ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

**MASTER THESIS**

in

Natural Language Processing

# Fine-Tuning Neural Codec Language Models from Feedback with Reinforcement Learning

CANDIDATE

Lorenzo Pratesi

SUPERVISOR

Prof. Paolo Torroni

Academic Year 2022/2023

Session 3rd

# ABSTRACT

Neural codec language models (NCLMs) are speech synthesizers that address the text-to-speech (TTS) task as a language modeling task rather than continuous signal regression as in previous work. They showed an impressive generalization capability, surpassing previous state-of-the-art zero-shot TTS models by means of speaker similarity and naturalness. Although addressing speech synthesis as a language modeling task in part allows to train on large and diverse speech data crawled from the Internet, it also brings some issues common to those of large language models (LLMs) for text generation. While LLMs may generate outputs with made up facts or biased and toxic contents, neural codec language models suffers from synthesis robustness and expressiveness. Reinforcement Learning from Human Feedback (RLHF) has emerged to tackle the issues of LLMs, by using human feedback to align the generated responses to the user preferences. Using RLHF helped LLMs to reduce the amount of generated toxic content and false facts. Motivated by the success of RLHF in the text generation domain, this work proposes to fine-tune NCLMs from feedback with reinforcement learning following the RLHF training pipeline. We conduct a series of experiments with VALL-E pretrained on LibriTTS, fine-tuning it to optimize different kind of feedback: intelligibility, naturalness, speaker similarity and waveform duration. Our results show that fine-tuning helped to increase the intelligibility of the model, showing a WER reduction up to 20.954%, but also to change the speech duration according to the reward signal. Finally, we delineate limitations of our experimental setup and propose practical mitigations, to be explored in future work.

# CONTENTS

# CHAPTER 1

# INTRODUCTION

Recently, generative AI demonstrated impressive capabilities in various fields, attracting unprecedented attention in both industrial and academic areas [72]. Generative AI refers to technologies that can generate novel content rather than taking actions and decision on existing one. Many generative AI models, such as ChatGPT [73] and DALL-E [46] demonstrated the extraordinary capability to create new and diverse content [72]. These powerful models have shown unprecedented capabilities in synthesizing realistic images, audio, text, and other data modalities [14].

Text-to-speech (TTS), or speech synthesis, is an example of generative AI task, which aims to generate natural and intelligible speech from text [56]. With the advent of deep learning, neural network-based speech synthesis has thrived, and a large amount of research work comes out focusing on different aspects of neural TTS [55]. Currently, cascaded text-to-speech systems [51] [47] [30] often leverage a three-stage pipeline with a text analysis module, an acoustic model and a vocoder using mel spectrograms or other acoustic features as the intermediate representations. More advanced TTS models can synthesize high-quality speech from single or multiple speakers [32] [68], but they still require high-quality clean data, making training on large-scale data crawled from the Internet unfeasible [55]. Because the training data is relatively small, current TTS systems still suffer from poor generalization. To tackle those issues, neural codec language models (NCLMs) have emerged, addressing speech synthesis as a conditional language modeling task rather than continuous signal regression. NCLMs are language models trained

to predict discrete acoustic representations learned through neural audio codecs (NACs) [70] [12] [48]. Those models demonstrated impressive zero-shot TTS results, while at the same providing high speech diversity, acoustic environment preservation and speaker emotion maintenance [62] [20]. However, since these models are trained to solve the language modeling task and are trained also on unclean data, they suffer from synthesis robustness (words may be unclear, missed, or duplicated in speech synthesis) and expressiveness (the ability of representing different speaking styles and voice types) [62] [20]. Large language models (LLMs) are another example of successful type of generative AI models. LLMs are transformer language models that contain a number of parameters in the order of bilions, which are trained on huge amount of text data, such as GPT-3 and LLaMA [10] [57]. They showed impressive capabilities in text generation, commonsense reasoning, spatial reasoning, mathematical reasoning or programming assistance [72]. However, many LLMs still suffer from several issues, such as making up facts, generating toxic or biased content and not following the user input [4] [19] [65] [40]. The reason of this behavior is in part because of the large language modeling objective (predict the next word on a large dataset of Internet text), is different from the "following the user's requests helpfully and safely" objective [40]. Reinforcement learning from human feedback (RLHF) has emerged as a practical way to introduce learning goals that are more closely aligned with human values and intents, promoting ethically sound and socially responsible AI systems [18]. RLHF is a variant of reinforcement learning (RL) that learns from human feedback instead of relying on hand-crafted reward functions. RLHF has seen a number of successful applications, ranging from image-generation, robotics, control and games to the domain of LLMs [18]. In the LLMs domain, RLHF not only demonstrated to reduce toxicity and biased content, but also to reduce the number of made up answers and to increase the alignment of the responses with the preferences of the users [39] [40].

Motivated by the success of RLHF in the text synthesis domain, this work proposes to fine-tune NCLMs from feedback with reinforcement learning following the RLHF training pipeline, trying to mitigate the issues of NCLMs and exploring ways of optimizing other non-differentiable objectives. This thesis proposes to make a first step to demonstrate that it is possible to optimize feedback received from reward signals or preferences for a neural codec language model, to enhance different TTS metrics such as intelligibility and speaker similarity. We believe that such method could be of extreme interest for

both business applications and research purposes, since it could show a way to mitigate the issues related to training on large, diverse and unclean TTS data.

To demonstrate that NCLMs can be successfully fine-tuned from feedback with RL, we first pretrain a NCLM on an available open source TTS dataset to obtain both a baseline and the model we optimize with RL. We then define some reward functions to provide a preference feedback to the network, with the goal of optimizing some TTS objectives. To fine-tune the pretrained model, we first build a dataset containing the prompts to be used by the NCLM for the generation of synthetic speech. For a single prompt, the model generates multiple responses and the reward model gives a score to them, providing an intrinsic preference ranking. Finally, we use RL to optimize the feedback for the pretrained NCLM on the prompt dataset. We evaluate the NCLM by means of objective TTS metrics on unseen test data from an open source TTS dataset. Specifically, we measure the naturalness, intelligibility, speaker similarity maintenance and average duration of the speech generated by the NCLM.

As a further contribution, we open-source our code through a publicly available GitHub repository[1] and we provide a collection of demo samples for most of the experiments[2].

[1] https://github.com/prahtz/nclm-feedback
[2] https://prahtz.github.io/nclm-feedback

# CHAPTER 2

# BACKGROUND

## 2.1 Language Models

Language Models (LMs) are probabilistic models of natural language. They are usually used to capture the regularities of natural language and use them to solve various tasks, such as machine translation, natural language generation and information retrieval. Language models can be trained to predict the next word, character or, in general, element in a sequence, also known as language modeling task.

Statistical language models, exemplified by n-gram models [16], were a pioneering method in addressing the language modeling task. These models estimated the probability distribution of word sequences based on observed frequencies in a training corpus. Simple and interpretable, they were widely used, but their limitations included struggles with capturing long-range dependencies (relationships between elements separated by large distances in a sequence) and curse of dimensionality. In contrast, Neural Language Models, leveraging the expressive power of neural networks, have emerged as a transformative force in language modeling. Embodied by architectures like recurrent neural networks [31] and transformers [60], these models naturally learn intricate language patterns, allowing them to capture contextual dependencies across extended sequences. Their superior performance in generalization and adaptability across diverse Natural Language Processing (NLP) tasks has reshaped the landscape of language modeling [5] [13] [43] [60] [28].

## 2.2 Speech Signals

Speech signals are audio signals defined as pressure variations traveling through the air. Speech emerges from a speakers mouth, nose and cheeks, is a one-dimensional function (air pressure) of time [49]. Microphones can capture those air fluctuations and convert them into continuous electrical signals. An analogue-to-digital process can transform those electric signals into a digital representation of the speech. The process consists of two steps: sampling and quantization. The former is defined by the sampling rate, that indicates the number of data points to capture from the analogue signal over a specific time-window, while the latter limits the range of amplitudes to a discrete set. The quantization step must be defined to represent the sampled data points into a digital system, by choosing a fixed number of bits to represent them. Usually, amplitudes are represented by 16, 32, or 64 bit integers. In the following sections and chapters, we will use the term "waveform" to indicate a digital speech signal. A waveform is a graphical representation of the amplitudes of a signal plotted over time.

## 2.3 TTS - Text-To-Speech

Text-to-Speech (TTS) is the process of generating spoken language given written text or any kind of textual representation. The primary goal of TTS systems is to generate human-like speech from input text, allowing computers or devices to communicate with users through spoken words. This technology has numerous applications, including accessibility for visually impaired individuals, voice assistants, audiobooks, navigation systems, and more. Traditional text-to-speech systems often relied on rule-based or concatenative methods, where pre-recorded segments of human speech were concatenated to generate the desired output [53]. In contrast, modern TTS systems are neural. They leverage deep neural networks, such as recurrent neural networks (RNNs) [64] [59], convolutional neural networks (CNNs) [1] [54], or more recently, transformer-based architectures [68] [7]. These neural networks are trained on large datasets of text and corresponding audio to learn the complex relationships between text and speech. Neural TTS has several advantages over traditional methods, including improved naturalness, expressiveness, and the ability to handle various speaking styles. Additionally, these systems can generalize better to new and unseen data, making them more adaptable and capable of generating high-quality synthetic speech [68] [64] [62].

### 2.3.1 Cascaded TTS models

A cascaded TTS system addresses the speech synthesis task in sequential steps. A general speech generation pipeline can be described by three cascaded modules. First, there is the text analysis module, that converts the input representation into a set of linguistic features, such as phonemes and prosody information. Those are then processed by the acoustic model, that generates intermediate acoustic features such as spectrograms or acoustic embeddings. The final module, called vocoder, takes the intermediate acoustic features generated in the previous step to produce a waveform. Examples of such models are FastSpeech [47] and Transformer TTS [30]. Although such models allow to control many aspects of speech generation thanks to their three-stage processing pipeline, they usually suffer of poor generalization and require high-quality clean data to produce good results.

### 2.3.2 End-To-End TTS models

In end-to-end (E2E) TTS models, the entire process, from input text to synthesized speech, is handled by a single neural network or model. Such models are responsible for both understanding the linguistic content of the input text and generating the corresponding speech waveform. VITS [68] and YourTTS [11] are examples of E2E TTS models. Since E2E TTS models generate the waveform directly, without learning explicitly intermediate acoustic representations such as spectrograms, they have the potential to capture prosodic features (such as pitch, rhythm, and intonation) more naturally, as these elements are implicitly learned during training [68].

### 2.3.3 Evaluation

A TTS system is usually evaluated under two dimensions: intelligibility and naturalness. Intelligibility measures how well words of the TTS system's outputs are understood, while naturalness refers to how closely the synthesized speech resembles the natural and expressive qualities of human speech. Several factors contribute to the perception of naturalness in synthesized speech, such as prosody, emotional expression, accent and speaker similarity.

There are two main ways to measure a TTS system, by either using subjective tests or objective metrics. Subjective evaluations, such as Mean Opinion Score (MOS) [61] and

MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) [34], are usually audio only listening test, where participants are asked to rank the quality of the audio according to their preferences. They are used to evaluate aspects of speech that are hard to capture using algorithms, such as speaker similarity or emotion. Subjective metrics are usually used to evaluate the naturalness of a TTS system. However, such tests are expensive to perform, since they require a sufficiently high number of human participants to be reliable. Objective tests are instead ways to evaluate the TTS system in a quantifiable and measurable way.

WER (Word Error Rate), PER (Phonemes Error Rate) and SER (Sentence Error Rate) are examples of objective metrics. Those are used to measure the intelligibility of a TTS system, by using Automatic Speech Recognition (ASR) pretrained models, such as wav2vec2.0 [3] or HuBERT [15]. Given the audio produced by the TTS system under analysis, the ASR model computes its transcription, that is compared with the input text/phonemes used as input by the TTS model. The WER, PER or SER between the compared textual representations is used to asses the intelligibility of the TTS model.

Some aspects of the naturalness of a speech, such as speaker similarity, can be measured also with objective evaluations. For instance, to measure the speaker similarity between two audio samples, in our work we use TitaNet [23], a neural network trained for extracting speaker representations from a given speech. Given two speech samples, one generated by the TTS system and the ground truth, TitaNet can extract their speaker embedding, a vector representation of the speakers. Cosine similarity can be used to measure the distance between the two embeddings. The lower the distance, the higher the speaker similarity between the two samples.

There are also attempts to predict the MOS of a given speech without human intervention, by training neural networks to predict real MOS scores in supervised way on a MOS-labeled dataset. Speech Quality Assessment using Non-Matching References (NORESQA-MOS) [33] is a an example of such models, where the model is trained to predict the MOS of a given speech and a non-matching reference. Although the results of such models are not on par with those obtained by human participants, they can be used to measure the overall naturalness of a speech in an objective way.

## 2.4 Speech quantization

Speech Quantization refers to the use of quantization techniques for speech signals. If we represent waveforms with 16 bit integers, a generative speech model requires to output $2^{16} = 65,536$ probabilities at each time-step to synthesize raw speech (speech as a one-dimensional function of time), for a number of time-steps that is usually huge with high sample rates, making raw speech generation intractable [62]. Speech quantization techniques compress raw speech into discrete representations and reduce its sequence length. Learning discrete representation of raw data have proven to be beneficial for various tasks, such as image generation [46] [45], video generation [66] and speech synthesis [26] [62]. Although various speech quantization methods are proven to be efficient for speech generation and discrete audio representation learning [37] [2] [27], this section focuses its attention on Neural Audio Codecs (NACs), describing their core architecture and their working principles.

### 2.4.1 VQ-VAE - Vector Quantized-Variational AutoEncoder

VQ-VAE is a type of variational autoencoder, a deep neural network used to learn neural discrete representations of complex data, such as video and audio sources [38]. It is capable to encode a data source into a discrete latent space, usually with lower dimensions, and decode the result to reconstruct the original source. VQ-VAE is an extension of the variational autoencoder framework, a type of autoencoder that encodes and decodes the latent space probabilistically [22]. In addition to the variational autoencoder, VQ-VAE relies on vector quantization techniques to discretize (quantize) the continuous representation produced by the encoder into a discrete number of learnable vectors called codebook. The encoder maps input data to the nearest vector in the codebook, introducing also a form of compression. The decoder is trained to reconstruct the input data by using the learned vector instead of the output of the encoder.

### 2.4.2 Neural Audio Codecs

Audio Codecs are algorithms that compress (encode) or decompress (decode) a codec that encodes or decodes audio data. A codec is any compressed representation of an audio data. Audio codecs are crucial for compressing audio data, reducing file sizes for efficient storage and transmission. They enable compatibility across devices, support

real-time processing, and preserve audio quality, making them essential for applications like streaming, online communication, and multimedia production. Examples of such algorithms are Opus [58], MP3 and AAC [8].

Neural Audio Codecs (NACs) are audio codecs that leverage neural networks. They learn from data, adapting to different audio content, aiming for high compression efficiency while maintaining audio quality. Neural Audio Codecs such as SoundStream [70], EnCodec [12] and TiCodec [48] are examples of state-of-the-art Neural Audio Codecs. They are VQ-VAE-based models, trained on a vast amount of clean and noisy speech, music and general audio to allow the generation of high-quality audio content from quantized embeddings. Differently from pure VQ-VAEs, where the latent space is quantized as a single codebook, SoundStream, EnCodec and TiCodec build a cascade of codebooks through as many sequential residual vector quantizers [70] [12]. In a residual vector quantization pipeline, the unquantized vector is passed to a vector quantizer, and the closest entry in the codebook is extracted. Then, iteratively, the residual between the unquantized vector and the extracted vector is passed to the following vector quantizer.

### 2.4.3 TiCodec

TiCodec [48] is a NAC for speech with time-invariant codes based on a VQ-VAE architecture. Motivated by the success of Neural Codec Language Models (Section 2.5), TiCodec introduces a way to reduce the number of discrete codes needed to represent an input speech, by adding an additional module that captures time-invariant features of speech, such as timbre and acoustic environment, reducing the amount of time-level information that needs encoding and effectively decreasing the number speech codes [48]. Specifically, given an input waveform, an intermediate representation of TiCodec's encoder is used as time-invariant embedding. This embedding is then processed by a Group Vector Quantization module [67], that divides it into eight groups, resulting in eight different discrete tokens [48]. These codes are used by TiCodec's decoder, together with the time-level discrete codes, to reconstruct the input waveform. The authors of TiCodec showed that their model can enhance the quality of reconstruction speech with fewer tokens and also increase the similarity and naturalness on LibriTTS [71], a TTS dataset collected from audiobooks (see Section 4.2).

## 2.5 Neural Codec Language Models

Neural Codec Language Models are models that address the TTS task as language modeling approach with discrete codes obtained by an off-the-shelf neural audio codec. Specifically, they approach the TTS task as a conditional language modeling task rather than continuous signal regression, where a language model is conditioned on a discrete textual representation (the text to be synthesized) and, optionally, on discrete audio codes [62] [20]. Such models generate discrete acoustic tokens given the textual and acoustic prompts; those tokens can then be used by the neural audio codec's decoder to generate the waveform. This pipeline allows the model to generate personalized speech leveraging the power of language models, including the use large, diverse, and multi-speaker speech data. VALL-E [62] and CLaM-TTS [20] are examples of Neural Codec Language Models. They surpassed the current state-of-the-art models for speech generation in zero-shot scenarios, showing impressive generalization capabilities [62] [20].

### 2.5.1 VALL-E

VALL-E [62] is a conditional neural codec language model for text-to-speech synthesis. Given a textual prompt and an acoustic prompt, the model is originally trained to predict discrete acoustic tokens of audio samples, encoded by Encodec [12]. The model features the use of an autoregressive (AR) transformer [60] to predict the discrete codes of the first quantizer of Encodec, and the use of a non-autoregressive (NAR) transformer to predict sequentially the codes of the other seven quantizers. The predicted codes are then used by Encodec's decoder to produce a waveform.

Formally, given a waveform $y$ and its phoneme transcription $x$, they encode the audio sample with Encodec, obtaining the acoustic matrix $C^{T\times 8} = Encodec(y)$, where $T$ is the utterance length and $8$ is the number of codebooks. The AR transformer is trained as a language model to predict the codes of the first quantizer $c_{:,1}$, conditioned on the phoneme transcription $x$ and the encoding of the acoustic prompt $\hat{c}_{:,1}$, formulated as

$$p(c_{:,1}|\hat{c}_{:,1}, x; \theta_{AR}) = \prod_{t=0}^{T} p(c_{t,1}|c_{<t,1}, \hat{c}_{:,1}, x; \theta_{AR})$$

The NAR transformer is trained as a language model to predict the codes of the remaining seven quantizers $c_{:,2:8}$, conditioned again on $x$ and the while acoustic prompt matrix $\hat{C}$,

formulated as

$$p(C_{:,2:8}|\hat{C}, x; \theta_{NAR}) = \prod_{j=2}^{8} p(c_{:,j}|c_{:,<j}, \hat{C}, x; \theta_{NAR})$$

## 2.6 Reinforcement Learning

Reinforcement learning (RL) is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment [17]. An agent is an entity that takes actions in the environment based on rewards and punishments. The goal of an agent is to learn how to maximize the cumulative reward, the total reward accumulated over time by following the actions. RL is modeled as a Markov decision process. A Markov decision process (MDP) is a mathematical framework for modeling sequential decision making problems under uncertainty. Formally, an MDP is defined as a tuple $M = (S, A, P, R, \rho)$; at any timestep, the environment exists in a state $s \in S$, with $\rho$ being the initial state distribution. The agent takes an action $a \in A$ which transitions the environment to a new state $s^{'} \in S$. The stochastic transition function $P$ governs the dynamics of such transitions. For each transition, the agent receives a reward $R : S \times A \to \mathbb{R}$. Furthermore, the agent acts according to a policy $\pi$ in a space of policies $\Pi$, that produces a probability distribution over actions given a state [17]. The task of an RL algorithm is to interact with the MDP by simulating its transition dynamics $P$ and reward function $R$ and learn the optimal policy $\pi^*$, the policy that, if followed, maximizes the expected cumulative reward obtained by the agent.

Early RL research mainly focused on tabular and approximation-based algorithms to search for the optimal policy. Due to the lack of representation ability, traditional RL algorithms can only solve tasks with low-dimensional state and action spaces. However, tasks that are more complex and closer to the real-world situations usually have a higher dimensional state space and continuous action space, thus limiting the application of RL [63]. With the advent of deep learning, RL extended its range of applications, thanks to the powerful representation capabilities of neural networks. Since the successful application of deep Q-network [36] in game playing, more and more deep learning techniques and algorithms have been combined with RL, which have been used to solve tasks such as robot control, games, NLP and autonomous driving [63].

Deep learning has been successfully used to approximate policies for policy gradient methods. The basic idea behind policy gradient methods is to adjust the parameters of the

policy in the direction that increases the probability of taking actions that lead to higher expected rewards. The objective is to find a policy that maximizes the expected cumulative reward over time. Policy gradient methods are versatile and can be applied to a wide range of problems, including both discrete and continuous action spaces. Popular algorithms within the policy gradient framework include REINFORCE [52], A3C [35], and Proximal Policy Optimization (PPO) [50]. These methods have been successfully used in various domains, such as robotics, game playing, and natural language processing.

## 2.7 Reinforcement Learning from Human Feedback

Reinforcement Learning from Human Feedback (RLHF) is a machine learning algorithm that uses human feedback to optimize agents with Reinforcement Learning (RL) algorithms, such as PPO [50]. Standard RL is used to train models acting in an environment to maximize a reward function, so that they learn to produce more accurate outcomes. RLHF incorporates human feedback in the reward model so that the model to optimize can be aligned to their preferences. RLHF has been successfully used to fine-tune language models from human preferences for NLP tasks such as stylistic continuation and text summarization [74], but also to reduce toxic output generation and to improve truthfulness [40] [73].

A RLHF pipeline for language models optimization can be summarized as follows. First, a human labeled dataset with prompts and responses is collected. Then, the language model is fine-tuned or pretrained on the collected dataset with supervised learning. The learned model will be used to generate multiple responses given the same prompt, for a big enough number of prompts. Then, human participants are asked to indicate their preferences over the responses generated by a prompt, for each prompt in the dataset, and the results will be used to label each sample and create a new preference dataset. This dataset is then used to train a reward model with supervised learning, maximizing the likelihood of predicting the human preferences given the language model responses. Finally, the supervised fine-tuned (SFT) language model is trained with RL to optimize its policy, its probability distribution over words, to maximize the outcomes of the reward model given the prompts and the LM's responses. This process can be repeated over time, so that a new reward model can be trained on newly generated data from the current RL optimized language model.

RLHF implements a way to prevent the RL policy from moving too far from the initial pretrained model. Without any constraint, nothing prevents the policy to fool the reward function to give extremely high rewards to generated nonsense text. To avoid this from happening, a penalty with expectation $\beta KL(\pi, \rho)$ is added to the reward, where $KL$ is the Kullback-Leibler divergence, $\pi$ is the policy to be fine-tuned, $\rho$ is the pretrained language model and $\beta$ can be a constant or adjusted dynamically [74]. The KL divergence measures how one probability distribution $\pi$ is different from a second reference probability distribution $\rho$ [25]. The idea is to adjust the reward given to the generated responses so to penalize those that are generated by a policy that is too distant from the pretrained language model. Constraining the policy to be reasonably close to the probability distribution of the pretrained model automatically prevents it to generate gibberish texts.

# CHAPTER 3

# FINE-TUNING NEURAL CODEC LANGUAGE MODELS FROM FEEDBACK WITH REINFORCEMENT LEARNING

Ad described in Chapter 1, the goal of this thesis is to see if it is possible to optimize neural codec language models from feedback with reinforcement learning, motivated by the success of RLHF in the field of NLP with LLMs [73] [40]. In particular, this thesis proposes a training pipeline to optimize a NCLM to improve the produced speech under several dimensions, such as speaker similarity and intelligibility.

## 3.1 Problem formulation

Given a dataset $D = \{x_i, y_i\}$ where $y_i$ is an audio sample and $x_i$ is its phoneme transcription, we use a pre-trained neural audio codec to encode each audio sample into discrete audio codes, denoted as $Codec(y)$. We use just the first codebook of the NAC to encode the audio samples, then $Codec(y) = c^T$, where $c$ denotes the acoustic code vector and $T$ is the downsampled utterance length. Given an autoregressive NCLM $\rho$, we train it to predict $c^T$, conditioned on the phoneme transcription $x$ and a discrete acoustic prompt $\hat{c}$,

formulated as

$$p(c_{:}|\hat{c}, x; \theta_\rho) = \prod_{t=0}^{T} p(c_t|c_{<t}, \hat{c}, x; \theta_\rho)$$

The prompt $\hat{c}$ is a discrete acoustic representation obtained by encoding a speech sample $\hat{y}$ with the NAC, so that $\hat{c} = Codec(\hat{y})$. The recording sample $\hat{y}$ should represent the acoustic features of the speech we want to generate, such as speaker identity, emotion and pitch. After training, $\rho$ is able to generate discrete audio codes of the first codebook of the NAC given the phoneme transcription and the acoustic prompt. Codes are a discrete representation of the synthetic speech to be generated. To obtain a waveform, the generated codes can be decodes using the NAC's decoder. We denote the decoding process as $Decodec(c) = w$ where $c$ is the code vector and $w$ is the waveform.

Once $\rho$ is trained, we fine-tune it with the RLHF pipeline following [74] to maximize the outcomes of a reward model. We first need to build a prompt dataset $\hat{D}$, containing the prompts used by the NCLM to generate synthetic speech. A prompt is a tuple $q_i = (x_i, \hat{c}_i)$ where $x_i$ is the phoneme representation of the utterance to be generated and $\hat{c}_i$ is a discrete acoustic code representation of an audio prompt. The NCLM $\rho$ is asked to generate multiple responses $z_i = (z_{i1}, z_{i2}, ..., z_{in})$ given the prompt $q_i$. If we want to incorporate human feedback in the RL loop, the generated dataset $\hat{D} = \{q_i, z_i\}$ must be labeled by human participants according to their preferences, then fit a reward model $r$ to it, as described in [74]. Otherwise, is it possible to choose any kind of reward function $r$ to provide feedback to the NCLM. A reward function $r : X \times Y \rightarrow \mathbb{R}$ is used to provide a score to each response of $z_i$. Each score provides an intrinsic preference feedback to each response in $z_i$, allowing the NCLM to learn which is the most aligned with the provided feedback and which not.

As done in [74], we initialize the policy to fine-tune as $\pi = \rho$ and build a modified reward model $R(x, y) = r(x, y) - \beta KL(\pi, \rho) = r(x, y) - \beta log\frac{\pi(y,x)}{\rho(y,x)}$ to keep $\pi$ from moving too far from $\rho$. The value $\beta$ varies dynamically to reach a particular target value of $KL(\pi, \rho)$ [74]. Finally, we fine-tune $\pi$ with PPO [50] with reward $R$ on $q{\sim}\hat{D}$.

## 3.2 Feedback

In this section, we describe the feedback used to optimize the NCLM with RL. We do not incorporate human feedback for our experiments, since collecting crowd-sourced data would have been too expensive for the resources available for this thesis. We rely instead

on objective TTS metrics to provide feedback to the network in the RL loop.

**Intelligibility** We evaluate the intelligibility of the model with HuBERT-L [15] fine-tuned on 960h of LibriSpeech for ASR. Specifically, given the response codes generated by the policy, we use TiCodec's decoder to decode the waveform and transcribe it with HuBERT-L. We then compare the transcription with the textual prompt to compute the WER of the model and scale it in $[0, 1]$. Since RL maximizes a reward function, the feedback we provide is computed as $1 - WER$.

**Increase Waveform Duration** We provide a feedback to increase the average duration of waveforms generated by the model. We assign a reward of 1 to waveforms having a duration of around 6 times the duration of the audio prompt, that linearly decreases down to 0 for empty waveforms.

**Decrease Waveform Duration** This feedback is used to decrease the average duration of waveforms generated by the model. The reward model looks at the duration of the generated waveform and assigns a reward of 1 to those one second long responses, that linearly decreases down to 0 for waveforms around 6 times long the duration of the audio prompt. We also assign 0 to code vectors having a duration less than 1s.

**Speaker Similarity** We provide a feedback to the model to increase the speaker similarity of the generated audio sample with respect to the original waveform. We use TitaNet-L [23] from the NeMo toolkit [24] to extract speaker embeddings from waveforms. Specifically, given the waveform generated by TiCodec and VALL-E, and the waveform of the original audio sample, we compute their speaker embeddings with TitaNet-L, $S_g$ and $S_t$. We compute the cosine similarity between the two vectors $cos(\theta) = \frac{S_g \cdot S_t}{||S_g|| \, ||S_t||}$ and scale it in the range $[0, 1]$ to obtain the speaker similarity $SPK = \frac{\cos(\theta)+1}{2}$. Finally, the speaker similarity $SPK$ is provided as reward function in the RL loop to fine-tune VALL-E.

**Naturalness** We try to increase the naturalness and the overall speech quality of the generated synthetic speech by incorporating an objective TTS metric in the RL loop. We use NORESQA-MOS [33] for estimating the MOS of synthetic speech generated by VALL-E and TiCodec, and use it as a reward signal. NORESQA-MOS computes

a relative MOS between the speech we want to estimate and a non-matching reference speech, with the assumption of perfect MOS rating. We provide a MOS-based feedback to the model by computing the NORESQA-MOS on the generated synthetic waveform as test speech and the original audio sample as non-matching reference. Finally, we scale the reward in the range $[0, 1]$.

# CHAPTER 4

# EXPERIMENTS

## 4.1 Neural Audio Codec

We use TiCodec as pre-trained NAC for our experiments. For computational resources and time constraints, we adopt the 24kHz pre-trained checkpoint trained on LibriTTS with just one quantizer, allowing us to experiment with VALL-E without the NAR transformer. This version of TiCodec takes as input 24 kHz audios and its encoder produces 75 Hz embeddings, which is a 320-fold reduction in the sampling rate. Each embedding is modeled by a vector quantization module, with a codebook having 1024 entries. In this setting, a 1 second waveform is encoded with 75 codes plus 8 codes from the time-invariant quantizer.

## 4.2 Dataset

We use LibriTTS [71] as dataset for each experiment. LibriTTS is a cleaned an preprocessed version of LibriSpeech [41], a dataset derived from audiobooks. LibriTTS contains 586 hours of speech data from 2456 speakers in English language, with audio files at 24kHz sampling rate and their transcriptions. We phonemize all the transcriptions with phonemizer [6]. We use TiCodec [48] to extract the discrete acoustic codes from each audio sample. We use the first 25% of the original audio samples as audio prompts

for the RL optimization steps, up to a maximum of 3 seconds. With TiCodec and TitaNet-L, we extract and store from the audio prompts the time invariant codes and the speaker embeddings respectively.

## 4.3 Pre-training details

We decided to use VALL-E as NCLM for our experiments. Since there is not an official implementation of VALL-E, we rely on `lifeiteng/vall-e` [29], a PyTorch [42] unofficial implementation. We also followed the same training recipes described on the repository, unless otherwise specified. We use a downscaled version of the VALL-E autoregressive transformer as NCLM for each experiment. Specifically, a transformer with 6 layers, an embedding dimension of 256, 8 attention heads, a feedforward dimension of 4096 and dropout 0.1. We train this version of VALL-E to predict the codes produced by TiCodec described in the previous section. The model is trained using one NVIDIA RTX 2080 Ti 11GB GPU with batches containing a maximum of 7500 acoustic tokens for 100 epochs. Each batch is sampled so that it contains audio code vectors with similar lengths, to improve memory efficiency. We optimize with ScaledAdam [69] with an initial learning rate of 5e-2 and 200 warm-up steps, following [29].

## 4.4 Reinforcement Learning details

We fine-tune our pre-trained VALL-E with PPO [50] on LibriTTS train set. We filter-out the audio samples longer that 15 seconds before training. Given a prompt (phonemes and acoustic), VALL-E generates two responses and a reward model produces a reward for both. We experiment with two $KL(\pi, \rho)$ target values: 12 and 100; the first is a value that should keep the two policies close but has less flexibility to change, while the second has the opposite characteristics. We train for four PPO epochs with a batch size of 256 with one-sized mini-batches, for a total of 250k prompt-response pairs. We optimize with Adam [21] with an initial learning rate of 1e-5 and a cosine decay learning rate scheduler. The rest of the PPO hyper-parameters are the same as those described in [74]. To generate responses, we sample from the policy with a temperature $T = 2$.

## 4.5 Evaluation

After fine-tuning a model with RL, we evaluate it on the test-clean and test-other sets of LibriTTS. Each model is evaluated under several dimensions: intelligibility, speaker similarity, naturalness and waveform duration. We evaluate the first three dimension using the corresponding methods described in Section 3.2, while the latter is measured directly on the decoded waveform.

## 4.6 Experiments

This section shows the results of the experiments we conducted to fine-tune a NCLM from feedback with RL. We fine-tune the pre-trained VALL-E model with the different types of feedback described in Section 3.2, but also with two different target values of $KL(\pi, \rho)$, 12 and 100.

### 4.6.1 Waveform Length

With this type of experiments, we try to optimize the length of the waveform produced by the NCLM according to the feedback provided by the reward function. We tried to both decrease and increase the length of the generated audio sample down or up to a minimum or maximum length.

**Increasing Waveform Length** Figure 4.1 shows that in both experiments, the policies are able to reach the expected target $KL(\pi, \rho)$ and maintain it over time. When training with $KL(\pi, \rho) = 12$, the model is able to gradually increase the mean reward over time. When the target $KL(\pi, \rho) = 100$, the policy learns to obtain high mean rewards after few optimization steps.

Table 4.1 and Table 4.2 shows that the both the fine-tuned models are able to produce longer audio samples with respect to the pre-trained baseline. However, we also observe intelligibility, speaker similarity and naturalness degradation for both models. The degradation effect is reduced when the policy is constrained to reach a target $KL(\pi, \rho) = 12$.

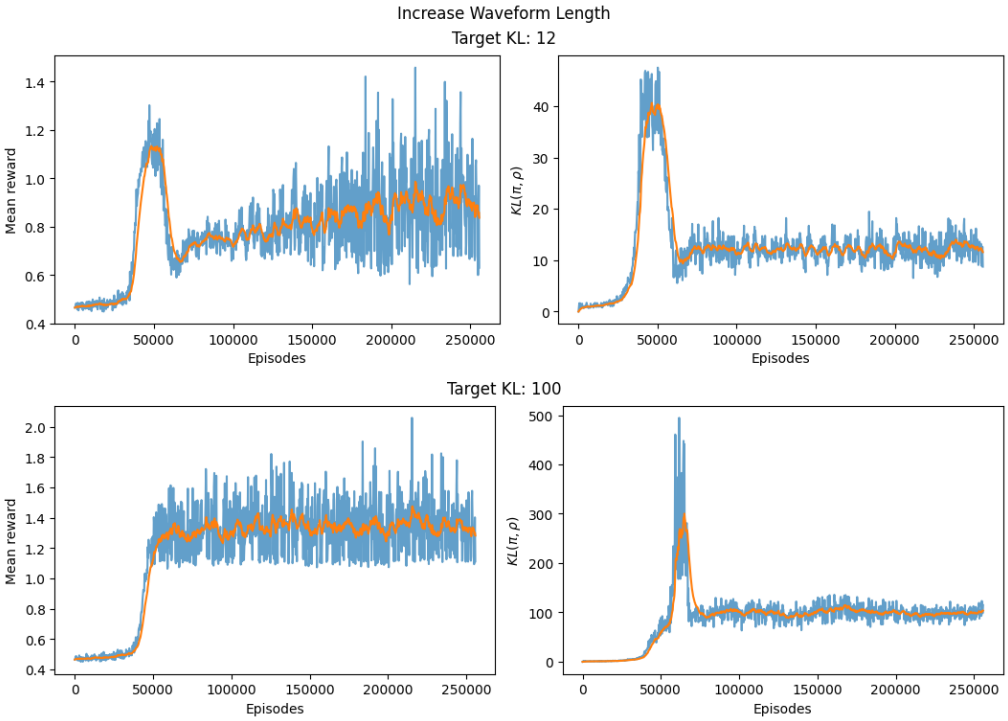


Figure 4.1: Learning curves for VALL-E fine-tuned with target $KL(\pi, \rho) = 12$ and $KL(\pi, \rho) = 100$ with increase waveform duration feedback.

| **Increase Waveform Length test-clean** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.654 | 4.291 | 34.707 | 4.002 |
| $KL(\pi, \rho) = 12$ | 0.637 | 4.217 | 44.758 | 5.828 |
| $KL(\pi, \rho) = 100$ | 0.500 | 4.121 | 85.276 | **9.872** |

Table 4.1: Results of VALL-E baseline and Fine-Tuned models from Increase Duration Feedback on LibriTTS test-clean

| **Increase Waveform Length test-other** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.636 | 4.171 | 51.820 | 3.377 |
| $KL(\pi, \rho) = 12$ | 0.627 | 4.005 | 61.271 | 4.957 |
| $KL(\pi, \rho) = 100$ | 0.510 | 3.634 | 91.550 | **8.239** |

Table 4.2: Results of VALL-E baseline and Fine-Tuned models from Increase Duration Feedback on LibriTTS test-other

**Decreasing Waveform Length** Figure 4.2 shows that, when the target $KL(\pi, \rho) = 12$, the model learns to reach and maintain the expected KL value, while when the target

$KL(\pi, \rho) = 100$ it does not. Despite that, both models learns to maximize the mean reward after few optimization steps.

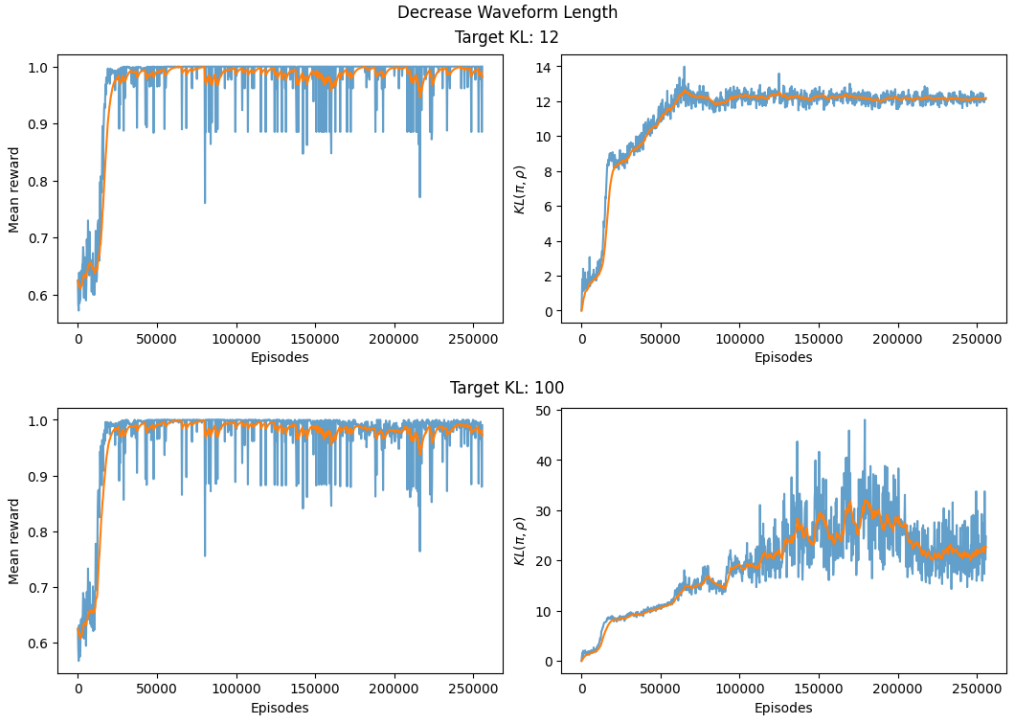


Figure 4.2: Learning curves for VALL-E fine-tuned with target $KL(\pi, \rho) = 12$ and $KL(\pi, \rho) = 100$ with decrease waveform duration feedback.

Tables 4.3 and 4.4 shows that both policies are able to produce shorter waveforms after fine-tuning. However, as in the previous experiment, we observe degradation for the other metrics.

| **Decrease Waveform Length test-clean** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.654 | 4.291 | 34.707 | 4.002 |
| $KL(\pi, \rho) = 12$ | 0.591 | 4.104 | 68.392 | 0.746 |
| $KL(\pi, \rho) = 100$ | 0.573 | 4.075 | 79.098 | **0.731** |

Table 4.3: Results of VALL-E baseline and Fine-Tuned models from Decrease Duration Feedback on LibriTTS test-clean

| **Decrease Waveform Length test-other** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.636 | 4.171 | 51.820 | 3.377 |
| $KL(\pi, \rho) = 12$ | 0.587 | 3.947 | 76.972 | 0.751 |
| $KL(\pi, \rho) = 100$ | 0.574 | 3.917 | 86.121 | **0.748** |

Table 4.4: Results of VALL-E baseline and Fine-Tuned models from Decrease Duration Feedback on LibriTTS test-other

### 4.6.2 Intelligibility

This experiment aims to increase the intelligibility of VALL-E by promoting the generation of waveforms whose transcription has a lower WER than the text prompt.

Figure 4.3 shows that when the target $KL(\pi, \rho) = 12$, the mean reward follows a constant trend over time, while when the target $KL(\pi, \rho) = 100$, the model learns to increase the reward after around 150k episodes.

From Tables 4.5 and 4.6 we can see that the fine-tuned models outperform the baseline in intelligibility. Specifically, the $KL(\pi, \rho) = 12$ model shows a WER reduction of 8.014% on the test-clean set and 10.107% on the test-other set, while the $KL(\pi, \rho) = 100$ model shows a WER reduction of 15.324% on test-clean and 20.954% on test-other. We also observe that the average duration of the synthetic waveforms is lower than the baseline for both models. For the $KL(\pi, \rho) = 100$ model, we observe a small degradation of speaker similarity and naturalness on both the datasets.

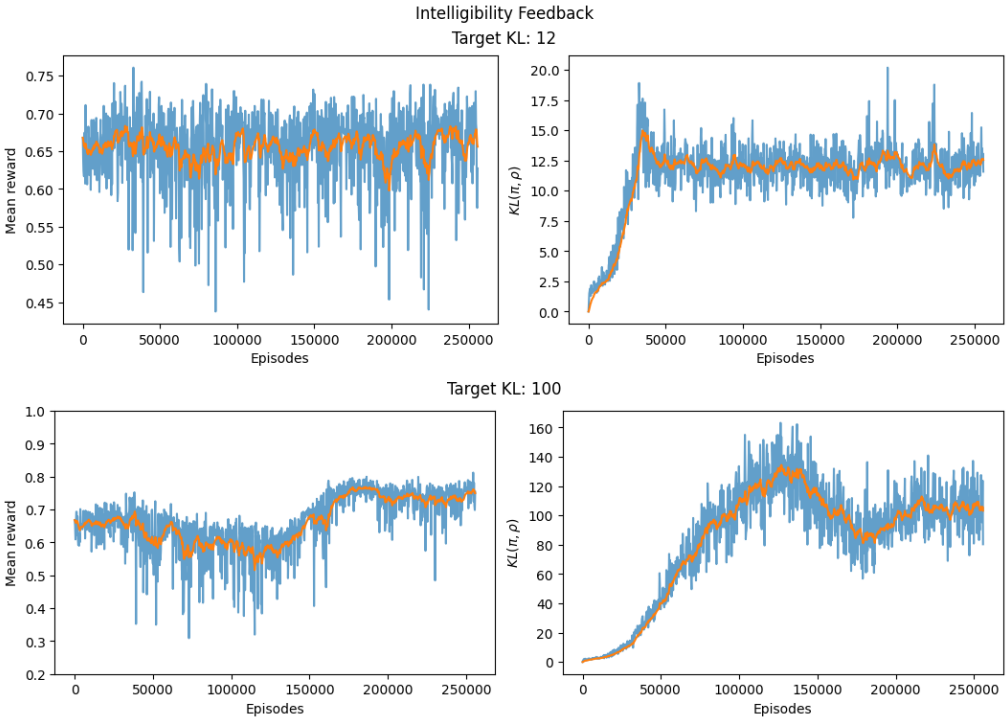


Figure 4.3: Learning curves for VALL-E fine-tuned with target $KL(\pi, \rho) = 12$ and $KL(\pi, \rho) = 100$ with increase intelligibility feedback.

| **Intelligibility test-clean** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.654 | 4.291 | 34.707 | 4.002 |
| $KL(\pi, \rho) = 12$ | 0.657 | 4.299 | 26.693 | 3.474 |
| $KL(\pi, \rho) = 100$ | 0.637 | 4.292 | **19.383** | 2.978 |

Table 4.5: Results of VALL-E baseline and Fine-Tuned models from WER Feedback on LibriTTS test-clean

| **Intelligibility test-other** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.636 | 4.171 | 51.820 | 3.377 |
| $KL(\pi, \rho) = 12$ | 0.633 | 4.165 | 41.713 | 2.907 |
| $KL(\pi, \rho) = 100$ | 0.606 | 4.086 | **30.866** | 2.524 |

Table 4.6: Results of VALL-E baseline and Fine-Tuned models from WER Feedback on LibriTTS test-other

### 4.6.3 Speaker Similarity

With this experiment we aim to fine-tune VALL-E so that it generates speech whose speaker is as similar as possible to the original one (the speaker of the audio prompt), by

increasing their SPK.

Figure 4.4 shows the learning curves for the speaker similarity experiments. Although the models reached and maintained the target $KL(\pi, \rho)$ value, we observe a decreasing mean reward as episodes grow for both models.

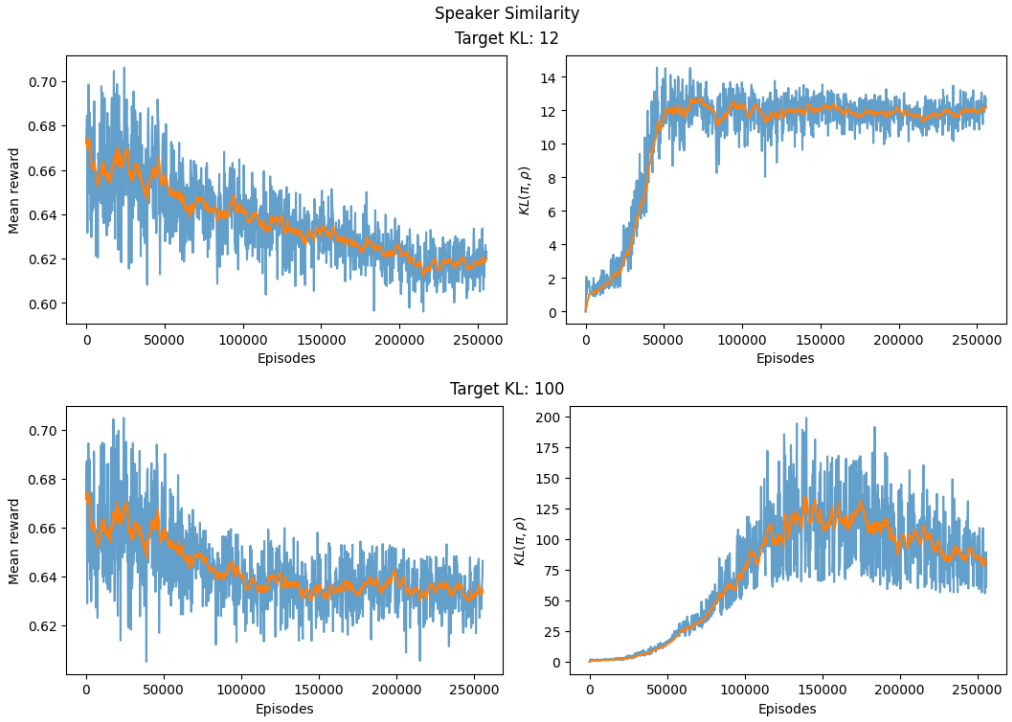


Figure 4.4: Learning curves for VALL-E fine-tuned with target $KL(\pi, \rho) = 12$ and $KL(\pi, \rho) = 100$ with increase speaker similarity feedback.

Table 4.7 and table 4.8 confirm that both models attempt to maximize the speaker similarity metric failed, since the baseline outperforms them, indicating a $SPK$ degradation after the fine-tuning process. We also observe WER degradation and a waveform duration reduction on both sets.

| **Speaker Similarity test-clean** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | **0.654** | 4.291 | 34.707 | 4.002 |
| $KL(\pi, \rho) = 12$ | 0.609 | 4.161 | 64.473 | 0.911 |
| $KL(\pi, \rho) = 100$ | 0.627 | 4.207 | 69.672 | 1.565 |

Table 4.7: Results of VALL-E baseline and Fine-Tuned models from SPK Feedback on LibriTTS test-clean

| Speaker Similarity test-other | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | **0.636** | 4.171 | 51.820 | 3.377 |
| $KL(\pi, \rho) = 12$ | 0.605 | 4.015 | 72.088 | 1.002 |
| $KL(\pi, \rho) = 100$ | 0.614 | 4.083 | 77.723 | 1.801 |

Table 4.8: Results of VALL-E baseline and Fine-Tuned models from SPK Feedback on LibriTTS test-other

**Ablation study** We argue that a possible reason for the results degradation of the fine-tuned NCLM is that the model struggles to generate responses with better speaker similarity with respect to the baseline. PPO is purely guided by exploration of the action space, where in our scenario actions are predicted discrete codes. We argue that the identity of a speaker can be seen as a time-invariant feature of a waveform, thus encoded by TiCodec's time-invariant quantizer rather than the time-level one. Since our action space does not include the time-invariant codes, a possible reason for the observed fine-tuning failure is the impossibility of the network to explore and find responses that encodes time-invariant features and thus, speaker identity features.

To support this claim, we conducted an ablation study on the baseline, VALL-E pre-trained on LibriTTS, where we evaluate it with random time-invariant codes instead of those extracted from the audio prompts. Table 4.9 shows that there is a big SPK gap between the waveform generated with true time-invariant codes and with random ones, providing an evidence that supports the claim above.

| Ablation study: speaker similarity | | |
|---|---|---|
| model | SPK (test-clean) | SPK (test-other) |
| True time-invariant codes | **0.654** | **0.636** |
| Random time-invariant codes | 0.318 | 0.305 |

Table 4.9: Results of the ablation study conducted on VALL-E baseline with and without true time-invariant codes

### 4.6.4 Naturalness with MOS predictor

From Figure 4.5, similarly as for the speaker similarity experiment, we observe a mean reward degradation as the number of episodes increases for both models, although the target KL value is reached and maintained.

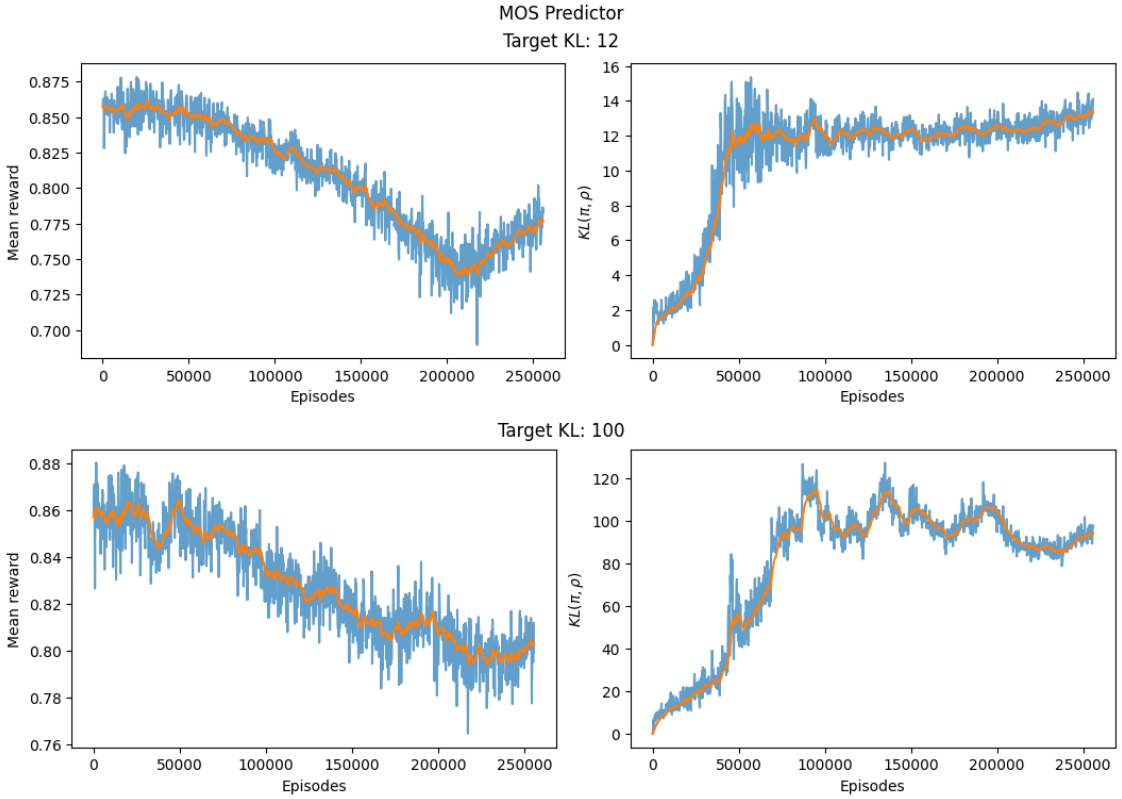


Figure 4.5: Learning curves for VALL-E fine-tuned with target $KL(\pi, \rho) = 12$ and $KL(\pi, \rho) = 100$ with increase NORESQA-MOS feedback.

Tables 4.10 and 4.11 shows that the fine-tuned model failed to maximize the feedback, since their NORESQA-MOS deteriorated in both experiments and test sets. We also observe WER, SPK and waveform duration degradation for both models.

| **Naturalness test-clean** | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.654 | **4.291** | 34.707 | 4.002 |
| $KL(\pi, \rho) = 12$ | 0.559 | 3.898 | 75.483 | 0.457 |
| $KL(\pi, \rho) = 100$ | 0.559 | 4.034 | 75.109 | 0.555 |

Table 4.10: Results of VALL-E baseline and Fine-Tuned models from NORESQA-MOS Feedback on LibriTTS test-clean

With a further experiment, we investigate the causes of the optimization failure. As for the speaker similarity experiment, we argue that a possible reason for the failure is that the NCLM struggles to generate responses having a NORESQA-MOS higher than the ones generated by the baseline.

We suspect that NORESQA-MOS struggles to output realistic values for audios that

| Naturalness test-other | | | | |
|---|---|---|---|---|
| model | SPK | NORESQA-MOS | WER | AVG Waveform Duration (s) |
| Pretrained | 0.636 | **4.171** | 51.820 | 3.377 |
| $KL(\pi, \rho) = 12$ | 0.563 | 3.812 | 82.768 | 0.486 |
| $KL(\pi, \rho) = 100$ | 0.562 | 3.925 | 82.547 | 0.636 |

Table 4.11: Results of VALL-E baseline and Fine-Tuned models from NORESQA-MOS Feedback on LibriTTS test-other

are already similar to the original ground truth speech samples, probably because it does not generalize well on data out of domain. We also suspect that NORESQA-MOS does not output realistic values for speech samples that are worsening during the RL optimization, but rather optimistic values. To support the first claim, instead of evaluating synthetic speech samples generated by the baseline, we evaluate the NORESQA-MOS of ground truth speech samples against themselves. Instead, to support the second claim, we silence a random 50% of the speech samples generated by the baseline and evaluate them with NORESQA-MOS.

Table 4.12 shows that, although the MOS values of the ground truth are higher than those of the baseline, the gap between them is relatively small, $0.062$ for test-clean and $0.130$ for test-other. We think that this evidence supports the first claim, since we expected much higher MOS values for ground truth samples, close to the maximum value of 5. We also observe that the silenced samples obtained MOS values that are lower than those of the baseline, but the gap between them is again relatively small, $0.149$ for test-clean and $0.082$ for test-other. We think that NORESQA-MOS should output lower values for those samples, since speech with such interruptions must not be considered of good naturalness, supporting the second claim.

| Speech samples | NORESQA-MOS (test-clean) | NORESQA-MOS (test-other) |
|---|---|---|
| Pretrained | 4.291 | 4.171 |
| Ground Truth | 4.353 | 4.301 |
| Pretrained w/ 50% silence | 4.142 | 4.089 |

Table 4.12: NORESQA-MOS results for different speech samples on LibriTTS test sets

# CHAPTER 5

# CONCLUSIONS

## 5.1 Conclusions, Limitations and Future Works

We have conducted experiments to fine-tune VALL-E with RL to optimize four types of feedback: waveform duration, intelligibility, speaker similarity and naturalness. We achieved our results by directly applying reward learning to speech generation, rather than build task specific training techniques.

We obtained mixed results. We were able to achieve good results when increasing or decreasing the generated waveform duration, since VALL-E easily learned to maximize the reward signal. However, compared to the baseline, the generated speech has lost its speaker identity, naturalness and intelligibility. For the intelligibility optimization task instead, we achieved positive results, since the fine-tuned model learned to decrease its WER over time after few RL steps. Moreover, the fine-tuned model showed a huge intelligibility increase on both unseen test datasets, while also preserving the speaker identity and naturalness. We were not able to improve the speaker similarity and NORESQA-MOS of the generated speech with reward learning. With both feedback, the fine-tuned model shows SPK, WER and NORESQA-MOS degradations on the test sets. We argue that a possible limiting factor for learning to increase the speaker similarity feedback is the time-invariant quantizer of our neural audio codec, TiCodec. With the ablation study we showed that a big percentage of the speaker identity resides in the time-invariant

codes, that are independent of the RL process; thus, the network could struggle to obtain reward increments when exploring. We also showed that NORESQA-MOS outputs values for ground truth samples that are really close to those of the baseline's samples. Moreover, NORESQA-MOS predicts values that are too optimistic for synthetic samples that are clearly unnatural. We suspect that both factors affect negatively the fine-tuning process, since the network could struggle to obtain reward increments for better samples and reward decrements for worse samples when exploring.

Although we achieved some positive results, our work has some limitations. With the available resources for this thesis, we were not able to use the original scaled version of VALL-E and neither working with bigger datasets than LibriTTS. The baseline we trained is not as strong as expected by means of intelligibility, so the improvements we achieved need to be consolidated by other experiments with stronger baselines trained on bigger datasets. Also, we only use objective metrics to evaluate the naturalness and speaker similarity of the fine-tuned models. Since the results of such metrics are yet not on par with those obtained by human perception, to further consolidate the results of the experiments we can replace objective metrics with subjective ones to evaluate the synthetic speeches generated by the fine-tuned NCLMs. As previously mentioned, another limitation in this work is the use of TiCodec's time-invariant codes; using a different neural audio codec could led to successful enhancements of speaker similarity. A last limitation to mention is the use of a MOS predictor as reward function instead of a MOS feedback derived by crowd-sourced human preferences. Using human feedback to evaluate naturalness is more realistic than objective metrics and it could help the fine-tuning process.

In the future, we would like to experiment with other NCLMs and different TTS datasets to consolidate the positive results of our experiments. We would like also experiment with mixed feedback functions, to prevent mode collapsing and degradation of speech quality. We could also incorporate human feedback in the RL loop, following the original RLHF pipeline [74]. Finally, since our experimental setup follows the theoretical Bradley-Terry preference model [9], we could replace the RL optimization setup with Direct Preference Optimization (DPO), a method that showed better or the same results than existing methods in fine-tuning LMs to align with human preferences [44].

# BIBLIOGRAPHY


[1] S. Ö. Arik et al. "Deep Voice: Real-time Neural Text-to-Speech". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 195–204. URL: http://proceedings.mlr.press/v70/arik17a.html.

[2] A. Baevski, S. Schneider, and M. Auli. "vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rylwJxrYDS.

[3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html.

[4] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? ". In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610623.

ISBN: 9781450383097. DOI: 10.1145/3442188.3445922. URL: https://doi.org/10.1145/3442188.3445922.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (2003), pp. 1137–1155. URL: http://jmlr.org/papers/v3/bengio03a.html.

[6] M. Bernard and H. Titeux. "Phonemizer: Text to Phones Transcription for Multiple Languages in Python". In: *Journal of Open Source Software* 6.68 (2021), p. 3958. DOI: 10.21105/joss.03958. URL: https://doi.org/10.21105/joss.03958.

[7] J. Betker. *Better speech synthesis through scaling*. 2023. arXiv: 2305.07243 [cs.SD].

[8] K. Br, E. Mp, A. Explained, and K. Brandenburg. "Mp3 And Aac Explained". In: (Mar. 2001).

[9] R. A. Bradley and M. E. Terry. "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345. ISSN: 00063444. URL: http://www.jstor.org/stable/2334029 (visited on 02/27/2024).

[10] T. B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[11] E. Casanova, J. Weber, C. D. Shulby, A. C. Júnior, E. Gölge, and M. A. Ponti. "YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for Everyone". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 2709–2720. URL: https://proceedings.mlr.press/v162/casanova22a.html.

[12] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi. "High Fidelity Neural Audio Compression". In: *CoRR* abs/2210.13438 (2022). DOI: 10.48550/ARXIV.2210.13438. arXiv: 2210.13438. URL: https://doi.org/10.48550/arXiv.2210.13438.

[13] J. Devlin, M. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[14] R. Gozalo-Brizuela and E. C. Garrido-Merchán. “ChatGPT is not all you need. A State of the Art Review of large Generative AI models”. In: *CoRR* abs/2301.04655 (2023). DOI: 10.48550/ARXIV.2301.04655. arXiv: 2301.04655. URL: https://doi.org/10.48550/arXiv.2301.04655.

[15] W. Hsu, B. Bolte, Y. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed. “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units”. In: *IEEE ACM Trans. Audio Speech Lang. Process.* 29 (2021), pp. 3451–3460. DOI: 10.1109/TASLP.2021.3122291. URL: https://doi.org/10.1109/TASLP.2021.3122291.

[16] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. USA: Prentice Hall PTR, 2000. ISBN: 0130950696.

[17] L. P. Kaelbling, M. L. Littman, and A. W. Moore. “Reinforcement Learning: A Survey”. In: *CoRR* cs.AI/9605103 (1996). URL: https://arxiv.org/abs/cs/9605103.

[18] T. Kaufmann, P. Weng, V. Bengs, and E. Hüllermeier. “A Survey of Reinforcement Learning from Human Feedback”. In: *CoRR* abs/2312.14925 (2023). DOI: 10.48550/ARXIV.2312.14925. arXiv: 2312.14925. URL: https://doi.org/10.48550/arXiv.2312.14925.

[19] Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik, and G. Irving. “Alignment of Language Agents”. In: *CoRR* abs/2103.14659 (2021). arXiv: 2103.14659. URL: https://arxiv.org/abs/2103.14659.

[20] J. Kim, K. Lee, S. Chung, and J. Cho. “CLaM-TTS: Improving Neural Codec Language Model for Zero-Shot Text-to-Speech”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=ofzeypWosV.

[21] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[22] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2014. URL: http://arxiv.org/abs/1312.6114.

[23] N. R. Koluguri, T. Park, and B. Ginsburg. “TitaNet: Neural Model for Speaker Representation with 1D Depth-Wise Separable Convolutions and Global Context”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*. IEEE, 2022, pp. 8102–8106. DOI: 10.1109/ICASSP43922.2022.9746806. URL: https://doi.org/10.1109/ICASSP43922.2022.9746806.

[24] O. Kuchaiev et al. “NeMo: a toolkit for building AI applications using Neural Modules”. In: *CoRR* abs/1909.09577 (2019). arXiv: 1909.09577. URL: http://arxiv.org/abs/1909.09577.

[25] S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79 –86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[26] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville. “MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 14881–14892.

URL: https://proceedings.neurips.cc/paper/2019/hash/6804c9bca0a615bdb9374d00a9fcba59-Abstract.html.

[27] K. Lakhotia et al. “Generative Spoken Language Modeling from Raw Audio”. In: *CoRR* abs/2102.01192 (2021). arXiv: 2102.01192. URL: https://arxiv.org/abs/2102.01192.

[28] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *CoRR* abs/1910.13461 (2019). arXiv: 1910.13461. URL: http://arxiv.org/abs/1910.13461.

[29] F. Li. *VALL-E: A neural codec language model*. 2023. URL: http://github.com/lifeiteng/vall-e.

[30] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu. “Neural Speech Synthesis with Transformer Network”. In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 6706–6713. DOI: 10.1609/AAAI.V33I01.33016706. URL: https://doi.org/10.1609/aaai.v33i01.33016706.

[31] Z. C. Lipton. “A Critical Review of Recurrent Neural Networks for Sequence Learning”. In: *CoRR* abs/1506.00019 (2015). arXiv: 1506.00019. URL: http://arxiv.org/abs/1506.00019.

[32] Y. Liu, R. Xue, L. He, X. Tan, and S. Zhao. “DelightfulTTS 2: End-to-End Speech Synthesis with Adversarial Vector-Quantized Auto-Encoders”. In: *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*. Ed. by H. Ko and J. H. L. Hansen. ISCA, 2022, pp. 1581–1585. DOI: 10.21437/INTERSPEECH.2022-277. URL: https://doi.org/10.21437/Interspeech.2022-277.

[33] P. Manocha and A. Kumar. “Speech Quality Assessment through MOS using Non-Matching References”. In: *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*. Ed. by H. Ko and J. H. L. Hansen. ISCA, 2022, pp. 654–658. DOI: 10.21437/INTERSPEECH.2022-407. URL: https://doi.org/10.21437/Interspeech.2022-407.

[34] *Method for the subjective assessment of intermediate quality level of audio systems*. ITU-R BS.1534-3, 2015.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by M. Balcan and K. Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 1928–1937. URL: http://proceedings.mlr.press/v48/mniha16.html.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[37] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio”. In: *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*. ISCA, 2016, p. 125. URL: http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html.

[38] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. “Neural Discrete Representation Learning”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett. 2017, pp. 6306–6315. URL: https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

[39] OpenAI. “GPT-4 Technical Report”. In: *CoRR* abs/2303.08774 (2023). DOI: 10.48550/ARXIV.2303.08774. arXiv: 2303.08774. URL: https://doi.org/10.48550/arXiv.2303.08774.

[40] L. Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

[41] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. “Librispeech: An ASR corpus based on public domain audio books”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*. IEEE, 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964. URL: https://doi.org/10.1109/ICASSP.2015.7178964.

[42] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[43] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[44] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”. In: *CoRR* abs/2305.18290 (2023). DOI: 10.48550/ARXIV.2305.18290. arXiv: 2305.18290. URL: https://doi.org/10.48550/arXiv.2305.18290.

[45] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. “Hierarchical Text-Conditional Image Generation with CLIP Latents”. In: *CoRR* abs/2204.06125 (2022). DOI: 10.48550/ARXIV.2204.06125. arXiv: 2204.06125. URL: https://doi.org/10.48550/arXiv.2204.06125.

[46] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. “Zero-Shot Text-to-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8821–8831. URL: http://proceedings.mlr.press/v139/ramesh21a.html.

[47] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu. “FastSpeech: Fast, Robust and Controllable Text to Speech”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 3165–3174. URL: https://proceedings.neurips.cc/paper/2019/hash/f63f65b503e22cb970527f23c9ad7db1-Abstract.html.

[48] Y. Ren, T. Wang, J. Yi, L. Xu, J. Tao, C. Zhang, and J. Zhou. “Fewer-token Neural Speech Codec with Time-invariant Codes”. In: *CoRR* abs/2310.00014 (2023). DOI: 10.48550/ARXIV.2310.00014. arXiv: 2310.00014. URL: https://doi.org/10.48550/arXiv.2310.00014.

[49] M. R. Schroeder. “The Speech Signal”. In: *Computer Speech: Recognition, Compression, Synthesis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 105–108. ISBN: 978-3-662-03861-1. DOI: 10.1007/978-3-662-03861-1_7. URL: https://doi.org/10.1007/978-3-662-03861-1_7.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[51] J. Shen et al. “Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 4779–4783. DOI: 10.1109/ICASSP.2018.8461368.

[52] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[53] Y. Tabet and M. Boughazi. “Speech synthesis techniques. A survey”. In: *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*. 2011, pp. 67–70. DOI: 10.1109/WOSSPA.2011.5931414.

[54] H. Tachibana, K. Uenoyama, and S. Aihara. “Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*. IEEE, 2018, pp. 4784–4788. DOI: 10.1109/ICASSP.2018.8461829. URL: https://doi.org/10.1109/ICASSP.2018.8461829.

[55] X. Tan, T. Qin, F. K. Soong, and T. Liu. “A Survey on Neural Speech Synthesis”. In: *CoRR* abs/2106.15561 (2021). arXiv: 2106.15561. URL: https://arxiv.org/abs/2106.15561.

[56] P. Taylor. *Text-to-Speech Synthesis*. Cambridge University Press, 2009.

[57] H. Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *CoRR* abs/2302.13971 (2023). DOI: 10.48550/ARXIV.2302.13971. arXiv: 2302.13971. URL: https://doi.org/10.48550/arXiv.2302.13971.

[58] J. Valin, G. Maxwell, T. B. Terriberry, and K. Vos. “High-Quality, Low-Delay Music Coding in the Opus Codec”. In: *CoRR* abs/1602.04845 (2016). arXiv: 1602.04845. URL: http://arxiv.org/abs/1602.04845.

[59] S. Vasquez and M. Lewis. “MelNet: A Generative Model for Audio in the Frequency Domain”. In: *CoRR* abs/1906.01083 (2019). arXiv: 1906.01083. URL: http://arxiv.org/abs/1906.01083.

[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[61] *Vocabulary for performance and quality of service*. ITU-T Rec. P.10, 2006.

[62] C. Wang et al. “Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers”. In: *CoRR* abs/2301.02111 (2023). DOI: 10.48550/ARXIV.2301.02111. arXiv: 2301.02111. URL: https://doi.org/10.48550/arXiv.2301.02111.

[63] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao. “Deep Reinforcement Learning: A Survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–15. DOI: 10.1109/TNNLS.2022.3207346.

[64] Y. Wang et al. “Tacotron: Towards End-to-End Speech Synthesis”. In: *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*. Ed. by F. Lacerda. ISCA, 2017, pp. 4006–4010. DOI: 10.21437/INTERSPEECH.2017-1452. URL: https://doi.org/10.21437/Interspeech.2017-1452.

[65] L. Weidinger et al. “Ethical and social risks of harm from Language Models”. In: *CoRR* abs/2112.04359 (2021). arXiv: 2112.04359. URL: https://arxiv.org/abs/2112.04359.

[66] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas. “VideoGPT: Video Generation using VQ-VAE and Transformers”. In: *CoRR* abs/2104.10157 (2021). arXiv: 2104.10157. URL: https://arxiv.org/abs/2104.10157.

[67] D. Yang, S. Liu, R. Huang, J. Tian, C. Weng, and Y. Zou. “HiFi-Codec: Group-residual Vector quantization for High Fidelity Audio Codec”. In: *CoRR* abs/2305.02765 (2023). DOI: 10.48550/ARXIV.2305.02765. arXiv: 2305.02765. URL: https://doi.org/10.48550/arXiv.2305.02765.

[68] Y. Yao, X. Wang, Y. Ma, H. Fang, J. Wei, L. Chen, A. Anaissi, and A. Braytee. "Conditional Variational Autoencoder with Balanced Pre-training for Generative Adversarial Networks". In: *9th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2022, Shenzhen, China, October 13-16, 2022*. Ed. by J. Z. Huang, Y. Pan, B. Hammer, M. K. Khan, X. Xie, L. Cui, and Y. He. IEEE, 2022, pp. 1–10. DOI: 10.1109/DSAA54385.2022.10032367. URL: https://doi.org/10.1109/DSAA54385.2022.10032367.

[69] Z. Yao, L. Guo, X. Yang, W. Kang, F. Kuang, Y. Yang, Z. Jin, L. Lin, and D. Povey. "Zipformer: A faster and better encoder for automatic speech recognition". In: *CoRR* abs/2310.11230 (2023). DOI: 10.48550/ARXIV.2310.11230. arXiv: 2310.11230. URL: https://doi.org/10.48550/arXiv.2310.11230.

[70] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi. "SoundStream: An End-to-End Neural Audio Codec". In: *IEEE ACM Trans. Audio Speech Lang. Process.* 30 (2022), pp. 495–507. DOI: 10.1109/TASLP.2021.3129994. URL: https://doi.org/10.1109/TASLP.2021.3129994.

[71] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu. "LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech". In: *CoRR* abs/1904.02882 (2019). arXiv: 1904.02882. URL: http://arxiv.org/abs/1904.02882.

[72] C. Zhang et al. "A Complete Survey on Generative AI (AIGC): Is ChatGPT from GPT-4 to GPT-5 All You Need?" In: *CoRR* abs/2303.11717 (2023). DOI: 10.48550/ARXIV.2303.11717. arXiv: 2303.11717. URL: https://doi.org/10.48550/arXiv.2303.11717.

[73] C. Zhang et al. "One Small Step for Generative AI, One Giant Leap for AGI: A Complete Survey on ChatGPT in AIGC Era". In: *CoRR* abs/2304.06488 (2023). DOI: 10.48550/ARXIV.2304.06488. arXiv: 2304.06488. URL: https://doi.org/10.48550/arXiv.2304.06488.

[74] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. F. Christiano, and G. Irving. "Fine-Tuning Language Models from Human Preferences". In: *CoRR* abs/1909.08593 (2019). arXiv: 1909.08593. URL: http://arxiv.org/abs/1909.08593.