utterances as in Section 4.4 to each of the ten test speaker descriptions. First, the network in Figure 6.3 maps the description of each test target speaker to a distribution over the 90 enrollment speakers. Then we perform kNN-VC conversion to each enrollment speaker as target. Finally, we interpolate the output WavLM features in the ratio provided by the attention weights of the model in Figure 6.3 before vocoding to achieve the final converted output. Using the set of converted outputs, speech inputs, and original reference utterances, we assess performance with the same metrics as in Section 6.5.1.

Results

The text-to-voice conversion results are presented in Table 6.9. We see that intelligibility is not harmed when using textually described target voices – it actually improves slightly compared to the 1-minute topline. This makes sense since the target is based on a combination of the training speakers, for which we have much more data to use as matching data compared to the topline. However, this comes at a cost: the similarity to the target speaker is much worse compared to the topline (lower EER). While not certain, we hypothesize this is largely due to the highly subjective nature of mapping utterances from a speaker to a textual description of that speaker's voice. E.g. what one person might label as a "squeaky and animated voice" might differ greatly from how another person describes the same voice. This is compounded by the sensitivity of the speaker embedding model used to evaluate EER, since it is trained to be sensitive enough to

[7]We thank Chris-Mari Schreuder for the help in producing the textual annotations of the voices in this experiment.

Table 6.9: Text-to-voice conversion performance on LibriSpeech, where unseen source speakers are converted to target speakers specified with a textual description. Topline performance when 1 minute of speech audio is used to specify the target speaker (instead of text) is provided as comparison.

Method	WER ↓	CER ↓	EER (%) ↑
1 min speech kNN-VC topline	7.73	3.09	**35.05**
kNN-VC three-layer MLP	**6.32**	**2.51**	3.15

discern even similar speakers of the same gender and age. Overall, the results indicate that we **can** perform text-to-voice conversion to some degree, but there is still more work required to accurately map to the desired target speaker.

6.6. Summary and conclusion

This chapter aimed to answer our fourth research question: given the techniques and knowledge we have developed in the previous chapters, can our ideas and techniques around disentanglement translate to benefits on downstream speech synthesis tasks? To answer this we identified several practical downstream tasks associated with voice conversion. We then attempted to apply the ideas of disentanglement we have developed so far to each one, and then assess whether our methods led to benefits over baselines.

The first task looked at using voice conversion for ASR data augmentation to improve ASR performance in very low-resource settings. For this setting, we investigate whether we can design a voice conversion system that performs and generalizes well enough to be useful for very low-resource ASR augmentation, without existing disentangled features from large SSL models trained on massive datasets. With a design based around HGSTs, we showed that a voice conversing system trained on one well-resourced language can be used to generate additional training data for unseen low-resource languages. When labelled resources are very limited (roughly 10 min), ASR performance improved on all four low-resource languages considered. Additional experiments also demonstrated SpecAugment's inability to improve ASR performance in these very low-resource settings, while our method did yield benefits.

Our goal with this first task was to show that ASR improvements are possible with voice conversion based augmentation. While several ablations were performed to validate speaker disentanglement, we did not exhaustively explore how different design decisions within the voice conversion model impacts downstream ASR performance. Also, the interplay between voice conversion and traditional augmentation methods remains largely unexplored – possible directions for future work. However, we did show such augmentation was (a) possible, and (b) did lead to improvements on this practical task.

We then went on to explore four additional non-standard downstream tasks involving voice conversion. For these tasks, we decided to use the kNN-VC model developed in Chapter 4 to truly test its usefulness and generalization in practical settings. In stuttered voice conversion and cross-lingual voice conversion, kNN-VC retained high performance, achieving convincing results compared to an existing voice conversion baseline. For music conversion, however, the results were more mixed, with kNN-VC being largely unable to produce high-quality conversions except when converting to drums. The results for textually described voice conversion were also mixed: the output was still intelligible but was a poor match to the target speaker compared to a standard system using audio rather than text to specify the target speaker, albeit this may be due to the fuzzy nature of textual voice descriptions.

Taken together, the experiments in this chapter showed that our ideas and methods around disentanglement *can* be applied to useful downstream tasks. When we apply our SSL-based voice conversion model (kNN-VC) to out-of-domain downstream tasks without additional training or fine-tuning, it *does* improve or achieve similar results compared to prior methods. And, when attempting to reconcile the scale limitation of Chapter 3 with the impressive disentanglement afforded by large SSL models, we showed that we can develop a voice conversion model on a modest amount of English data without using existing SSL models. This model, when applied to the hard task of ASR augmentation in low-resource settings, *does* yield improvements over existing augmentation methods. This largely validates our overarching motivation around the benefit of designing-for-disentanglement (Section 1.1) and fulfills in part our research aim. But, it is not without limitations: the voice conversion system developed without SSL features is fairly complex and does not have state-of-the-art voice conversion performance, and applying our simple kNN-VC to certain downstream tasks (e.g. music conversion) did not perform very well. Getting these methods to perform better while remaining simple (i.e. making fewer assumptions about speech during model design) is another important direction for future work.

Chapter 7

Summary and conclusions

The goal of speech synthesis is to produce systems that can convincingly generate human speech, and a key part of this is generalization. Models that only work on narrowly defined data close to its training distribution fail when applied to many real-world scenarios. This thesis investigates a key facet limiting current generalization: the tension between the continuous nature of speech, and the explicit factorizations and disentanglements imposed on speech in the current paradigm in many speech processing tasks. The overarching aim is to improve the generalization and performance of speech synthesis and processing systems by attempting to learn and utilize continuous, implicitly disentangled representations of speech. To this end, we performed several investigations and made contributions in four main focus areas: (i) unconditional speech synthesis, (ii) voice conversion, (iii) denoising diffusion speech recognition, and (iv) practical downstream applications of speech synthesis.

This chapter first summarizes our findings, covering our four research questions and the implication in terms of our overall research aim. We then go on to discuss extensions and improvements to our work, both proposed future ideas and efforts by other authors building on our work. Finally, we provide a discussion on the unresolved areas of the continuous-discrete and explicit-implicit disentanglement tension, and speculate on the nature of some of the remaining problems and limitations.

7.1. Findings

The main findings and outputs of this thesis are given below, ultimately showing that our goal – exploiting implicit disentanglement to improve speech processing tasks – is largely fulfilled. To fulfill our research aim we needed to validate our motivation in Section 1.1, which centers around two ways to overcome the explicit vs implicit tension: designing models which *implicitly* disentangle speech (i.e. do not include domain-expert knowledge about how speech should be factorized), and utilizing/learning continuous latent spaces where linear directions correspond to high-level factors of speech variation (e.g. phonetic differences). So, we proposed specific research questions and associated investigations in an attempt to probe and validate our reasoning that, if we design models around implicitly disentangled latent spaces, they should offer improved generalization and control compared to the current paradigm focused around explicit disentanglements of speech.

7.1.1. Research questions

To achieve our overall research aim, we separated it into (1) understanding the existing methods for disentanglement, (2) attempting to learn more ideal continuous disentangled latent spaces of speech, and (3) attempting to apply these representations to various speech processing tasks to identify whether they yield benefits compared to existing methods. The first point is addressed in the background chapter, and the latter two are answered by investigations into four research questions:

Research question 1: *Why can we not simply apply disentanglement techniques from the image domain?* We investigated this in Chapter 3. Observing that a related version of our motivation has been validated in the image domain, we attempted to adapt new techniques from image disentanglement to speech by proposing a novel model for unconditional speech synthesis. The model, ASGAN, adapted techniques from the StyleGAN3 [18] model for speech synthesis. In its design, we showed that we *can* apply techniques from the image domain to speech synthesis, however not directly. Several modifications and new additions, such as adaptive discriminator augmentation, were needed to train the model successfully. Evaluating it against several prior state-of-the-art models on unconditional speech synthesis, we showed an improvement in most metrics compared to prior models.

Most importantly, because of ASGAN's design, each speech utterance corresponds to a point in a latent space. Through evaluations, we showed that this space is highly disentangled, to the extent that we can apply ASGAN – without any fine-tuning – to several other speech processing tasks that it never saw during training. This is a key validation of our research motivation: focusing on disentanglement *does* lead to benefits in terms of generalization. However, our model – like others in the speech domain – has a grave limitation: it only works in restricted settings (fixed duration audio). This is not an issue in the image domain (e.g. fixed resolution images). So, purely adapting image domain techniques is insufficient to fully solve the problem of learning disentangled representations for speech synthesis, but it does yield some benefit.

Research question 2: *Given existing models which have learnt to implicitly disentangle speech for analysis, why can we not use their disentangled representations directly to solve problems such as voice conversion?* Chapter 4 attempts to answer this by first identifying that no prior voice conversion models make use of the linearly disentangled nature of features produced by SSL models (Section 2.5.1), instead opting for increasingly more complex methods to explicitly disentangle speaker identity from the rest of speech. Seeing the gap, we proposed kNN-VC, a simple voice conversion algorithm which performs the conversion by a simple nearest neighbors comparison on SSL features, where we made the key insight that linear distances between certain SSL features reflect *phonetic differences* (i.e. linguistic content).

Through several evaluations in Section 4.4 we demonstrated how this simple model is either on par or superior to all prior voice conversion models. Moreover, because the conversion component is a non-parametric kNN algorithm, our investigation shows that – when we have continuous, strongly disentangled features provided by SSL models – there is no need for complex and comparatively fragile conversion networks. Simple linear distance comparisons in a (linearly) disentangled latent space are all that is needed. This chapter validates the key part of the motivation in Section 1.1, where we reasoned that designing around implicit disentanglements of speech should yield improvements over explicit disentanglements. Namely, we showed that when we appropriately use strongly disentangled continuous representations of speech, we *do* observe improvements in speech synthesis tasks (namely voice conversion), where performance is improved while also substantially simplifying the model design compared to prior explicit disentanglement methods.

Research question 3: *Given these disentangled SSL features, can they be applied to improve tasks necessitating a discrete output, such as ASR?* Some tasks in speech processing require discrete units, but perhaps applying continuous methods, such as the SSL features from the previous investigation, can still yield improvements. Chapter 5 investigated this by integrating two key techniques to ASR: pretrained continuous SSL features and denoising diffusion. Concretely, we proposed the first formulation of ASR as a conditional discrete diffusion task, where we iteratively refine a transcript using a multinomial DDPM task, conditioned on frozen, continuous SSL features. This is drastically different to the typical way ASR is formulated, typically ignoring any use of disentangled features and trained either as a sequence-to-sequence task (as in Whisper [59]) or a CTC task (Section 5.1.1).

We compared our new ASR model, TransFusion, to existing state-of-the-art CTC-type models on a standard ASR benchmark. Despite this being the first foray into this new framework for ASR, we demonstrated comparable performance to existing models (Table 5.1) and showcased some of the advantages allowed during decoding in this new ASR formulation. Overall, our design and experiment results indicate that continuous techniques (implicitly disentangled SSL features in particular) *can* be applied to discrete-output tasks, yielding similar performance to existing methods. However, it is not perfect: our model is still not superior to existing CTC-type models. Given that this is the first work into conditional DDPM-based ASR, and the amount of effort that has gone into perfecting CTC and sequence-to-sequence type models, it is largely to be expected. Importantly, some trends from the results in Table 5.1 indicate that much more powerful decoding methods may be possible for TransFusion, leaving much room open for future work.

Research question 4: *Can our focus and motivation around disentanglement yield benefits in practical downstream speech synthesis tasks?* While the prior investigations are academically useful, a key part of our goal is to be practically useful: for our motivation to truly be verified, we must be able to show improvements or generalization on less well-explored, practical tasks. In Chapter 6 we explored two sets of tasks. In the first, we are interested in whether voice conversion can be used to improve ASR performance in very low-resource settings on South African languages. Due to the low-resource constraints, we combined the lessons of the prior chapters to develop a new voice conversion model that is applicable cross-lingually to unseen low-resource languages.

The second set of downstream tasks looked to apply our existing kNN-VC model from

Chapter 4 to four hard tasks to truly test its generalization capabilities without any additional training or fine-tuning. In the first task, we showed how kNN-VC can be used to assist speakers with stuttering impairments by correcting stuttered speech through voice conversion, and to do this with superior performance than existing voice conversion models. Similarly in a cross-lingual setting, kNN-VC retained high performance and achieved on-par or superior results compared to a baseline. For further out-of-domain tasks, like music instrument conversion and textually described voice conversion, kNN-VC struggled. It was only able to have moderate performance converting to drums for music conversion, and failed to achieve large similarity to the target speaker in textually described voice conversion (which may be due in part to the fuzzy nature of textual voice descriptions). Taken together, these experiments indicate that we *can* use our methods, particularly for voice conversion, on hard, unseen settings. And, in many of them, it *does* yield an improvement in performance or capabilities compared to existing methods.

7.1.2. Disentanglement and the continuous-discrete tension

With the above investigations, we can assess our motivation from Chapter 1 – whether systems designed around the implicit disentanglement of speech have better generalization and controllability compared to methods based on explicit disentanglement. Recall from our discussion in Section 2.2, we know there are two areas of tension: the difficulty of learning probability representations of high-dimensional data, and the tension between explicit (as is the current paradigm) and implicit disentanglement (as is the case in other, better-scaled machine learning fields) – the second tension is our primary focus throughout this thesis. With this in mind, we have been able to bridge the tension between continuous and discrete approaches to speech to a moderate degree.

The work of Chapter 3 made inroads into learning probability estimates for high-dimensional data (the $\mathbf{z} \to \mathbf{w}$ latent space sampling) by utilizing a GAN setup with special training techniques. In addition, it did not introduce explicit factorizations of speech in model design, but attempted to learn them implicitly in the $\mathbf{w}$ latent space. However, it did not scale and so neither tension is fully solved. Chapter 4 had more success in overcoming the second tension by building a conversion system that did not make complex domain-expert assumptions about speech, and we did show throughout Chapter 4 and the later parts of 6 that exploiting disentangled latent spaces does lead to generalization benefits across a range of tasks.

Lastly, the models developed in Chapter 5 and the first part of Chapter 6 also made slight inroads in addressing the explicit vs implicit tension. We did this by demonstrating how using implicitly disentangled latent spaces (SSL features for TransFusion, HGST output space for ASR augmentation model) led to new ways of approaching tasks with considerable performance. Looking back at our primary aim, the findings above allow us to claim that we *have* developed methods that – in part – bridge the explicit vs implicit disentanglement tension for speech processing. Additionally, exploiting the benefits of continuous latent spaces *does* – in many cases – lead to improvements in various tasks.

7.2. Future directions

While the investigations of this thesis have been moderately successful, many questions and areas of improvement remain. The models we have developed have their shortcomings, and there are several areas where they may be improved – both to improve performance and gain a better understanding of how we might resolve the central tensions of this thesis.

7.2.1. Extensions of our models and framework

The unconditional speech synthesis model developed in Chapter 3 still only functions in a fairly restrictive data setting with fixed vocabulary and utterance duration. Even with substantial effort to attempt to scale the ASGAN model directly, it still is not yet on par with GSLM speech synthesis models (unconditional speech synthesis' discrete counterpart, see Section 2.4.2). And, from our investigations in Chapter 4 and 5, we know that SSL models *can* learn disentangled representations at scale. So, an obvious direction for future work is to attempt to combine the generality of the unsupervised speech synthesis task with the scalability of SSL architectures and losses. If a way could be devised whereby we retain the scalability of SSL models, but with the ability to *sample* new utterances (i.e. to keep within the speech synthesis paradigm), then we may see great improvements in several speech synthesis tasks.

Within voice conversion, our contribution of kNN-VC has received substantial interest. Since its publication, several other authors have attempted to rectify its chief limitation: the need for more than roughly 5 min of reference speech to obtain a state-of-the-art clone of the target voice (Figure 4.3). Concretely, [121] proposed an extension to kNN-VC whereby they train a small 'phoneme hallucinator' module to artificially generate more SSL features for the matching set to obtain a more diverse coverage of phonemes present in the input speech, thereby allowing conversion with less reference speech.

Sha et. al. [120] proposed another extension whereby they feed the vocoder additional information about desired signal properties (e.g. pitch contour) to assist the vocoder in synthesis when the matching pool is limited. They find this also allows kNN-VC to work for singing voice conversion (as pitch conditioning can follow a desired melody). [166] even used the idea of using linear combinations over speakers in Section 6.5.4 to adapt kNN-VC for speaker anonymization and privacy protection. Even with these extensions, kNN-VC still has a common limitation that still needs to be addressed: it still requires a vocoder to be trained on SSL features. This ties in with the limitation of ASGAN – while we have disentangled SSL features here, we do not have a ready way to synthesize them, so a vocoder must be trained. The vocoder performance is not perfect, and could likely be substantially improved to make it more robust to harder data (such as for music conversion in Section 6.5.3). Future work could focus on attempting to modify newer spectrogram vocoder architectures (e.g. BigVSAN [167]) for SSL features.

Last, the TransFusion model developed in Chapter 5 has the most clear area for future work: decoding techniques. As mentioned in the conclusion for Chapter 5, our investigation is the first to phrase ASR as a conditional DDPM task. The decoding methods for a transcript-generating DDPM are still fairly rudimentary, particularly when compared to CTC-based ASR models, where there has been significant prior effort into optimizing these models and their decoding, integration with LMs, etc. So, the clear avenue for future work is to investigate and develop new and better decoding techniques, to make SSL-conditioned DDPM ASR systems better compete with (and even surpass) CTC-based ASR models.

7.2.2. Unresolved areas of the continuous-discrete tension

Looking back at the continuous-discrete tension explained in Section 2.2 and the results of our work in this thesis, it is clear that a substantial part of this tension remains. Given the limitations of ASGAN and our attempt at learning disentanglement for voice conversion in Chapter 6, it is clear that we have not fully solved the difficulties around learning probability estimates for high-dimensional signals such as speech. There is still work to be done, and it remains an open question how to design systems that can model the probability density of arbitrary speech signals, and ideally sample them in a controllable way as well. Perhaps the goal might even be to simply follow the text-domain's progress verbatim? E.g. perhaps we can simply scale discrete GSLM models to many tens of billions of parameters, and then verbally ask them if a recording is likely on a scale of 1-10, forgoing any consideration of disentanglement, learnt of otherwise, and operate in a purely discrete paradigm.

But, with the second part of the tension in Section 2.2 (the tension between explicit vs implicit factorizations of speech), we are closer to resolving it than the first. With kNN-VC and TransFusion, we showcased novel methods to perform speech processing tasks that lead to interesting capabilities. This is particularly the case for kNN-VC, where designing a model around using the implicitly disentangled SSL features allowed it to set new performance bounds in voice conversion and several other hard downstream tasks. Overall, we can see from nearly all of our experiments that designing systems that learn which aspects comprise speech leads to better generalization compared to those that use hand-crafted domain knowledge of how speech should be factored.

In summary, the explicit vs implicit tension is not fully solved since the act of moving away from explicit domain-expert knowledge makes the learning task harder. It poses the question about how to best control a desired aspect of speech if we have not specially designed a part in the model to explicitly control it. E.g. for text-to-voice conversion in Section 6.5.4, is our way of phrasing the task as finding a linear combination over known speakers optimal? Similarly for ASGAN in Chapter 3, is our latent interpolation method for denoising the best one? In other words, even if we have a perfectly linearly disentangled latent space of speech, it is not immediately clear what the optimal way to control generation – i.e. how do we best create an interface between some design (e.g. make the speech faster) and linear operations in the latent space? These are some of the areas warranting more investigation in future work.

7.3. Conclusion

This thesis investigated the application of disentanglement to speech processing to address the tension between the continuous nature of speech, and the discrete and explicit factorization approaches currently taken in many speech processing tasks. In our investigations, we made contributions to unconditional speech synthesis, speech recognition, and voice conversion. We proposed a new model for unconditional speech synthesis that set a new state-of-the-art performance level for the task. By focusing its design around learning a disentangled latent space, it can generalize to several tasks unseen during training, without any additional fine-tuning. In speech recognition, we proposed a novel framework for approaching speech recognition as a conditional denoising diffusion task. By exploiting the disentangled nature of features from speech representation models, we were able to design a model in this new framework which is competitive with top-performing contrastive models. And, for voice conversion, we proposed a new model based on the simple k-nearest neighbor algorithm applied to the disentangled features learnt by speech representation models. We showed that our model outperformed prior state-of-the-art models, with our model even inspiring several follow-up works by other authors.

The theme running through all these investigations is the attempt to use or learn disentangled latent spaces, with the ultimate aim of improving speech processing performance and generalization. Our unconditional speech synthesis model adapted ideas from image synthesis to build a latent space corresponding to speech whereby translations in specific linear directions corresponded to changing particular high-level characteristics of speech. However, it only works in a restricted setting. For voice conversion, our efforts perform the conversion by either a linear nearest-neighbor comparison in a latent space, or by learning a latent space for speaker identity which ensures disentanglement through a hierarchical approximation with linear subspaces. Using these voice conversion methods, we demonstrated their ability to yield improvements on several practical tasks, including low-resource ASR data augmentation, stuttered speech correction, and textually described voice conversion. Last, our proposed ASR method achieved its performance by combining the disentangled features from representation models with denoising diffusion, a typically continuous technique.

Overall, this work has shown that designing speech processing models that (i) *implicitly* learn to disentangle speech or (ii) *utilize* implicitly disentangled latent spaces *do* see benefits compared to models which do not, particularly in terms of generalization ability. However, the explicit-implicit tension in speech processing is not fully reconciled. Future work is needed to answer some of the key questions raised by our results, such as how to combine the scalability of speech representation models with the ability of more restricted models to sample from and learn probability distributions of speech.

Bibliography

[1] P. Taylor, *Text-to-speech synthesis.* Cambridge university press, 2009.

[2] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang *et al.*, “Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions,” in *ICASSP*, 2018.

[3] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, “Grad-TTS: A diffusion probabilistic model for text-to-speech,” in *ICML*, 2021.

[4] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao *et al.*, “FastSpeech 2: Fast and high-quality end-to-end text to speech,” in *ICLR*, 2021.

[5] K. Shen, Z. Ju, X. Tan, E. Liu, Y. Leng, L. He *et al.*, “NaturalSpeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers,” in *ICLR*, 2024.

[6] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas *et al.*, “Mistral 7B,” *arXiv preprint arXiv:2310.06825*, 2023.

[7] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton *et al.*, “Photorealistic text-to-image diffusion models with deep language understanding,” *arXiv preprint arXiv:2205.11487*, 2022.

[8] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with CLIP latents,” *arXiv preprint arXiv:2204.06125*, 2022.

[9] D. Zhang, S. Li, X. Zhang, J. Zhan, P. Wang, Y. Zhou *et al.*, “SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities,” *arXiv preprint arXiv:2305.11000*, 2023.

[10] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu *et al.*, “Neural codec language models are zero-shot text to speech synthesizers,” *arXiv preprint arXiv:2301.02111*, 2023.

[11] K. Qian, Y. Zhang, S. Chang, M. Hasegawa-Johnson, and D. Cox, “Unsupervised speech decomposition via triple information bottleneck,” in *ICML.* PMLR, 2020.

[12] B. van Niekerk, M.-A. Carbonneau, and H. Kamper, “Rhythm modeling for voice conversion,” *IEEE Signal Processing Letters*, vol. 30, pp. 1297–1301, 2023.

[13] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “HuBERT: Self-supervised speech representation learning by masked prediction of hidden units,” *arXiv preprint arXiv:2106.07447*, 2021.

[14] R. Sutton, “The bitter lesson,” *Incomplete Ideas (blog)*, vol. 13, p. 12, 2019.

[15] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *CVPR*, 2019.

[16] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of StyleGAN,” in *CVPR*, 2020.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair *et al.*, “Generative adversarial nets,” in *NeurIPS*, 2014.

[18] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen *et al.*, “Alias-free generative adversarial networks,” in *NeurIPS*, 2021.

[19] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” in *ICLR*, 2018.

[20] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, “Diffwave: A versatile diffusion model for audio synthesis,” *arXiv preprint arXiv:2009.09761*, 2020.

[21] K. Goel, A. Gu, C. Donahue, and C. Ré, “It’s raw! Audio generation with state-space models,” *arXiv preprint arXiv:2202.09729*, 2022.

[22] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *NeurIPS*, 2020.

[23] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen *et al.*, “WavLM: Large-scale self-supervised pre-training for full stack speech processing,” *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[24] S. wen Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin *et al.*, “SUPERB: Speech processing universal PERformance benchmark,” in *Interspeech*, 2021.

[25] S. H. Mohammadi and A. Kain, “An overview of voice conversion systems,” *Speech Commun.*, vol. 88, pp. 65–82, 2017.

[26] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” in *ICML*, 2006.

[27] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *ICML*, 2015.

[28] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool, “RePaint: Inpainting using denoising diffusion probabilistic models,” *CoRR*, 2022.

[29] X. An, Y. Wang, S. Yang, Z. Ma, and L. Xie, “Learning hierarchical representations for expressive speaking style in end-to-end speech synthesis,” in *ASRU*, 2019.

[30] J. Li, W. Tu, and L. Xiao, “FreeVC: Towards high-quality text-free one-shot voice conversion,” *arXiv preprint arXiv:2210.15418*, 2022.

[31] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "YourTTS: Towards zero-shot multi-speaker TTS and zero-shot voice conversion for everyone," in *PMLR*, 2022.

[32] Y. Zhao, W.-C. Huang, X. Tian, J. Yamagishi, R. K. Das, T. Kinnunen *et al.*, "Voice conversion challenge 2020: Intra-lingual semi-parallel and cross-lingual voice conversion," in *Proceedings of the Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, 2020.

[33] E. Gölge and K. Davis, "Introducing prompt-to-voice - describe it to hear it," 2022, accessed: 2023-08-05. [Online]. Available: https://archive.is/MotSP

[34] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *ICASSP*, 2018.

[35] M. Karjalainen, "Review of speech synthesis technology," *Helsinki University of Technology Laboratory of Acoustics and Audio Signal Processing*, 1999, Master's Thesis.

[36] N. Burton-Roberts and P. Carr, "On speech and natural language," *Language Sciences*, vol. 21, no. 4, pp. 371–406, 1999.

[37] O. D. Liu, H. Tang, and S. Goldwater, "Self-supervised predictive coding models encode speaker and phonetic information in orthogonal subspaces," in *Interspeech*, 2023.

[38] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "AutoVC: Zero-shot voice style transfer with only autoencoder loss," in *PMLR*, 2019.

[39] K. Lakhotia, E. Kharitonov, W.-N. Hsu, Y. Adi, A. Polyak, B. Bolte *et al.*, "Generative spoken language modeling from raw audio," *arXiv preprint arXiv:2102.01192*, 2021.

[40] A. Polyak, Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W.-N. Hsu *et al.*, "Speech resynthesis from discrete disentangled self-supervised representations," in *Interspeech*, 2021.

[41] E. Kharitonov, A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia *et al.*, "Text-free prosody-aware generative spoken language modeling," in *ACL*, 2022.

[42] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.

[43] Y. Stylianou, O. Cappe, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, pp. 131–142, 1998.

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

[45] G. Zhang, T. Merritt, S. Ribeiro, B. Tura-Vecino, K. Yanagisawa, K. Pokora *et al.*, “Comparing normalizing flows and diffusion models for prosody and acoustic modelling in text-to-speech,” in *Interspeech*, 2023.

[46] A. Polyak, Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W.-N. Hsu *et al.*, “Speech resynthesis from discrete disentangled self-supervised representations,” in *Interspeech*, 2021.

[47] A. Varga and R. K. Moore, “Hidden Markov model decomposition of speech and noise,” in *ICASSP*. IEEE, 1990.

[48] M. Schroeder and B. Atal, “Code-excited linear prediction (CELP): High-quality speech at very low bit rates,” in *ICASSP*. IEEE, 1985.

[49] B. Yegnanarayana, C. d’Alessandro, and V. Darsinos, “An iterative algorithm for decomposition of speech signals into periodic and aperiodic components,” *IEEE Transactions on Speech and Audio processing*, vol. 6, no. 1, pp. 1–11, 1998.

[50] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, and A. o. Graves, “WaveNet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.

[51] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium,” in *NeurIPS*, 2017.

[52] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *NeurIPS*, 2016.

[53] J. Kong, J. Kim, and J. Bae, “HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis,” in *NeurIPS*, 2020.

[54] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training generative adversarial networks with limited data,” in *NeurIPS*, 2020.

[55] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for GANs do actually converge?” in *ICML*, 2018.

[56] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly *et al.*, “Tacotron: Towards end-to-end speech synthesis,” in *Interspeech*, 2017.

[57] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, 2019.

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[59] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *ICML*, 2023.

[60] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal *et al.*, “Language models are few-shot learners,” in *NeurIPS*, 2020.

[61] D. Suundermann, G. Strecha, A. Bonafonte, H. Höge, and H. Ney, “Evaluation of VTLN-based voice conversion for embedded speech synthesis.” in *Interspeech*, 2005.

[62] Z.-W. Shuang, R. Bakis, S. Shechtman, D. Chazan, and Y. Qin, “Frequency warping based on mapping formant parameters,” in *Interspeech*, 2006.

[63] D. Erro and A. Moreno, “Weighted frequency warping for voice conversion,” in *Interspeech*, 2007.

[64] M. Baas and H. Kamper, “StarGAN-ZSVC: Towards zero-shot voice conversion in low-resource contexts,” in *SACAIR*, 2020.

[65] X. An, Y. Wang, S. Yang, Z. Ma, and L. Xie, “Learning hierarchical representations for expressive speaking style in end-to-end speech synthesis,” in *ASRU*, 2019.

[66] Z. Yang, W. Zhang, Y. Liu, and X. Xing, “Cross-lingual voice conversion with disentangled universal linguistic representations,” in *Interspeech*, 2021.

[67] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, “StarGAN-VC: Non-parallel many-to-many voice conversion using star generative adversarial networks,” in *IEEE SLT Workshop*, 2018.

[68] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. S. Kudinov, and J. Wei, “Diffusion-based voice conversion with fast maximum likelihood sampling scheme,” in *ICLR*, 2022.

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

[70] B. van Niekerk, L. Nortje, M. Baas, and H. Kamper, “Analyzing speaker information in self-supervised models to improve zero-resource speech processing,” in *Interspeech*, 2021.

[71] T. A. Nguyen, M. de Seyssel, P. Rozé, M. Rivière, E. Kharitonov, A. Baevski *et al.*, “The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling,” *arXiv preprint arXiv:2011.11588*, 2020.

[72] W.-N. Hsu, H. Tang, and J. Glass, “Unsupervised adaptation with interpretable disentangled representations for distant conversational speech recognition,” in *Interspeech*, 2018.

[73] L. McInnes, J. Healy, N. Saul, and L. Großberger, “UMAP: Uniform manifold approximation and projection,” *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[74] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “LibriSpeech: an ASR corpus based on public domain audio books,” in *ICASSP*, 2015.

[75] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford *et al.*, “Zero-shot text-to-image generation,” in *ICML*, 2021.

[76] G. Beguš, “Generative adversarial phonology: Modeling unsupervised phonetic and phonological learning with neural networks,” *Frontiers in artificial intelligence*, vol. 3, 2020.

[77] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2021.

[78] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *ICLR*, 2021.

[79] Y. Xu, Z. Liu, M. Tegmark, and T. Jaakkola, "Poisson flow generative models," in *NeurIPS*, 2022.

[80] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, "Argmax flows and multinomial diffusion: Learning categorical distributions," in *NeurIPS*, 2021.

[81] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *ICLR*, 2022.

[82] M. Kwon, J. Jeong, and Y. Uh, "Diffusion models already have a semantic latent space," in *ICLR*, 2023.

[83] M. Baas and H. Kamper, "GAN you hear me? Reclaiming unconditional speech synthesis from diffusion models," in *IEEE SLT*, 2022.

[84] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *ICLR*, 2018.

[85] G. Beguš, "Local and non-local dependency learning and emergence of rule-like representations in speech data by deep convolutional generative adversarial networks," *Computer Speech & Language*, vol. 71, 2022.

[86] G. Beguš, T. Lu, and Z. Wang, "Basic syntax from speech: Spontaneous concatenation in unsupervised deep neural networks," *arXiv preprint arXiv:2305.01626*, 2023.

[87] J. Chen and M. Elsner, "Exploring how generative adversarial networks learn phonological representations," *arXiv preprint arXiv:2305.12501*, 2023.

[88] M. Jiralerspong and G. Gidel, "Generating diverse vocal bursts with StyleGAN2 and mel-spectrograms," in *ICML ExVo Generate*, 2022.

[89] G. Zhu, Y. Wen, M.-A. Carbonneau, and Z. Duan, "EDMSound: Spectrogram based diffusion models for efficient and high-quality audio synthesis," *arXiv preprint arXiv:2311.08667*, 2023.

[90] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal *et al.*, "Fourier features let networks learn high frequency functions in low dimensional domains," in *NeurIPS*, 2020.

[91] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[92] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross *et al.*, “fairseq: A fast, extensible toolkit for sequence modeling,” in *NAACL-HLT*, 2019.

[93] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.

[94] S. Tao and J. Wang, “Alleviation of gradient exploding in GANs: Fake can be real,” in *CVPR*, 2020.

[95] J. F. Kaiser, “Digital filters,” in *System analysis by digital computer.* Wiley New York, NY, 1966, pp. 218–285.

[96] P. Warden, “Speech Commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209*, 2018.

[97] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *CVPR*, 2017.

[98] S. Gurumurthy, R. Kiran Sarvadevabhatla, and R. Venkatesh Babu, “DeLiGAN: Generative adversarial networks for diverse and limited data,” in *CVPR*, 2017.

[99] Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang *et al.*, “Activation maximization generative adversarial nets,” in *ICLR*, 2018.

[100] W.-C. Huang, E. Cooper, Y. Tsao, H.-M. Wang, T. Toda, and J. o. Yamagishi, “The VoiceMOS Challenge 2022,” *arXiv preprint arXiv:2203.11389*, 2022.

[101] J. Benesty, S. Makino, and J. Chen, *Speech enhancement.* Springer Science & Business Media, 2006.

[102] B. van Niekerk, M.-A. Carbonneau, J. Zaïdi, M. Baas, H. Seuté, and H. Kamper, “A comparison of discrete and soft speech units for improved voice conversion,” in *ICASSP*. IEEE, 2022.

[103] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, “Perceptual evaluation of speech quality (PESQ) - a new method for speech quality assessment of telephone networks and codecs,” in *ICASSP*, 2001.

[104] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, “A short-time objective intelligibility measure for time-frequency weighted noisy speech,” in *ICASSP*, 2010.

[105] H. J. Park, B. H. Kang, W. Shin, J. S. Kim, and S. W. Han, “MANNER: Multi-view attention network for noise erasure,” in *ICASSP*, 2022.

[106] M. M. Kashyap, A. Tambwekar, K. Manohara, and S. Natarajan, “Speech denoising without clean training data: A Noise2Noise approach,” in *Interspeech*, 2021.

[107] D. Roich, R. Mokady, A. H. Bermano, and D. Cohen-Or, “Pivotal tuning for latent-based editing of real images,” in *CVPR*, 2021.

[108] D. Wang, L. Deng, Y. T. Yeung, X. Chen, X. Liu, and H. Meng, "VQMIVC: Vector quantization and mutual information-based unsupervised speech representation disentanglement for one-shot voice conversion," in *Interspeech*, 2021.

[109] J.-c. Chou and H.-Y. Lee, "One-shot voice conversion by separating speaker and content representations with instance normalization," in *Interspeech*, 2019.

[110] S. Liu, Y. Cao, D. Wang, X. Wu, X. Liu, and H. Meng, "Any-to-many voice conversion with location-relative sequence-to-sequence modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1717–1728, 2021.

[111] K. Fujii, J. Okawa, and K. Suigetsu, "High-individuality voice conversion based on concatenative speech synthesis," *International Journal of Electrical and Computer Engineering*, vol. 1, no. 11, pp. 1625 – 1630, 2007.

[112] Z. Jin, A. Finkelstein, S. DiVerdi, J. Lu, and G. J. Mysore, "Cute: A concatenative method for voice conversion using exemplar-based unit selection," in *ICASSP*, 2016.

[113] D. Sundermann, H. Hoge, A. Bonafonte, H. Ney, A. Black, and S. Narayanan, "Text-independent voice conversion based on unit selection," in *ICASSP*, 2006.

[114] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *IEEE ASRU*, 2021.

[115] G.-T. Lin, C.-L. Feng, W.-P. Huang, Y. Tseng, T.-H. Lin, C.-A. Li *et al.*, "On the utility of self-supervised models for prosody-related tasks," in *IEEE SLT*, 2023.

[116] E. Dunbar, N. Hamilakis, and E. Dupoux, "Self-supervised language learning from raw audio: Lessons from the zero resource speech challenge," *Journal of Selected Topics in Signal Processing*, 2022.

[117] B. Sisman, J. Yamagishi, S. King, and H. Li, "An overview of voice conversion and its challenges: From statistical modeling to deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 132–157, 2021.

[118] E. Fix, *Discriminatory analysis. Nonparametric discrimination: Consistency properties.* USAF school of Aviation Medicine, 1985, vol. 1.

[119] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *ICASSP*. IEEE, 2018.

[120] B. Sha, X. Li, Z. Wu, Y. Shan, and H. Meng, "Neural concatenative singing voice conversion: rethinking concatenation-based approach for one-shot singing voice conversion," *arXiv preprint arXiv:2312.04919*, 2023.

[121] S. Shan, Y. Li, A. Banerjee, and J. B. Oliva, "Phoneme hallucinator: One-shot voice conversion via set expansion," *arXiv preprint arXiv:2308.06382*, 2023.

[122] Y.-A. Chung, Y. Zhang, W. Han, C.-C. Chiu, J. Qin, R. Pang *et al.*, "W2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," in *ASRU*, 2021.

[123] J. R. Movellan and P. Mineiro, "A diffusion network approach to visual speech recognition," in *AVSP*, 1999.

[124] A. Hannun, "Sequence modeling with CTC," *Distill*, 2017, https://distill.pub/2017/ctc.

[125] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

[126] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *PMLR*, 2021.

[127] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[128] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[129] Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang *et al.*, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504*, 2020.

[130] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li *et al.*, "Improved noisy student training for automatic speech recognition," in *Interspeech*, 2020.

[131] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré *et al.*, "Libri-Light: A benchmark for ASR with limited or no supervision," in *ICASSP*, 2020.

[132] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: A survey," *Speech Commun.*, vol. 56, pp. 85–100, 2014.

[133] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.

[134] M. Hunt and C. Lefebvre, "A comparison of several acoustic representations for speech recognition with degraded and undegraded speech," in *ICASSP*, 1989.

[135] G. Lathoud, M. Magimai-Doss, B. Mesot, and H. Bourlard, "Unsupervised spectral subtraction for noise-robust ASR," in *ASRU*, 2005.

[136] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, and E. o. Cubuk, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, 2019.

[137] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (VTLP) improves speech recognition," in *ICML*, 2013.

[138] J. Bellegarda, P. de Souza, A. Nadas, D. Nahamoo, M. Picheny, and L. Bahl, "The metamorphic algorithm: a speaker mapping approach to data augmentation," *IEEE/ACM TASLP*, vol. 2, no. 3, pp. 413–420, 1994.

[139] A. Laptev, R. Korostik, A. Svischev, A. Andrusenko, I. Medennikov, and S. Rybin, "You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation," in *CISP-BMEI*, 2020.

[140] M. A. Hedderich, L. Lange, H. Adel, J. Strötgen, and D. Klakow, "A survey on recent approaches for natural language processing in low-resource scenarios," in *NAACL*, 2021.

[141] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM TASLP*, vol. 23, no. 9, pp. 1469–1477, 2015.

[142] G. Huybrechts, T. Merritt, G. Comini, B. Perz, R. Shah, and J. Lorenzo-Trueba, "Low-resource expressive text-to-speech using data augmentation," in *ICASSP*, 2021.

[143] S. Shahnawazuddin, N. Adiga, K. Kumar, A. Poddar, and W. Ahmad, "Voice conversion based data augmentation to improve children's speech recognition in limited data scenario," in *Interspeech*, 2020.

[144] S. Zhao, T. H. Nguyen, H. Wang, and B. Ma, "Towards natural bilingual and code-switched speech synthesis based on mix of monolingual recordings and cross-lingual voice conversion," in *Interspeech*, 2020.

[145] G. Keskin, T. Lee, C. Stephenson, and O. H. Elibol, "Measuring the effectiveness of voice conversion on speaker identification and automatic speech recognition systems," *arXiv preprint arXiv:1905.12531*, 2019.

[146] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor *et al.*, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *ICML*, 2018.

[147] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli, "Efficient self-supervised learning with contextualized target representations for vision, speech and language," in *ICML*, 2023.

[148] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–80, 1997.

[149] D. van Niekerk, C. van Heerden, M. Davel, N. Kleynhans, O. Kjartansson, M. Jansche *et al.*, "Rapid development of TTS corpora for four South African languages," in *Interspeech*, 2017.

[150] T. I. Modipa, M. H. Davel, and F. de Wet, "Implications of Sepedi/English code switching for ASR systems," in *PRASA*, 2013.

[151] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *SPIE AIML MDO*, 2019.

[152] K. Suzuki, S. Sakamoto, T. Taniguchi, and H. Kameoka, "Speak like a dog: Human to non-human creature voice conversion," in *APSIPA ASC*, 2022.

[153] M. Chu, M. Yang, C. Xu, Y. Ma, J. Wang, Z. Fan *et al.*, "E-DGAN: An encoder-decoder generative adversarial network based method for pathological to normal voice conversion," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 5, pp. 2489–2500, 2023.

[154] M. Baas, B. van Niekerk, and H. Kamper, "Voice conversion with just nearest neighbors," in *Interspeech*, 2023.

[155] K. Arjun, S. Karthik, D. Kamalnath, P. Chanda, and S. Tripathi, "Automatic correction of stutter in disfluent speech," *Procedia Computer Science*, vol. 171, pp. 1363–1370, 2020.

[156] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one transformer vae," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1953–1967, 2023.

[157] A. Barahona-Ríos and T. Collins, "NoiseBandNet: Controllable time-varying neural synthesis of sound effects using filterbanks," *arXiv preprint arXiv:2307.08007*, 2023.

[158] C. Lea, V. Mitra, A. Joshi, S. Kajarekar, and J. P. Bigham, "SEP-28k: A dataset for stuttering event detection from podcasts with people who stutter," in *ICASSP*, 2021.

[159] H. Liu, X. Liu, Q. Kong, Q. Tian, Y. Zhao, D. Wang *et al.*, "VoiceFixer: A unified framework for high-fidelity speech restoration," in *Interspeech*, 2022.

[160] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "MLS: A large-scale multilingual dataset for speech research," *ArXiv*, vol. abs/2012.03411, 2020.

[161] S. Mohanty, "Musical instrument's sound dataset," 2022, accessed: 2023-08-06. [Online]. Available: https://archive.is/BtIGp

[162] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A reference-free metric for evaluating music enhancement algorithms," in *Interspeech*, 2019.

[163] E. Cheng, "Morphing between voices using kNN-VC," 2023, accessed: 2023-08-07. [Online]. Available: https://eccheng.github.io/ml/audio/vc/2023/07/04/knn-vc-morph.html

[164] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-Networks," in *EMNLP*, 2019.

[165] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[166] Y. Lv, J. Yao, P. Chen, H. Zhou, H. Lu, and L. Xie, "SALT: Distinguishable speaker anonymization through latent space transformation," *arXiv preprint arXiv:2310.05051*, 2023.

[167] T. Shibuya, Y. Takida, and Y. Mitsufuji, "BigVSAN: Enhancing GAN-based neural vocoders with slicing adversarial network," *arXiv preprint arXiv:2309.02836*, 2023.