

Enhancing Emotional Expression in Text-to-Speech Models through Reinforcement Learning with AI Feedback

Roshan Nagaram



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-23

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-23.html>

April 28, 2024

Copyright © 2024, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Enhancing Emotional Expression in Text-to-Speech Models through Reinforcement Learning with AI Feedback

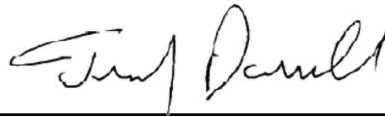
by Roshan Nagaram

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,
University of California at Berkeley, in partial satisfaction of the requirements for the
degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

Committee:



Professor Trevor Darrell
Research Advisor

4/27/2024

(Date)

* * * * *



Professor Sergey Levine
Second Reader

4/29/24

(Date)

Enhancing Emotional Expression in Text-to-Speech Models through Reinforcement
Learning with AI Feedback

by

Roshan Nagaram

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Sergey Levine

Spring 2024

Enhancing Emotional Expression in Text-to-Speech Models through Reinforcement
Learning with AI Feedback

Copyright 2024
by
Roshan Nagaram

Abstract

Enhancing Emotional Expression in Text-to-Speech Models through Reinforcement Learning with AI Feedback

by

Roshan Nagaram

Master of Science in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

This paper investigates the application of Reinforcement Learning (RL), particularly Reinforcement Learning with Artificial Intelligence Feedback (RLAIF), to enhance emotional expression in text-to-speech (TTS) models. RLAIF stems from RLHF (Reinforcement Learning with Human Feedback). These techniques aim to leverage feedback from humans or other AI models in order to train and tune a set of AI Models. Techniques such as RLAIF are powerful when training on a set of objectives that are complex, abstract, and difficult to mathematically define. Through its capabilities, RLAIF has become crucial in ensuring Large Language Models align with human values. In this paper, we intend to utilize these same capabilities in the text-to-speech domain. Similar to how RLAIF can be used to align Large Language Models with complex objectives such as mitigating toxicity, this paper aims to use RLAIF to align text-to-speech models to express certain emotions. By exploring different base model architectures in addition to specific RL techniques such as Reward Weighted Regression (RWR) and Proximal Policy Optimization (PPO), we are able to train text-to-speech models to better express emotion using feedback from an emotion predictor model. Ultimately, this research seeks to demonstrate, through quantitative and qualitative data, the effectiveness of RLAIF as a method for tuning TTS models to achieve more nuanced emotional expression.

To my family

Contents

Contents	ii
List of Figures	iv
List of Tables	vi
1 Introduction	1
2 Related Work	3
2.1 Training Diffusion Models With Reinforcement Learning	3
2.2 Reinforcement Learning for Emotional Text-to-Speech Synthesis With Improved Emotion Discriminability	3
2.3 Learning without Forgetting	4
3 Overview of Utilized Models and Datasets	5
3.1 Text-To-Speech Models	5
3.2 Emotion Predictors	7
3.3 LJ Speech Dataset	8
4 Methods and Strategies	10
4.1 Problem Statement	10
4.2 Reward Feedback and Quality	10
4.3 Reward Weighted Regression	11
4.4 Ensemble Based Learning for RWR	12
4.5 Markov Decision Processes (MDP) and Diffusion Models	13
4.6 PPO/DDPO	14
4.7 PPO/DDPO with Learning without Forgetting	15
5 Experiments and Analysis of Results	17
5.1 Experiments	17
5.2 Results	25
6 Discussion, Limitations, and Future Work	29

Bibliography	31
7 Appendix	33
7.1 Table of Audio URLs and Waveplots	33
7.2 Algorithms	43

List of Figures

3.1	Grad-TTS architecture (Popov et al.[16])	5
3.2	FastPitch Architecture (Łańcucki [11])	7
3.3	Wav2Vec Architecture (Baevski et al. [2])	8
4.1	RWR Architecture	12
4.2	Over-optimization from DDPO can occur if the reward signals only anger and not audio fidelity or fluency. The text-to-speech model can over time devolve from these reward signals into producing gibberish.	16
4.3	DDPO combined with Learning without Forgetting trains on both the angry signals as well as the original text-to-speech fluency objective allowing for more stable learning.	16
5.1	Performance for Reward Weighted Regression on various Learning Rates for Grad-TTS	18
5.2	RWR on Fastpitch using different sets of losses	19
5.3	Ensemble Eval Performance with various Learning Rates	20
5.4	Ensemble Dunn Predictor Performance with various Learning Rates	21
5.5	Ensemble Superb Predictor Performance with various Learning Rates	21
5.6	PPO Batch size ablation	22
5.7	PPO-LWF Beta ablation	24
5.8	Angry Reward vs Reward Queries for various RL Techniques	25
5.9	Waveplot for Checkpoint on prompt "that is very unfair"	27
5.10	Waveplot for RWR on prompt "that is very unfair"	27
5.11	Waveplot for PPO on prompt "that is very unfair"	27
5.12	Waveplot for PPO with LWF on prompt "that is very unfair"	28
7.1	Waveplot for 1a	34
7.2	Waveplot for 1b	34
7.3	Waveplot for 1c	35
7.4	Waveplot for 1d	35
7.5	Waveplot for 2a	36
7.6	Waveplot for 2b	36
7.7	Waveplot for 2c	37

7.8	Waveplot for 2d	37
7.9	Waveplot for 3a	38
7.10	Waveplot for 3b	38
7.11	Waveplot for 3c	39
7.12	Waveplot for 3d	39
7.13	Waveplot for 4a	40
7.14	Waveplot for 4b	40
7.15	Waveplot for 4c	41
7.16	Waveplot for 4d	41
7.17	Waveplot for 5a	42
7.18	Waveplot for 5b	42
7.19	Waveplot for 5c	43
7.20	Waveplot for 5d	43

List of Tables

5.1	Qualitative Comparison of Audio Samples Across Different RL Techniques with Hyperlinks (Click to Listen). Raw URL to the audio will be posted in the Appendix.	26
7.1	URLs and Waveplots 5.1	33

Acknowledgments

I would like to thank both of my professors, Professor Darrell and Professor Levine. I would like to thank Baifeng Shi for introducing me to the Darrell group. Thank you to Archit Raichura for all of his help and support. Thank you to all my friends and family.

Chapter 1

Introduction

Reinforcement Learning from Human Feedback (RLHF) has been one of the biggest developments recently in the field of machine learning. As shown in Wang et al. [22], with the rise of Large Language Models, RLHF has been critical for the tuning and alignment of these models in order to provide more helpful and harmless responses. However, RLHF due to its inherent dependency on human feedback faces fundamental challenges when it comes to scale. In order to address these challenges, researchers such as Lee et al. [12], have experimented with the use of Reinforcement Learning from AI feedback (RLAIF) and found comparable performance to RLHF while addressing potential issues of scale. Through the use of AI reward-based feedback, RLAIF allows models to gain a nuanced understanding of potentially complicated underlying reward functions. Given these capabilities, RLAIF has been used to tune models in fields such as natural language processing and computer vision where training objectives can be incredibly complex and therefore, difficult to formulate.

Text-to-speech synthesis is another domain that contains abstract training objectives but has been relatively underexplored in the application of RLAIF. Past work in the field of emotional text-to-speech synthesis has not considered the use of typical reinforcement learning algorithms, though emotion predictor feedback has been commonly incorporated into losses [8].

In this paper, our goal is to apply RLAIF to a text-to-speech model to dynamically adjust its output for conveying specific tones and emotions during synthesis. We hypothesize that if these techniques can effectively express one emotion with greater nuance, their efficacy should inherently extend to other emotional states. Thus, our primary focus here is to demonstrate the enhancement of text-to-speech models in conveying anger, with the intent to illustrate their potential applicability across various emotional contexts.

In order to accomplish this, we employed an existing diffusion-based text-to-speech model and a transformer-based text-to-speech model alongside a model capable of determining the probability of an emotion, given speech, thereby providing reward feedback. We experimented with various RL techniques for training such as Reward Weighted Regression (RWR) and Proximal Policy Optimization (PPO) across diffusion steps. In conjunction with exploring these various RL strategies, we experimented with modifying supervised learning strate-

gies such as Learning without Forgetting and utilizing an ensemble of models to generate reward feedback. Our various experiments demonstrated overall success in being able to tune a diffusion-based text-to-speech model to express emotions to a noticeably higher degree.

Our hope in this paper is to demonstrate the powerful capabilities of RLAIIF when it comes to tackling challenges in the text-to-speech domain. By specifically showcasing the ability to enhance the emotional nuance of generated speech, we hope to not only validate attributes such as the scalability of RLAIIF but also open the door to addressing more abstract issues specific to the text-to-speech domain.

Before we discuss in-depth aspects such as the models utilized, the RL techniques that were implemented, the results we obtained for our experiments, and the analysis of these results, it is important to first discuss previous work that helped lay the foundation for this paper.

Chapter 2

Related Work

While the use of RLAIIF has been underexplored in the domain of text-to-speech, related works in adjacent domains as well as various techniques within the text-to-speech field provide valuable inspiration and guidance for the implementations explored in this paper. The following section highlights key works that have laid the groundwork for this paper.

2.1 Training Diffusion Models With Reinforcement Learning

Black et al. [3] explores the use of RLAIIF for diffusion-based text-to-image models. A key insight of the paper that we rely on in the text-to-speech domain is the perspective of the denoising diffusion process as a multi-step Markov Decision Process (MDP). Using a diffusion-based text-to-speech model would allow us to follow a similar process as what is presented in the paper. Going from x_t to x_{t-1} is viewed as being taken under a policy, and the selection of state x_{t-1} under the denoising process is viewed as the action itself. This allows us to directly compute log probabilities for the actions as used in importance sampling or policy gradient methods and therefore consider any arbitrary reward function. The applications of this paper to our experiments will be discussed in greater detail in the methods and strategies section.

2.2 Reinforcement Learning for Emotional Text-to-Speech Synthesis With Improved Emotion Discriminability

Liu, Sisman, and Li [14] explore the idea of applying RL to emotional text-to-speech synthesis and the authors claim it to be "the first study of reinforcement learning in emotional text-to-speech synthesis." The encoder-decoder framework used for the underlying text-to-

speech model is based on Tacotron2, ultimately outputting mel-spectrograms. In order to handle emotion selection, an emotion embedding is also added between the encoder and decoder step and used to convert from a fixed set of emotion choices. The spectrogram is used in an emotion recognition model following the architecture in Chen et al. [5]. The probability of the requested emotion is used as the reward. The output spectrograms are viewed as actions in the RL setting and the log probabilities of spectrogram outputs from the decoder are used for policy gradient in the actual optimizations.

Our paper seeks to build off this paper by exploring different RL strategies such as those found in [3] as well as different base model architectures. Some key differences between our paper and this paper include the fact that [14] directly uses the mel-spectrogram for emotion recognition. In addition, only policy gradient is used here, for an importance sampling-based technique (like PPO) would not be used without a setup similar to diffusion.

2.3 Learning without Forgetting

Li and Hoiem [13] present Learning without Forgetting. This is a technique used in traditional supervised multi-task learning that is particularly popular for vision tasks. The concept centers around maintaining the older parameters and outputs before training the new task-specific parameters. When training for a new task, both the old task parameters and the new task-specific parameters are trained by taking the loss over the weighted sum of the losses for old tasks and new tasks. In the paper, only the old losses are weighted (with a weight ≤ 1 to encourage training for the new task). In our experiments, we take inspiration from this concept to help prevent over-optimization towards a reward function, which was mentioned as a concern with the training style in Black et al. [3], and prevent degradation of audio quality. We believe this is particularly important in the architecture setup that we use because unlike in [3] there is no intermediate description model for the diffusion output that can serve as a regularizer to ensure basic quality. The applications of this paper to our experiments will be discussed in greater detail in the methods and strategies section.

Chapter 3

Overview of Utilized Models and Datasets

In this chapter, we present descriptions of the models and datasets employed in our study. Given our objective to fine-tune a pre-existing text-to-speech model, careful consideration was given to selecting a text-to-speech model with robust learning capabilities, as well as an emotion detection model proficient in guiding the training process.

3.1 Text-To-Speech Models

Grad-TTS

Grad-TTS is one of the models used for our experiments because it is a diffusion-based text-to-speech model. This makes it suitable to use as the underlying model to train with the approach mentioned in Black et al. [3]. It is a common practice in text-to-speech models to

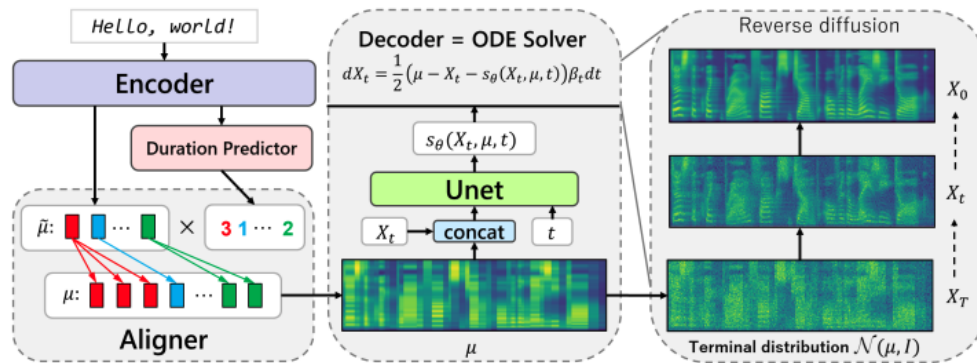


Figure 3.1: Grad-TTS architecture (Popov et al.[16])

predict a mel-spectrogram and then use a vocoder to convert it to speech because predicting end-to-end causes both speed and quality issues. These vocoders have explored generative style approaches, including diffusion and variational autoencoders, but a diffusion approach has not been used before for the spectrogram.

One of the most popular models in the past, Tacotron2 (Shen et al. [19]), used a sequence-to-sequence encoder-decoder architecture with attention and was autoregressive. The following year, FastSpeech was released which aimed to improve speed issues caused by the autoregressive nature of Tacotron2 but it required an external aligner (Ren et al. [17]). As a result, Glow-TTS (Kim et al. [10]) was developed to encapsulate learning alignment internally through an algorithm called Monotonic Alignment Search which maps text to spectrograms.

Taking a look at Figure 3.1, we see that Grad-TTS is composed of 3 main modules that together infer acoustic features that impact speech synthesis. The first two modules, the encoder and the duration predictor take in the input text data in addition to other data and produce monotonic alignments. Grad-TTS uses the monotonic alignment search found in Glow-TTS to pass aligned encoder outputs to the decoder (the last main module).

The decoder is a Diffusion Probabilistic Model, a type of model popular in other generation domains such as image and video synthesis.

$$dX_t = \frac{1}{2}(\mu - X_t - s_\theta(X_t, \mu, t))\beta_t dt$$

The equation shown above is then solved backwards and used in conjunction with the outputs of the monotonic aligner in order to perform the reverse diffusion process for generation. In the equation provided above $s_\theta(X_t, \mu, t)$ is a trained neural network fitted with parameters θ .

This last module, the decoder, will be important to the training presented later in the paper as it provides an opportunity to explore importance sampling based techniques.

FastPitch/FastSpeech

Fastpitch (developed by Łańcucki [11]) is a parallelized version of the text-to-speech model Fastspeech (developed by Ren et al. [17]). Fastpitch, unlike the previously mentioned Grad-TTS model, is not based on diffusion but rather the transformer-based architecture. For the purposes of this paper, Fastpitch is limited to policy gradient style techniques that take into account cumulative model output. Therefore, unlike Grad-TTS, we do not experiment with importance sampling-based techniques.

Despite this attribute, Fastpitch is unique in containing separate modules and losses for energy, pitch, and duration prediction. As shown in Figure 3.2, the Fastpitch architecture is based on two feedforward Transformer stacks. The first of these stacks is responsible for generating a hidden representation that a CNN can later process to predict attributes such as pitch, duration, and energy.

Having separate losses for pitch, duration, and energy increases customization for our RL techniques. In order to solidify the capabilities of baseline policy gradient techniques we

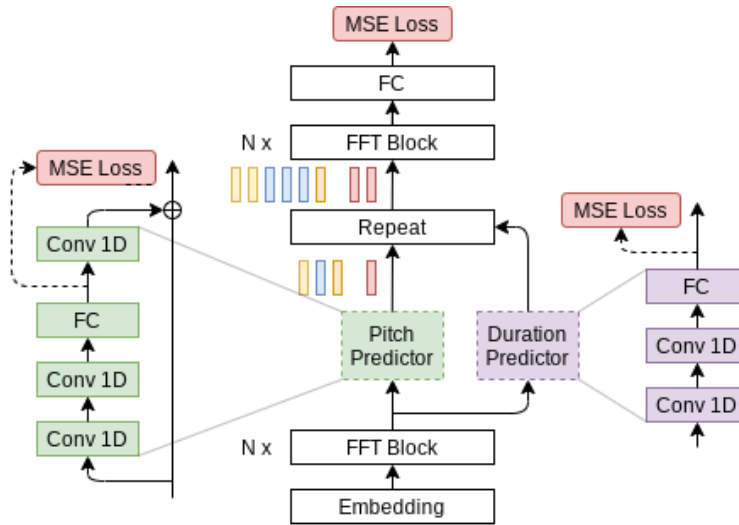


Figure 3.2: FastPitch Architecture (Łańcucki [11])

experiment with training on this architecture. Specifically, we can experiment with tuning only subsets of losses such as pitch and energy. These attributes may be more important in generating emotional speech, particularly anger.

Applying policy gradient style techniques to this architecture will allow us to get robust results for our baseline which we can then compare with importance sampling-based techniques used for tuning other architectures such as Grad-TTS. Furthermore, it helps better ascertain whether more advanced techniques such as importance sample-based techniques are better or if the architecture is a limiting factor towards performance.

3.2 Emotion Predictors

Superb Emotion Predictor

Emotion detection from speech is a problem that has been well explored in the speech synthesis domain. The problem is simple: given an audio file, classify the audio into one of the supported emotions and assign a probability to each emotion.

One of the emotion detectors that we employed in our training pipeline was the superb predictor (Superb [21]). The superb predictor is trained on the Interactive Emotional Dyadic Motion Capture (IEMOCAP) database (Busso et al. [4]). This database contains annotated video and audio data about emotional expressions in addition to attributes such as valence, activation, and dominance.

The creators of the model train a customized wav2vec model on the Superb Emotion Recognition Task (Yang et al. [23]). The wav2vec model is a model developed by Baevski

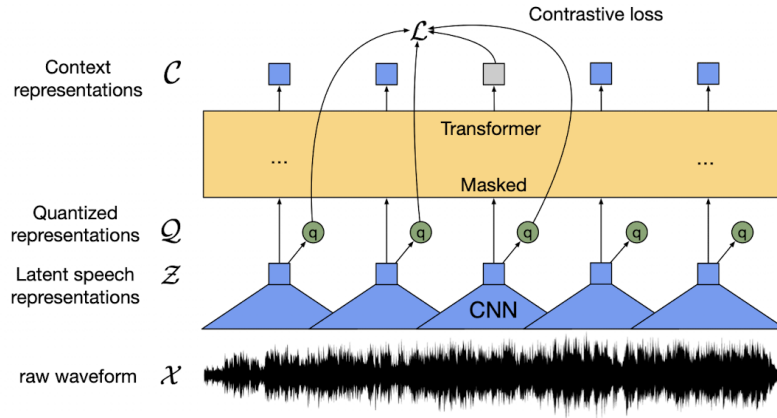


Figure 3.3: Wav2Vec Architecture (Baevski et al. [2])

et al. [2] that enables learning powerful speech representation through the architecture presented in Figure 3.3. These powerful representations, once learned, can be utilized on the Superb Emotion Recognition Task to train and test the emotion recognition capabilities of the model.

Dunn Predictor

The Dunn Predictor (DunnBC22 [6]) is another emotion detection model similar to the Superb Predictor. Like the Super Predictor, it is based on the wav2vec architecture. However, instead of being fine-tuned on the IEMOCAP dataset, the Dunn Predictor is tuned on a separate customized dataset of emotion recognition (Babko [1]).

When experimenting with the Dunn and Superb Emotion Predictor, we found the Superb Emotion Predictor to be qualitatively better at identifying emotions from audio clips. This could be due to a variety of reasons such as each model being trained on different datasets as well as different training procedures. However, the distinction between the performance will be important when experimenting with multiple reward signals for training.

3.3 LJ Speech Dataset

The LJ Speech Dataset (Ito and Johnson [9]) is a dataset of audio clips with corresponding text prompts. In all of our training pipelines, the LJ Speech Dataset is utilized. Although the LJ Speech Dataset is not composed of emotionally diverse data, just monotone audio and text prompts, it is one of the standard datasets for text-to-speech training. As such, it provides ease of use since many extraneous attributes pertinent to emotion recognition have already been calculated in many different text-to-speech training frameworks. This

saves us a great amount of time from calculating features redundantly such as valence. For the purposes of our RL-based training pipeline, we require a dataset diverse enough that produces strong, medium, and weak reward signals for the model to learn from. Although the audio is largely monotone for many of the audio clips, there are bursts of variance that we felt were sufficient enough for learning different reward signals. The dataset is an important aspect of the training pipeline, and further experimentation with different datasets will be important in gauging the efficacy of the different RL techniques.

Chapter 4

Methods and Strategies

4.1 Problem Statement

We address two scenarios in this paper.

In the first scenario, we assume that we have a pre-trained text-to-speech model and emotion predictor model to supply reward signals. In addition, we have access to a dataset of text prompts and associated audio. In this scenario the architecture of the base model is abstracted, and we are only concerned with the final reward signals produced. We must find a way to utilize the produced reward signals to tune the text-to-speech model to produce more angry outputs.

In the second scenario, we assume that we have a pre-trained diffusion-based text-to-speech model and emotion predictor model to supply reward signals. In addition, we have access to a dataset of text prompts and associated audio. In this scenario, the architecture is of importance for it allows us to use multiple training steps per audio-text pair.

4.2 Reward Feedback and Quality

Like in many reinforcement learning applications, the “quality” of the reward largely determines the efficacy of the training process. Factors such as the sparsity of the reward are handled by using a dense probabilistic distribution. Specifically, the rewards provided by our emotion predictor are well distributed and not condensed into the tails of the distribution (near 0 and 1). This allows for a more stable learning trajectory. Furthermore, selecting a model that outputs consistent rewards is another key factor. Audio samples should be graded consistently for anger so that the model can tune itself to learn the overall objective. Through qualitative experimentation, we determined that the Superb emotion predictor was one of the better models capable of both correctly identifying angry audio samples as well as determining the relative degree of anger in the audio samples. The main issue with the reward architecture is that it was susceptible to potential “reward hacking,” as described in Skalse et al. [20], a phenomenon where the rewards signal the model to optimize for an

objective that is unaligned with the true objective. In our case, our model is able to discern the degree of anger in an audio sample. However, it is unable to determine the fidelity of the original speech. This allows the model to be susceptible to learning noise. Since the model predictor must output a probability distribution across the range of emotions that sums to 1, random noise signals can be interpreted as anger. Due to this deficiency in the reward, we had to be careful during the training process, especially when using techniques such as multistep PPO that leveraged reward signals to a higher degree, to train the model in a stable region and avoid producing a decoder that outputted noise.

4.3 Reward Weighted Regression

To get a sense of a baseline from a simpler method, we begin by applying Reward Weighted Regression (RWR).

RWR is a simple, intuitive yet efficient technique to train models based on rewards. While there are many variants of RWR, the standard one utilized in this paper follows Black et al. [3]:

$$w = \frac{1}{Z} \exp(\beta r(x, c))$$

Where w is the weight utilized for training, Z is a scaling factor, β is the inverse temperature, and $r(x, c)$ is the reward given by the emotion predictor/feedback model.

In this equation, the reward feedback model is exponentiated and multiplied by a scaling constant. This value then scales the loss accordingly and provides feedback for the model to learn from. In this case, we use the standard losses from the underlying model and scale each of them by the exponentiated reward. The exponentiation follows convention to avoid negative rewards in the general setting but in the particular choice of reward function that we have, the reward is a probability and therefore positive anyways. The high reward results are considered more "important" in the gradient calculation.

We use RWR in two scenarios: our diffusion-based Grad-TTS model and the transformer-based Fastpitch model. In both scenarios, only a single step is utilized for each training example. Grad-TTS by default sums a diffusion loss, prior loss, and duration loss. These are all weighted by reward at the last step that maintains the batch dimension and then summed and scaled as usual.

The typical style of advantage for RWR uses a log probability of an action under the policy (Peng et al. [15]). This results in the loss for RWR as $-\log \pi(a|s) \exp(\frac{1}{\beta} R)$, where β is a hyperparameter and reward is R . Under this setting, the diffusion model underlying Grad-TTS is viewed as taking actions by selecting the next x_i going across diffusion steps x_t to x_1 .

However, Grad-TTS in practice is, by default, implemented as a deterministic diffusion model, using fixed dynamics to simulate the underlying stochastic Brownian motion-based model, for the authors say that this results in slightly better performance [16]. In this

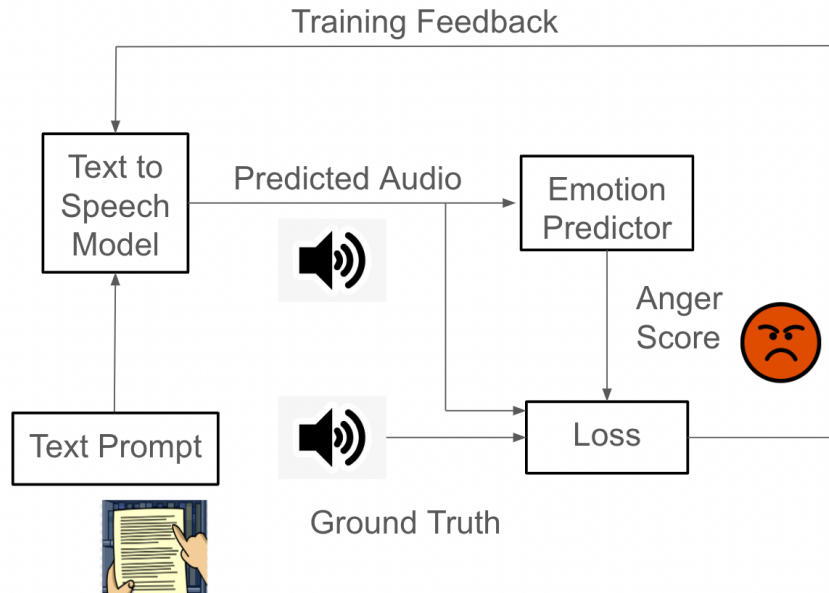


Figure 4.1: RWR Architecture

case, we view the loss to be in the form $-(a - \pi(s))^2 \times \exp(\frac{1}{\beta}R)$. The action is still seen as the actual output of the model but the loss takes a form closer to what is expected in typical supervised learning. There is an underlying ground truth that we are attempting to clone that comes from the original task goal of reconstructing the audio given the dataset. Therefore, this explains the usage in our case where we use the default implementation and simply scale each component of the loss associated with a particular sample by a function of the reward.

We utilize RWR at a similar capacity when it comes to implementing it within Fastpitch. Fastpitch has different losses for pitch, energy, and duration. Given that, we thought to maximize the capabilities of RWR and apply it to more focused subsets of the pitch, energy, and duration losses. Given how the angry emotion can be communicated with changes to only pitch and perhaps energy, we looked to see if RWR could take advantage of training a more focused objective.

4.4 Ensemble Based Learning for RWR

In many different applications for machine learning, oftentimes the ability to diversify across various tasks and sources of data leads to enhanced performance and robustness as discussed in Ganaie et al. [7]. With this knowledge in mind, we decided to add diversity to our rewards in the form of adding another emotion predictor model, the Dunn predictor (DunnBC22 [6]), to provide reward feedback. The Dunn predictor was a candidate to be the original emotion

predictor. However, when performing qualitative testing on its ability to discern angry speech and the degree of the anger, the Dunn predictor performed worse than the Superb predictor.

Using the average reward from our original emotion predictor and this new emotion predictor, we tuned a new model using the RWR mechanism. Over the course of many reward queries, we tracked the performance of the average score from the emotion predictors across the validation set as well as the individual scores of the emotion predictors. Our goal was to see how training with multiple predictors impacted scores across individual predictors and whether adding a qualitatively worse emotion predictor detracted from the overall training process or it made it more robust.

4.5 Markov Decision Processes (MDP) and Diffusion Models

A Markov Decision Process is a representation utilized in Reinforcement Learning to model problems related to decision-making. Here are some of the key components of an MDP:

- S: States
- A: Actions
- P ($s_{t+1} \mid s_t, a_t$): Probabilities
- R(s): Rewards

MDPs are utilized to find the optimal strategy for an actor to engage with their environment (what are the best actions to reach the states that give the best cumulative rewards).

Through this lens, Black et al. [3] write that conditional diffusion probabilistic models represent a distribution $p(x_0|c)$ over a dataset of samples x_0 and corresponding contexts c . Specifically, they write that these distribution models are "the reverse of a Markovian forward process $q(x_t|x_{t-1})$, which iteratively adds noise to the data. Reversing the forward process can be accomplished by training a neural network $\mu_\theta(x_t, c, t)$ with the following objective:

$$L_{DDPM}(\theta) = \mathbb{E}_{(x_0, c) \sim p(x_0, c), t \sim \mathcal{U}\{0, T\}, x_t \sim q(x_t | x_0)} \|\tilde{\mu}(x_0, t) - \mu_\theta(x_t, c, t)\|^2$$

where $\tilde{\mu}$ is the posterior mean of the forward process, a weighted average of x_0 and x_t . This objective is justified as maximizing a variational lower bound on the log-likelihood of the data" [3].

Extending off this logic, starting off with a random state and following the reverse diffusion process produces trajectories that allow us to apply MDP-styled RL techniques.

Denoising Diffusion Policy Optimization (DDPO) is a technique developed by Black et al. [3] that utilizes the aforementioned framework of viewing the denoising process as an MDP,

and through this lens optimizes the RL objective using policy gradient estimates. Black et al. [3] created two variants of DDPO.

- DDPO with REINFORCE: Uses denoising trajectories from current parameters
- DDPO with Importance Sampling: Uses denoising trajectories from old parameters

DDPO with Importance Sampling allows for multiple steps of optimization as compared to REINFORCE which only allows for a single step. Therefore, if leveraged correctly DDPO with Importance Sampling can be incredibly powerful. Hence we aimed to use the importance sampling version of DDPO in our experiments.

4.6 PPO/DDPO

The PPO paper (Schulman et al. [18]) presents a loss

$$\mathcal{L}_{PPO} = \mathbb{E}[\min(r_t A_t, \text{clip}(r_t, 1 - \epsilon, 1 + \epsilon) A_t)]$$

A_t is the advantage and r_t refers to the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}$. a_t and s_t are action and state respectively. θ_0 refers to the old policy and the θ is the current policy. The purpose of the ϵ is to clip the probability ratio, so as to ensure that the movement of the ratio outside of that 2ϵ range is not incentivized.

The benefit of PPO is that it provides us with better sample efficiency than a pure policy gradient by being more robust to changes in the policy because of the term resembling importance sampling. In the case of DDPO, this is particularly helpful as we can take multiple updates within a batch of data by merely updating the ratio as mentioned in Black et al. [3]. Further, the number of gradient calculations accumulated becomes a hyperparameter that can be tuned for increased confidence in the gradient. If the number of accumulation steps is equivalent to the number of total diffusion steps in the batch, then this is just like the policy gradient.

However, with PPO there are potential challenges with gradients diverging. In particular, each step of diffusion is viewed as selecting an action. So the probabilities are computed as $p(x_{t-1}|x_t)$. However, the new probabilities of the given base action can be vastly different from the old actions very quickly.

Additionally, we use the likelihood from continuous normal probability density function as a proxy for probability which can cause scaling issues. The probabilities vary greatly and potentially the gradients stacking over many accumulation steps in the same direction can lead to divergence quickly. As a result, we adjust for these issues by clamping the probability based on a hyperparameter for how far from the center of the pdf function we want to be. However, this has the effect of making the base log probabilities also potentially cause issues, so we apply the same technique to ensure that there is a further limitation imposed on either probability for numerical stability, instead of just using the ratio clipping from the original loss.

4.7 PPO/DDPO with Learning without Forgetting

In PPO there is no longer any connection to the original loss as it goes from being a supervised learning problem to purely optimizing for an angry score. This results in the risk of over-optimization and, occasionally, decreasing audio quality. Additionally, PPO is fairly inflexible with hyperparameter choices which can cause divergence and over-optimization issues to arise frequently, making hyperparameter search difficult.

Due to these issues of audio degradation in the standard model, we changed the training style to mimic the Learning without Forgetting concept from supervised learning even though that is typically used for architecture that has some shared model parameters with task-specific heads. In the adaptation for this architecture, we focus on the concept of optimizing the parameters with the old loss function gradients added to the new RL gradients. As a result, this will restrain the parameter movement by encouraging similar outputs as were produced prior to beginning the RL training. The simplest way to express this is to use:

$$\mathcal{L}_{total} = \mathcal{L}_{PPO} + \beta\mathcal{L}_{old}$$

For hyperparameter β which depends on the magnitudes of the two losses. This formulation follows fairly directly from what is expressed in Li and Hoiem [13], but adapted for our setting.

The computation of \mathcal{L}_{PPO} can be accumulated together and then the old loss is applied or if smaller numbers of accumulation steps are used, then that is handled internally first, and the step corresponding to \mathcal{L}_{old} is taken separately after the batch has completed PPO steps. We used that structure with a lower number of accumulation steps, so there are multiple steps taken based on \mathcal{L}_{PPO} for each time a step is taken for \mathcal{L}_{old} . The hyperparameter would need to be adjusted to take into account this relationship with accumulation steps. We still start with a pretrained checkpoint so as to begin with the base capability, but then proceed with this mixed loss.

One important detail is, considering the architecture for prompt-image alignment Black et al. [3] use, feedback exists such that the image may be described by another model before that description is fed into the reward model. In our structure, there is no natural sense for another model that can be an intermediary between the model audio and the reward function. That intermediary would function as a check for basic quality because the reward could be gamed by some trivial output otherwise. As a result of the domain and architecture we use, the over-optimization issue is more critical and therefore we present this technique to preserve audio quality without the use of another model.

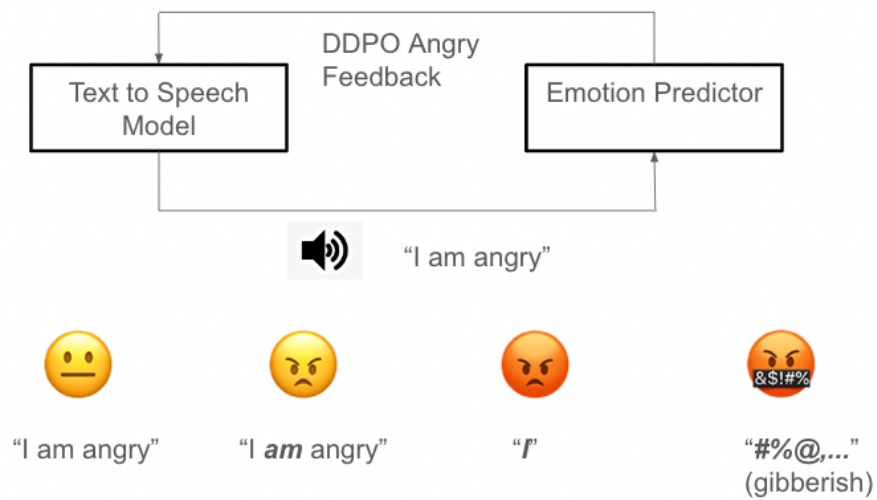


Figure 4.2: Over-optimization from DDPO can occur if the reward signals only anger and not audio fidelity or fluency. The text-to-speech model can over time devolve from these reward signals into producing gibberish.

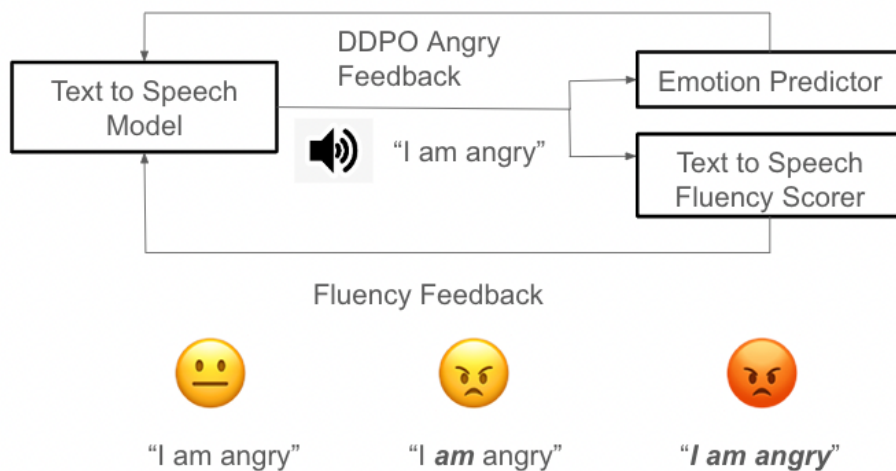


Figure 4.3: DDPO combined with Learning without Forgetting trains on both the angry signals as well as the original text-to-speech fluency objective allowing for more stable learning.

Chapter 5

Experiments and Analysis of Results

In this section we go over the experiments we conducted, the results of these experiments, as well as the analysis of these results.

The purpose of RLAIIF is to tackle complex and abstract objectives. However, this can make it more complicated to evaluate. We evaluate our experiments on two fronts:

- Quantitative Analysis, where we grade the performance of the text-to-speech model using the emotion predictor model over the course of many reward queries on a set of validation prompts.
- Qualitative Analysis of the audio generated

Combining these analyses we hope to provide a robust assessment of our experiments.

5.1 Experiments

RWR

For RWR we ran experiments on both the Grad-TTS model and the Fastpitch Model. Examples produced with this style of training are presented in the Results section.

Grad-TTS

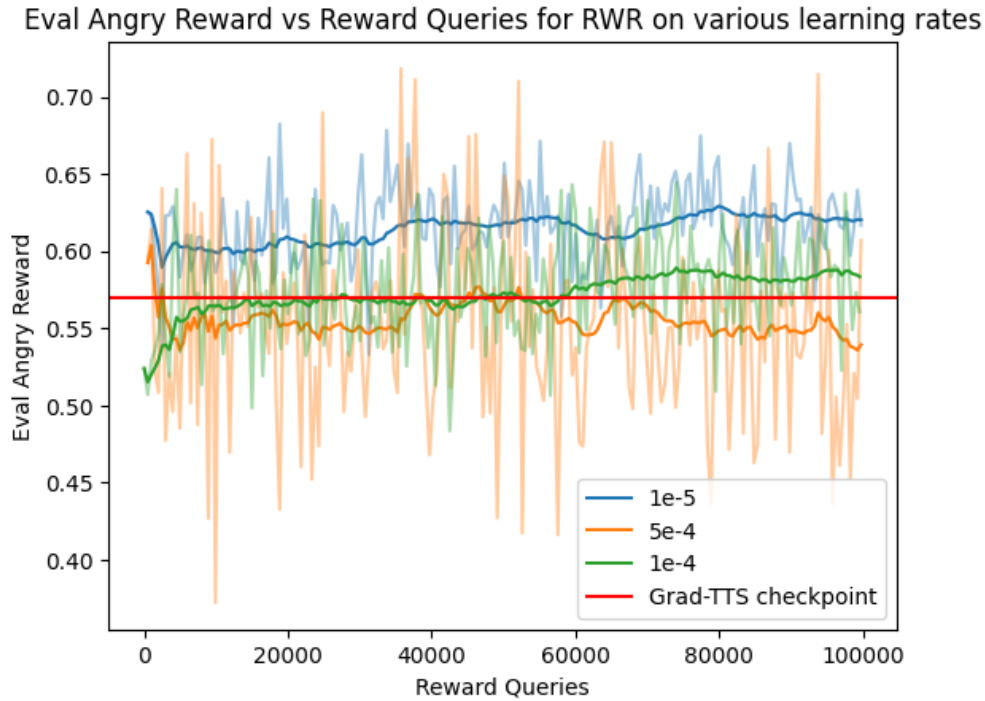


Figure 5.1: Performance for Reward Weighted Regression on various Learning Rates for Grad-TTS

We run this algorithm starting with the pretrained checkpoint from Grad-TTS. In our ablation study, (Figure 5.1), we consider the learning rate selection. Across all choices of learning rate, we can see that this technique is fairly unstable. As mentioned in Black et al. [3], this is a loose approximation of the objective which may contribute to this learning slowly and unstably. Additionally, the overall loss would still be in the direction of original training objective, although scaled to place more importance on particular samples. The increase in "anger" reward is a side effect to the standard loss minimization through weighted contributions from each sample. The choice of learning rate $1e - 5$ produces a significant change in the evaluation curve. This is likely due to the pattern that we see where there are large changes early on, which could explain why the smaller learning rate does better. The underlying data, prior to smoothing, is quite unstable and a smaller learning rate would be preferred in this case as we see in the ablation graph (Figure 5.1).

Nonetheless, there is a significant improvement shown in the data. The baseline of the pre-trained Grad-TTS checkpoint results in just a 0.5695 reward, while the RWR reached 0.62 at peak. The RWR training was quite slow in terms of real-time, with these runs taking

one week in our testing setup. Additionally, RWR took a high number of reward queries to show results as shown in Figure 5.1.

FastPitch

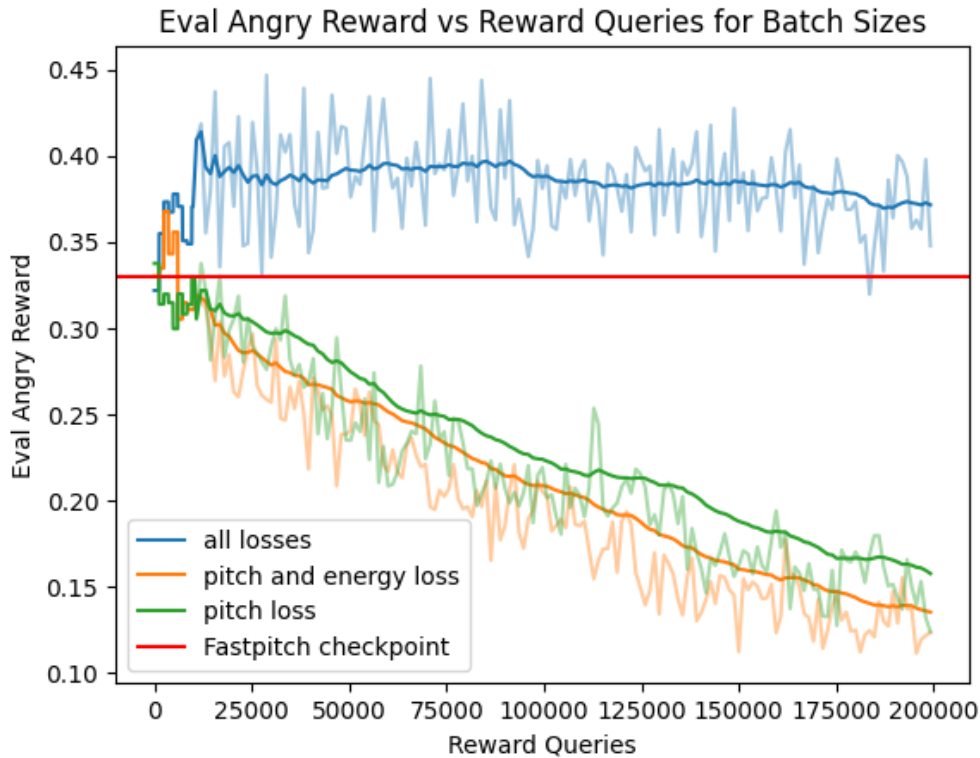


Figure 5.2: RWR on Fastpitch using different sets of losses

Since our goal in using the Fastpitch Model was to see if RWR could take advantage of training on condensed objectives, we experimented with 3 different objectives. The first objective included training on all losses (pitch, energy, and duration). The second objective trained on (pitch and energy), and the final objective trained on only (pitch). Since some of the key characteristics of an angry emotion lie in attributes such as pitch and energy, we thought that training on these losses would prove to be useful.

The results, however, show otherwise as shown in 5.2. Specifically, training on condensed objectives such as only pitch or pitch and energy displayed worse performance with the emotion predictor. Meanwhile, training with all the losses although not amazing proved to have some increased performance with the emotion predictor. RWR training on all the Fastpitch losses was similar to using RWR on Grad-TTS. Both were stable when it came to training; however, they both only provided small gains in performance.

These experiments showed that a balanced training approach is beneficial to getting good results, something echoed in later experiments. Hyperfocusing on certain attributes confuses the emotion predictor model and in fact degrades the overall performance of the text-to-speech model.

Faspitch due to its innate architecture was much better at processing reward queries and comparatively much faster than Grad-TTS, being able to process twice the amount of queries in a similar timespan.

Ensemble Based RWR with Grad-TTS

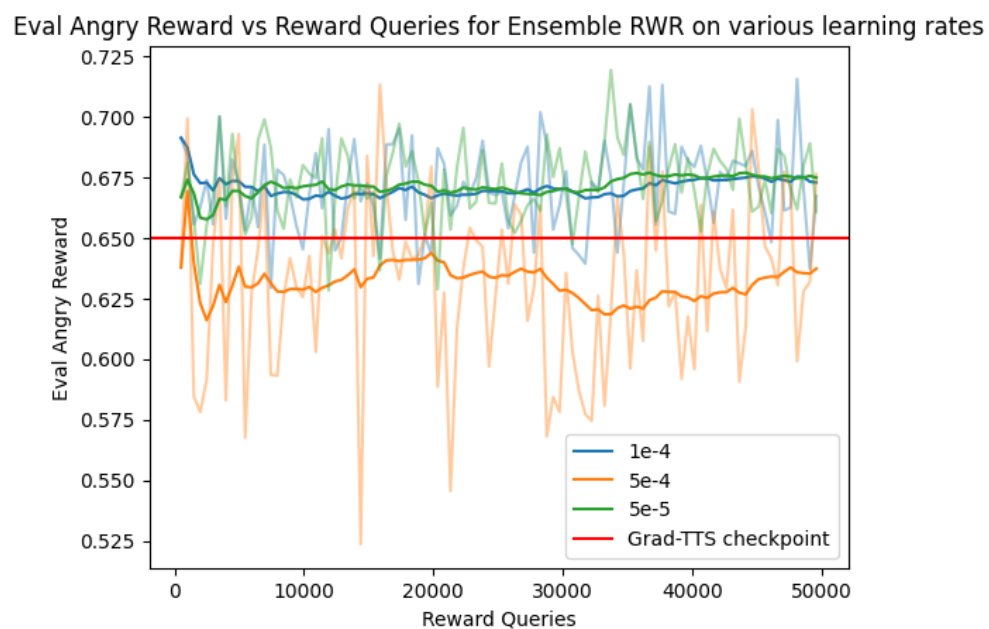


Figure 5.3: Ensemble Eval Performance with various Learning Rates

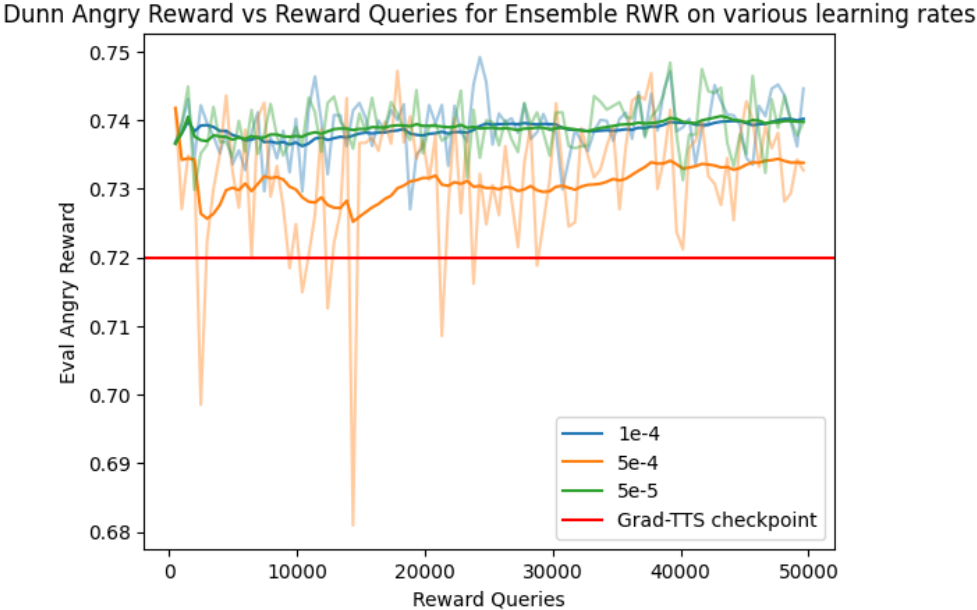


Figure 5.4: Ensemble Dunn Predictor Performance with various Learning Rates

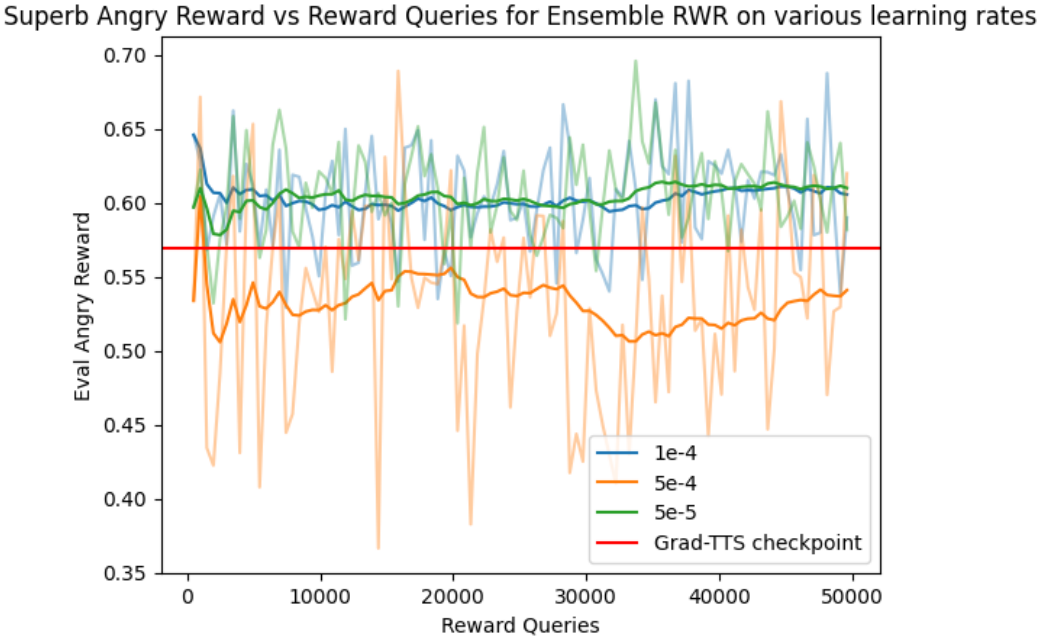


Figure 5.5: Ensemble Superb Predictor Performance with various Learning Rates

We conducted RWR experiments on the Grad-TTS model using the average feedback from two separate emotion predictor models.

Conducting these various experiments demonstrated the significance of quality when using emotion predictors. The Dunn predictor’s score rarely changes over the course of training as shown in Figure 5.4. Although the trajectory of the evaluation performance for the Dunn predictor steadily increases, the scale of this increase pales in magnitude to the Superb predictor. This discrepancy in magnitude essentially means that the reward provided by the Dunn predictor serves as extra noise for training on the Superb predictor as shown in Figure 5.5. The noise interferes with training and simply hampers the overall performance as shown in Figure 5.3. Regardless of the learning rate, we found these findings to be consistent suggesting that it is crucial when using RLAIIF in the specific use case to find emotion predictor models that are consistent and highly accurate. The lack of such qualities makes it difficult to isolate reward signals and train effectively.

DDPO

DDPO using PPO

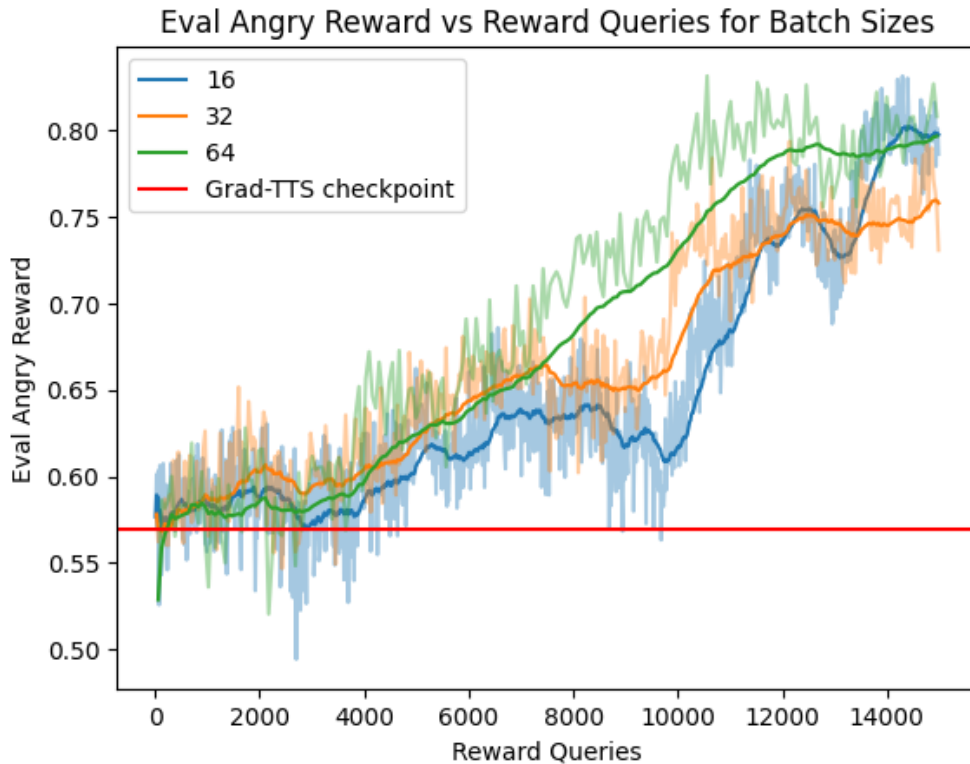


Figure 5.6: PPO Batch size ablation

In our ablations for DDPO using PPO, we consider the effect of batch size. For the implementation of DDPO using PPO, we use the normalized reward for the advantages, and batch size would significantly impact that. Additionally, when holding accumulation steps constant, the batch size impacts the number of gradient steps taken since the total diffusion steps is batch size \times the number of diffusion steps per sample. This impacts the drift from the base probabilities collected prior to any of the gradient steps and therefore the importance sampling-based coefficient in the gradient computed for each diffusion step.

Finally, we believed a larger batch would allow for more consistent gradient steps, as usual. However, as shown in Figure 5.6, the result of the ablation study revealed the smallest batch seemed to perform quite well. This could be due to the large accumulation size relative to the total number of steps or the fact that the smaller batch size introduces more variance and that happened to be favorable here. Among the 32 and 64 sizes, the 64 performed better as expected. Much smaller batch sizes ended up being very poor performing and causing divergence issues.

These runs took much less time to provide stronger results as compared to RWR, taking less than 2 days for the 15k reward queries shown here with the same compute setup and frequency of evaluation.

Examples produced with this style of training are presented in the Results section.

DDPO using PPO with Learning Without Forgetting

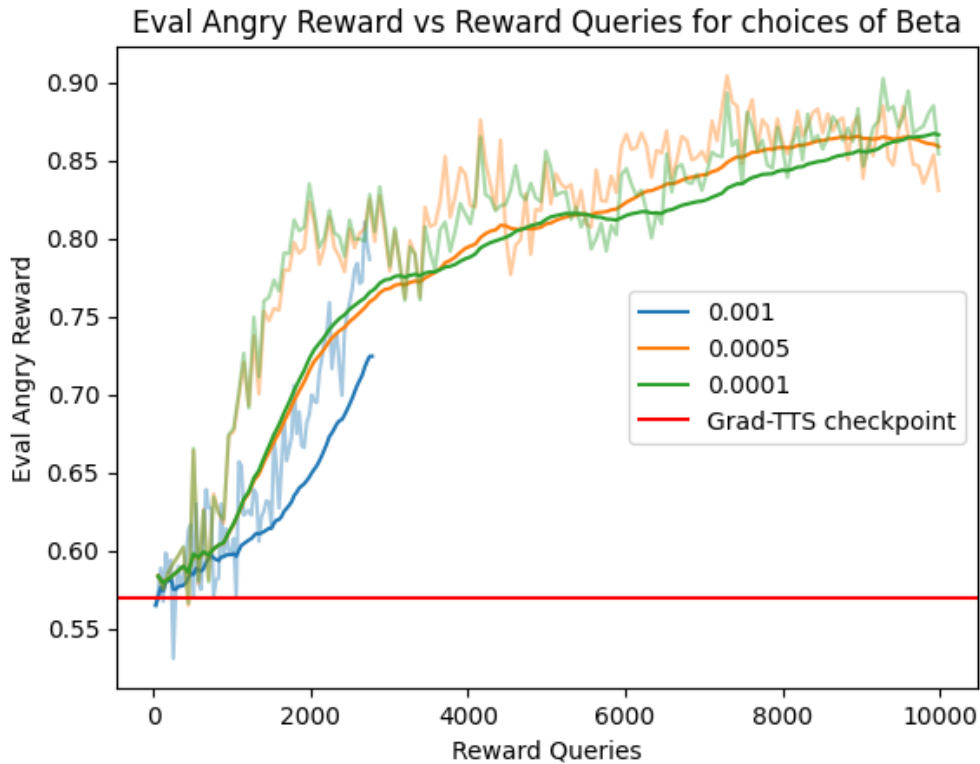


Figure 5.7: PPO-LWF Beta ablation

As mentioned in the methods section, Learning without Forgetting can help stabilize the training process by combining the new loss function with a weighted influence (β) of the old loss function. Specifically, we train on both the anger loss generated from the emotion predictor as well as the original fluency text-to-speech task.

To test out the choice of hyperparameter, β , we performed the ablation study shown in Figure 5.7. The choice of $\beta = 0.001$ did diverge relatively early, likely due to the fact that there are now more gradient steps taken and that $\beta = 0.001$ was a large choice of beta given the magnitude of our original loss. However, the mixture of the original loss did prove to be quite successful here, first of all for preserving audio quality for the samples above, but also purely looking at the evaluation curve. There are fewer samples used to reach convergence, as the evaluation score starts to flatten as early as four thousand reward queries with the final scores being near 0.85 as shown in the figure.

So, Learning without Forgetting ends up being faster at reaching a higher score when compared to PPO, though it does require the ground truth data for the underlying model. These are done with the same parameters (other than introducing LWF and its corresponding

β) as the 64-batch PPO experiment. While we initially expected introducing LWF to reduce the quality of training, since we are giving up the ability to produce noise and generate higher evaluation scores to some degree, it turns out that this actually improves the scores in our setup. A reason could be that producing the higher quality audio in fact helped the emotion model more accurately discern emotion since the true audio was the input to the emotion model. It could be that the noise was not always beneficial in terms of the reward, but rather that without the LWF style training there was no good feedback to directly penalize the impact of the audio quality loss as it could be mixed in with other changes more directly tied to improved reward.

Further examples produced with this style of training are presented in the Results section.

5.2 Results

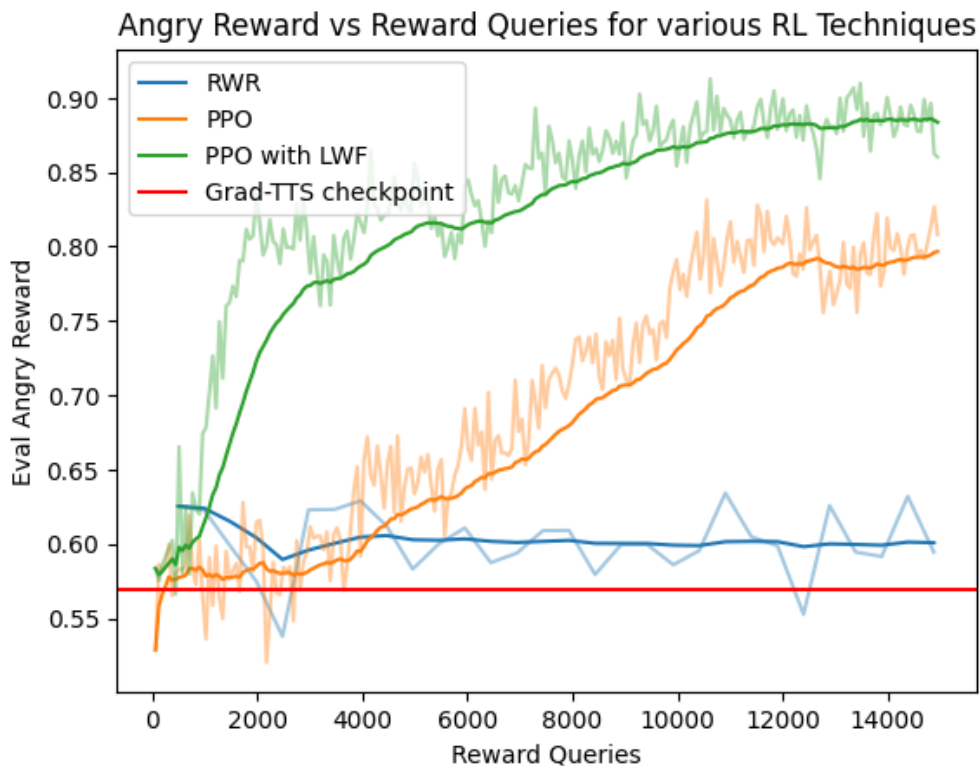


Figure 5.8: Angry Reward vs Reward Queries for various RL Techniques

To evaluate the effectiveness of different RL approaches, we compare their qualitative and quantitative performance metrics. Due to the large number of ablation studies and experiments conducted, we present results for the top-performing variants within each technique.

These variants are identified based on quantitative performance measured by the emotion detector. Specifically, we use the Grad-TTS checkpoint as a raw baseline, RWR on Grad-TTS with learning rate 1e-5, DDPO using PPO on Grad-TTS with batch size 64, DDPO using PPO with LWF with β value 0.0001. Future references for example to RWR in this section will refer to the RWR on Grad-TTS with learning rate 1e-5.

As shown in Figure 5.8, the Learning without Forgetting style of training performed the best in terms of evaluation score across all the methods shown in this plot, although all showed improvement from the baseline. That style of training was also far more efficient in terms of reward queries, as the curve is also far steeper than the others. This does come with the trade-off that the underlying ground truth losses are used, while PPO does not need any ground truth information. RWR training showed itself to be slow in addition to being weak at producing good quantitative results. Furthermore, even when given the chance to train on more condensed training objectives using Fastpitch losses, RWR was not able to provide any meaningful improvements in performance. The main positive about RWR was the ease of implementation and lack of divergence issues, while the other methods suffered from the risk of diverging at points in training. Therefore, it can be reasoned that the main expense of using one of the PPO-based approaches would come in the hyperparameter search and implementation.

Between PPO and PPO with LWF, it is apparent that the latter leads to an improved evaluation curve through the use of the underlying model’s training data. Additionally, the LWF style is far more robust as we get a regularizing effect to ensure that we are less likely to lose original model capabilities in pursuit of a higher reward, as can happen more frequently in the pure PPO case. Additionally, the LWF loss also has some connection to the concept of using a Kullback-Leibler (KL) divergence term to penalize shifts from the underlying model policy in the loss for PPO as expressed by Schulman et al. The reasoning for that is also to ensure that the shift is not too great from the original policy, which is a similar goal to what we aim for in adding the LWF loss.

Table 5.1: Qualitative Comparison of Audio Samples Across Different RL Techniques with Hyperlinks (Click to Listen). Raw URL to the audio will be posted in the Appendix.

	Grad-TTS checkpoint	RWR	PPO	PPO-LWF
Example 1	Hyperlink	Hyperlink	Hyperlink	Hyperlink
Example 2	Hyperlink	Hyperlink	Hyperlink	Hyperlink
Example 3	Hyperlink	Hyperlink	Hyperlink	Hyperlink
Example 4	Hyperlink	Hyperlink	Hyperlink	Hyperlink
Example 5	Hyperlink	Hyperlink	Hyperlink	Hyperlink

Table 5.1 showcases some example audios across the different RL techniques. Below are visual Waveplots for the prompt "that is very unfair" across the different models. Although amplitude is not the only factor in determining anger, it is often strongly associated and can be an effective indication of anger.

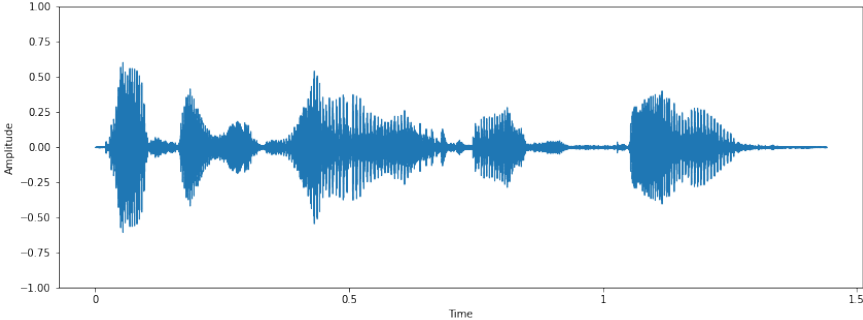


Figure 5.9: Waveplot for Checkpoint on prompt "that is very unfair"

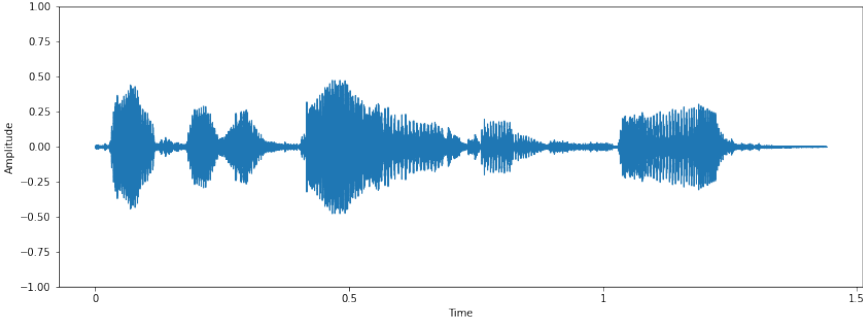


Figure 5.10: Waveplot for RWR on prompt "that is very unfair"

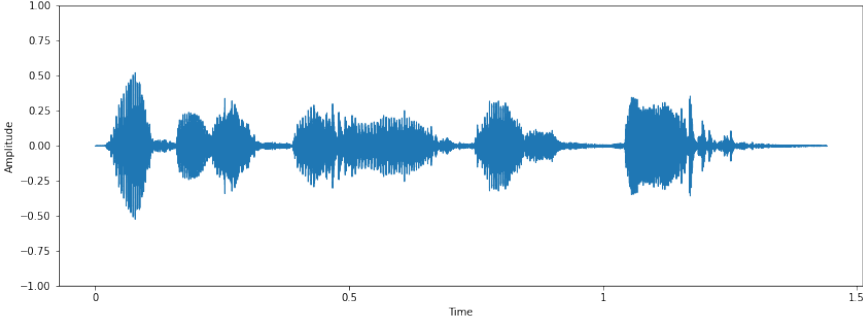


Figure 5.11: Waveplot for PPO on prompt "that is very unfair"

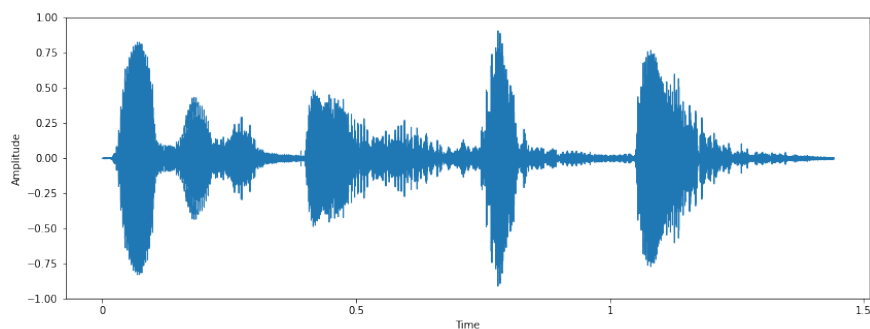


Figure 5.12: Waveplot for PPO with LWF on prompt "that is very unfair"

PPO with LWF, as seen in Figure 5.12, enunciated each of the words strongest compared to the rest of the models. RWR and PPO as seen in Figures 5.10 and 5.11 respectively enunciated some words stronger than the checkpoint model but not near the degree of PPO with LWF. We found this pattern to be consistent among many examples as shown in the Appendix.

From our qualitative study of the various RL techniques, it was obvious to us that PPO combined with Learning without Forgetting performed best. Particularly as seen in Examples 1, 2, and 3, PPO with LWF spoke very aggressively and learned to at times "bark" the input text. Other techniques such as RWR were much more subtle and took a more passive-aggressive approach towards expressing anger. PPO had a similar qualitative performance to RWR with the additional issue of occasional noise. In conclusion, PPO with LWF performed the best overall both qualitatively and quantitatively in presenting anger.

Chapter 6

Discussion, Limitations, and Future Work

Although our experiments were very successful in many capacities, we see the opportunity to improve the performance and the capabilities of our system. Some current limitations with our system are the issues related to divergence. Specifically, with certain training methods, our system becomes more susceptible to reward hacking. While we employed certain techniques to mitigate the effects of reward hacking, the system is fundamentally susceptible. The only way to improve upon this is to transform our reward objective to become more aligned with our true goal; therefore, the system will be unable to hack towards an unaligned objective. One way we can better align the reward is to introduce a second reward in addition to the angry score that computes the speech embedding similarity between the output audio and the intended text. Adding this reward should theoretically act as a regularizing term that ensures that the model does not simply learn to output noise that is considered “angry.” This second reward is similar to the idea of Learning without Forgetting, and in many cases, both of these techniques strive to achieve the same goal: ensure that the model produces high-fidelity audio representative of the input text.

In addition to using a second reward, we feel that a natural extension of the system’s capabilities would be to allow for the model to train upon multiple emotions. Specifically, the goal would be to emulate some of the developments found in Liu, Sisman, and Li [14] and allow for emotion embedding support in the system. Specifically, an emotional embedding can be added between the encoder and the decoder to allow for training and ultimate usage on a wide array of emotions. While introducing speech embeddings can raise a variety of challenges during training and testing, the model’s ability to learn singular emotions sparks confidence in training on multiple emotions. Additionally, we could explore styles of training and architectures that can support emotions that do not exist in the training data, so for instance making the model have the ability to express any inputted tone, even if never trained explicitly for that tone. This could be done with the use of LLM embeddings for the emotion.

Another area that has potential for improvement is the evaluation process. Due to

resource limitations, our method of evaluation was using our intended emotion predictor to score the model on an external validation set. It would be helpful to supplement this evaluation with Mean Opinion Scores regarding the anger of certain audio clips. Since RLAIIF is ultimately targeted at being applied to diverse and abstract reward targets, detailed human evaluation would be important in quantifying the performance of the overall system. Furthermore, our experiments were only targeted toward using Reward Weighted Regression and PPO. However, we can explore other techniques like REINFORCE still.

Bibliography

- [1] Dmytro Babko. *Speech Emotion Recognition (en)*. 2021. URL: <https://www.kaggle.com/datasets/dmitrybabko/speech-emotion-recognition-en>.
- [2] Alexei Baevski et al. *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. 2020. arXiv: 2006.11477 [cs.CL].
- [3] Kevin Black et al. *Training Diffusion Models with Reinforcement Learning*. 2023. arXiv: 2305.13301 [cs.LG].
- [4] Carlos Busso et al. *Iemocap: Interactive Emotional Dyadic Motion Capture Database*. 2008.
- [5] Mingyi Chen et al. “3-D Convolutional Recurrent Neural Networks With Attention Model for Speech Emotion Recognition”. In: *IEEE Signal Processing Letters* 25.10 (2018), pp. 1440–1444. DOI: 10.1109/LSP.2018.2860246.
- [6] DunnBC22. *wav2vec2-base-Speech_Emotion_Recognition*. 2023. URL: https://huggingface.co/DunnBC22/wav2vec2-base-Speech_Emotion_Recognition.
- [7] M.A. Ganaie et al. “Ensemble deep learning: A review”. In: *Engineering Applications of Artificial Intelligence* 115 (Oct. 2022), p. 105151. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2022.105151. URL: <http://dx.doi.org/10.1016/j.engappai.2022.105151>.
- [8] Yiwei Guo et al. *EmoDiff: Intensity Controllable Emotional Text-to-Speech with Soft-Label Guidance*. 2023. arXiv: 2211.09496 [eess.AS].
- [9] Keith Ito and Linda Johnson. *The LJ Speech Dataset*. <https://keithito.com/LJ-Speech-Dataset/>. 2017.
- [10] Jaehyeon Kim et al. *Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search*. 2020. arXiv: 2005.11129 [eess.AS].
- [11] Adrian Łańcucki. *FastPitch: Parallel Text-to-speech with Pitch Prediction*. 2021. arXiv: 2006.06873 [eess.AS].
- [12] Harrison Lee et al. *RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback*. 2023. arXiv: 2309.00267 [cs.CL].
- [13] Zhizhong Li and Derek Hoiem. *Learning without Forgetting*. 2017. arXiv: 1606.09282 [cs.CV].

- [14] Rui Liu, Berrak Sisman, and Haizhou Li. *Reinforcement Learning for Emotional Text-to-Speech Synthesis with Improved Emotion Discriminability*. 2021. arXiv: 2104.01408 [cs.CL].
- [15] Xue Bin Peng et al. *Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning*. 2019. arXiv: 1910.00177 [cs.LG].
- [16] Vadim Popov et al. *Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech*. 2021. arXiv: 2105.06337 [cs.LG].
- [17] Yi Ren et al. *FastSpeech: Fast, Robust and Controllable Text to Speech*. 2019. arXiv: 1905.09263 [cs.CL].
- [18] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].
- [19] Jonathan Shen et al. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. 2018. arXiv: 1712.05884 [cs.CL].
- [20] Joar Skalse et al. *Defining and Characterizing Reward Hacking*. 2022. arXiv: 2209.13085 [cs.LG].
- [21] Superb. *wav2vec2-base-superb-er*. 2021. URL: <https://huggingface.co/superb/wav2vec2-base-superb-er>.
- [22] Binghai Wang et al. *Secrets of RLHF in Large Language Models Part II: Reward Modeling*. 2024. arXiv: 2401.06080 [cs.AI].
- [23] Shu-wen Yang et al. *SUPERB: Speech processing Universal PERFORMANCE Benchmark*. 2021. arXiv: 2105.01051 [cs.CL].

Chapter 7

Appendix

7.1 Table of Audio URLs and Waveplots

Table 7.1: URLs and Waveplots 5.1

	Grad-TTS checkpoint	RWR	PPO	PPO-LWF
Example 1	1a	1b	1c	1d
Example 2	2a	2b	2c	2d
Example 3	3a	3b	3c	3d
Example 4	4a	4b	4c	4d
Example 5	5a	5b	5c	5d

1: "I am learning to be angry"

1a

URL: <https://drive.google.com/file/d/17mOqlPwapNC0iRrSZcfaKFIS3P2eNC7J/view?usp=sharing>

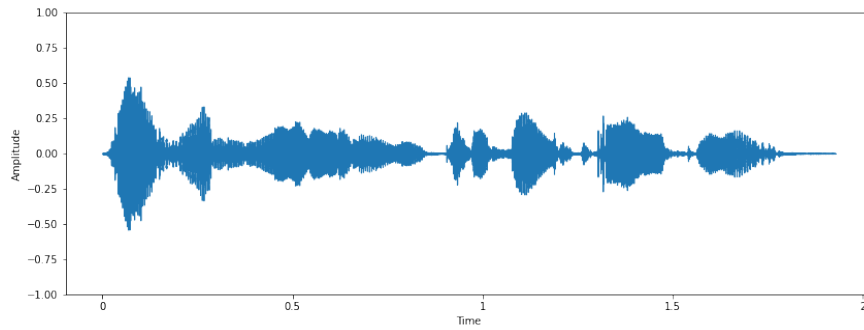


Figure 7.1: Waveplot for 1a

1b

URL: <https://drive.google.com/file/d/11iy9kEW8Aj0PtQ4JyMehhPPHkselw09v/view?usp=sharing>

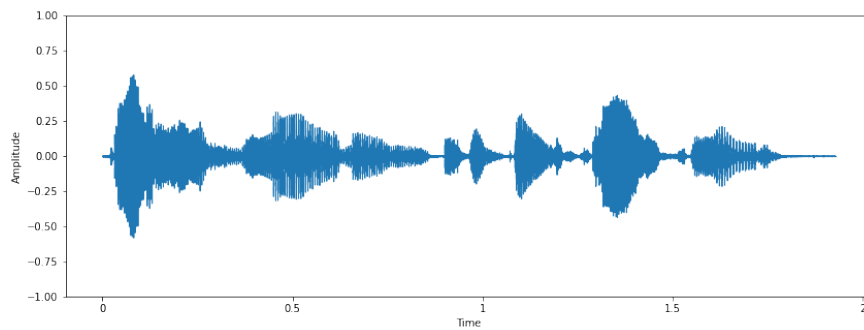


Figure 7.2: Waveplot for 1b

1c

URL: https://drive.google.com/file/d/1lz8dNOyuWuQl-Mz4CpZbUh_dSLcFKs4J/view?usp=sharing

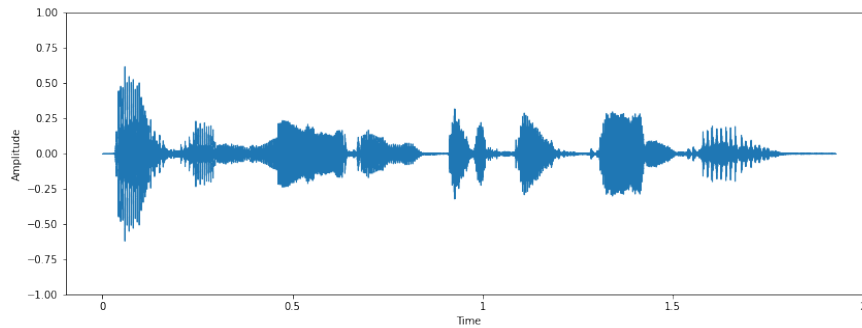


Figure 7.3: Waveplot for 1c

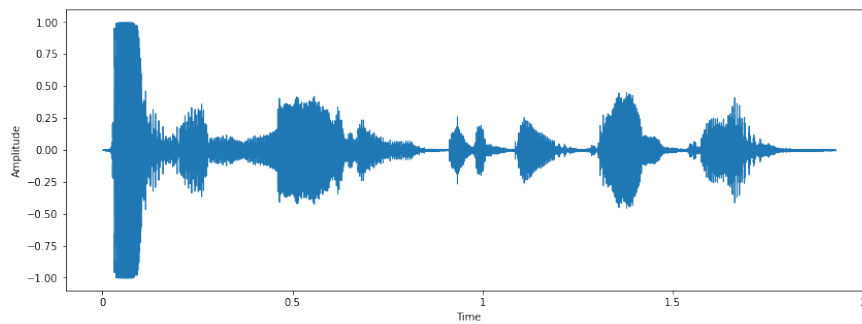
1dURL: <https://drive.google.com/file/d/1PzMcR8JgsJsLHxNI9ciZ-SbE-2NXMrD2/view?usp=sharing>

Figure 7.4: Waveplot for 1d

2: "I am very angry"**2a**URL: https://drive.google.com/file/d/1Eg0ZBzpd0_bPGoR7mZA5f0mYr7JYtXf/view?usp=sharing

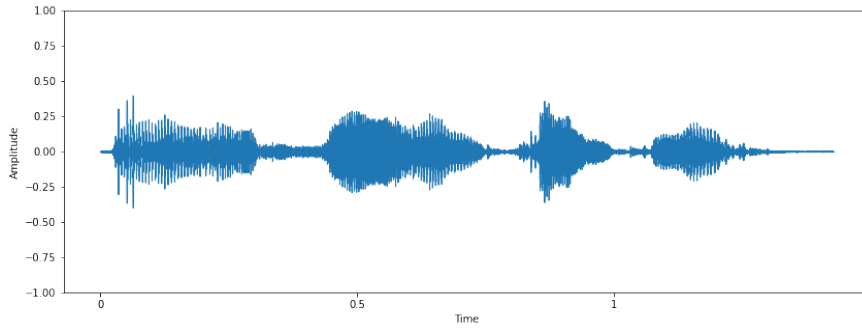


Figure 7.5: Waveplot for 2a

2b

URL: <https://drive.google.com/file/d/1wDHaiA5WkI0oVSf3JFX8YJIOxouVSsjy/view?usp=sharing>

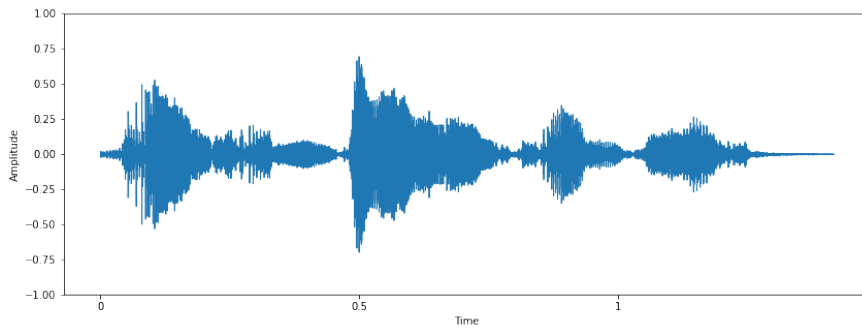


Figure 7.6: Waveplot for 2b

2c

URL: https://drive.google.com/file/d/18vTuG5k4_MubM_imIpGDQvfRLh8dPJWf/view?usp=sharing

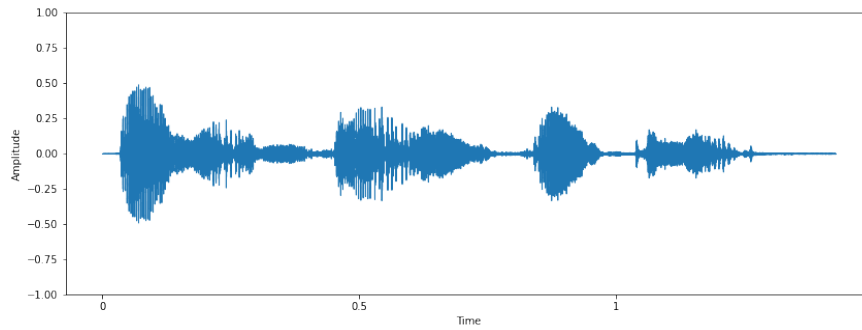


Figure 7.7: Waveplot for 2c

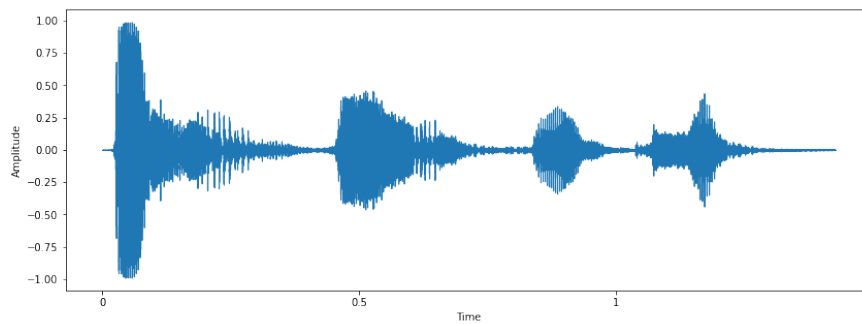
2dURL: <https://drive.google.com/file/d/1GQ9PCg2El6qRkl9tZtHTTv7EwyRtfBMQ/view?usp=sharing>

Figure 7.8: Waveplot for 2d

3: "I want you out of here"**3a**URL: https://drive.google.com/file/d/1v_8xWWiENbaZBd2K7vNP6Pv7g6fEMz1b/view?usp=sharing

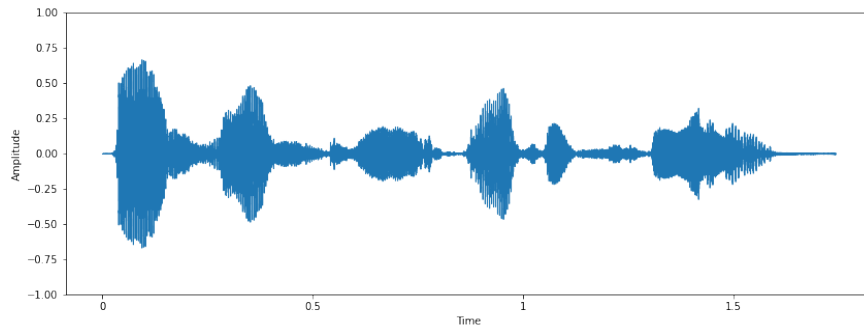


Figure 7.9: Waveplot for 3a

3b

URL: https://drive.google.com/file/d/1NopTO_cYcQL7ctXEZW8jdVc9HWDIf2tw/view?usp=sharing

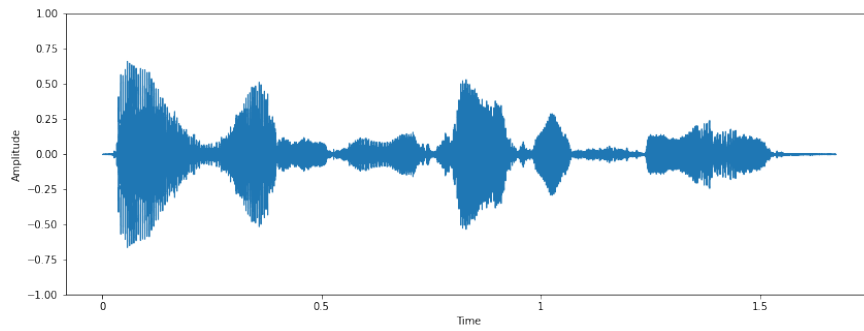


Figure 7.10: Waveplot for 3b

3c

URL: <https://drive.google.com/file/d/1PB0Pk1VzK564NopchQuTb1mhCURNeafj/view?usp=sharing>

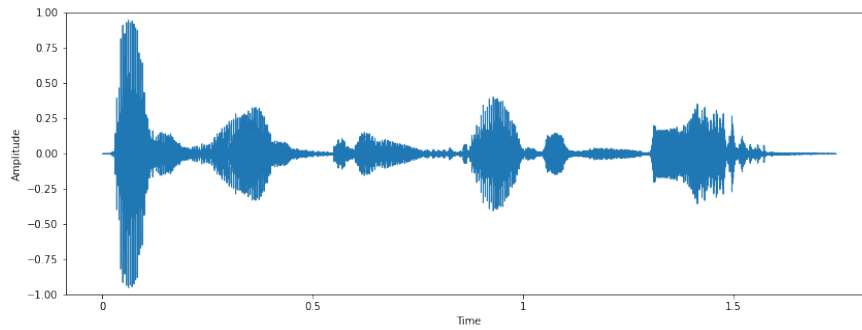


Figure 7.11: Waveplot for 3c

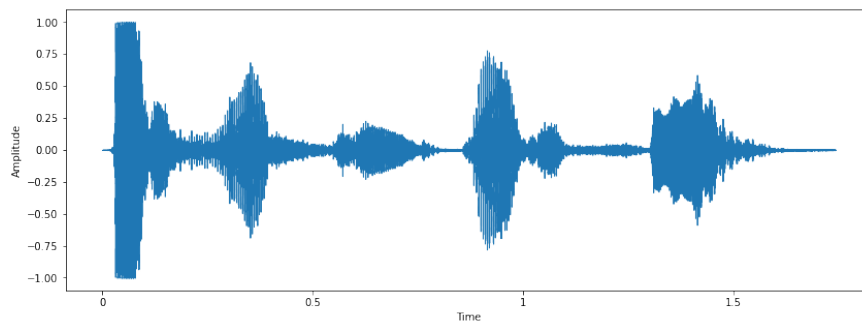
3dURL: https://drive.google.com/file/d/1fKUei5VK8L0y0EpAaHYf_0HVwCSr7Iuc/view?usp=sharing

Figure 7.12: Waveplot for 3d

4: "That is very unfair"**4a**URL: <https://drive.google.com/file/d/1i5WunkUvMuQqqfGUfaWnPjx20QNXDD-g/view?usp=sharing>

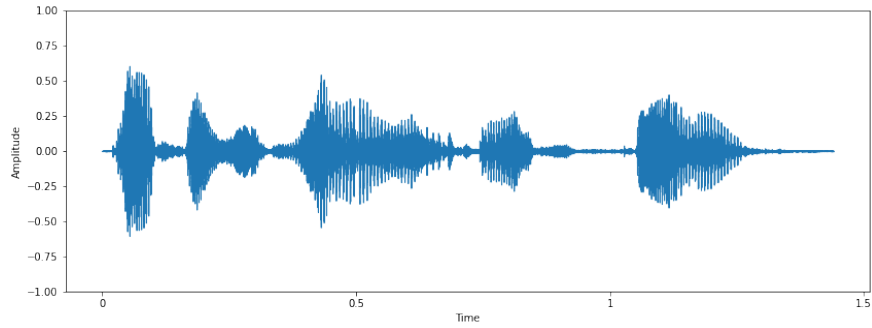


Figure 7.13: Waveplot for 4a

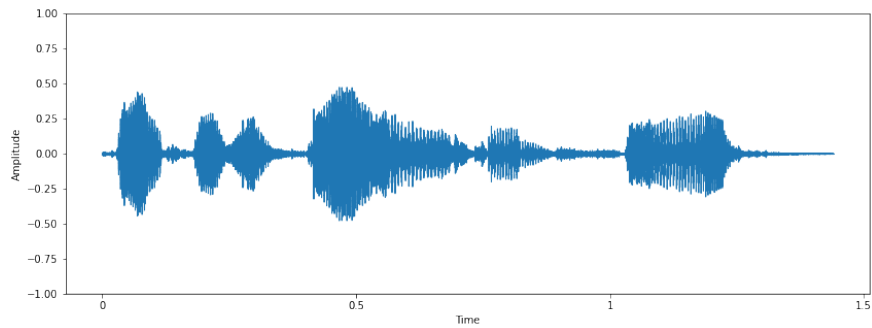
4bURL: https://drive.google.com/file/d/1Sp1mWkMI9LAFmLoCQ1cb4fHbciD_XKM6/view?usp=sharing

Figure 7.14: Waveplot for 4b

4cURL: <https://drive.google.com/file/d/13OfA-cLmDQo7PDFKRJIxaPMrCd6o2hhW/view?usp=sharing>

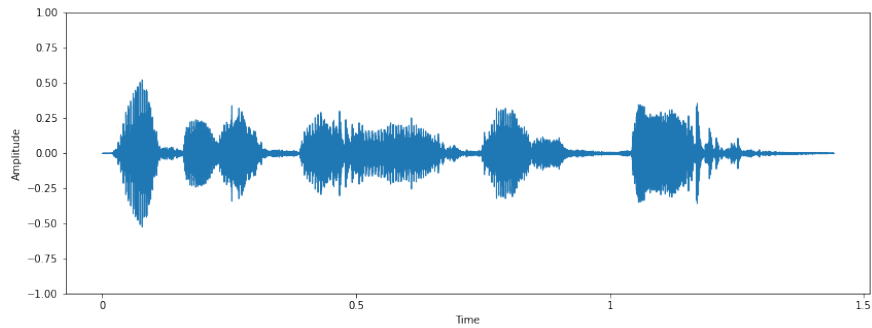


Figure 7.15: Waveplot for 4c

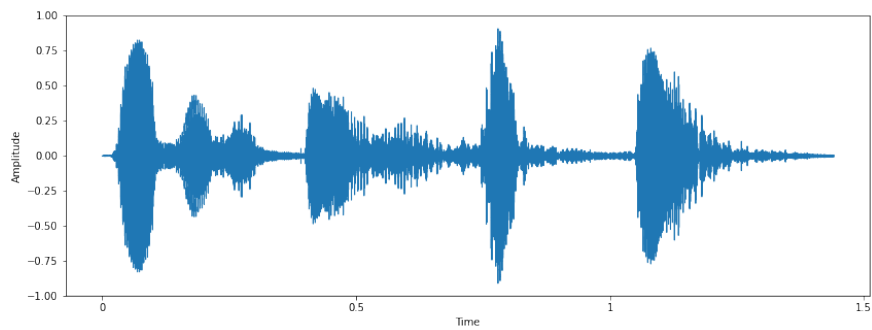
4dURL: <https://drive.google.com/file/d/1c06uBmkSc9lma6mnmIHU7ag0Rka44VoZ/view?usp=sharing>

Figure 7.16: Waveplot for 4d

5: "This ends here"**5a**URL: https://drive.google.com/file/d/1T9L-8Spsq5PijAizF6TyU0xzeI5aU_mu/view?usp=sharing

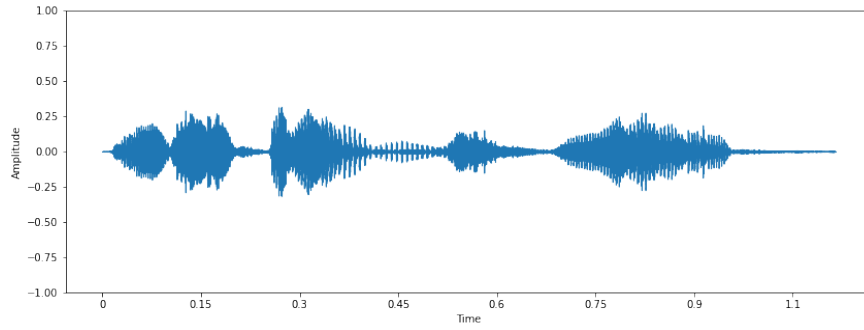


Figure 7.17: Waveplot for 5a

5b

URL: <https://drive.google.com/file/d/1CjU3xGlrUblU3BB76pUr5JCf0wF0IrG/view?usp=sharing>

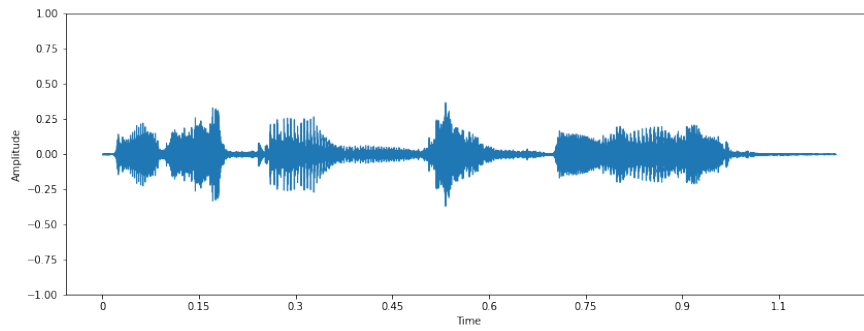


Figure 7.18: Waveplot for 5b

5c

URL: <https://drive.google.com/file/d/10JOYH7HUbbqMiooOloGaUwoJDpaF4FAqT/view?usp=sharing>

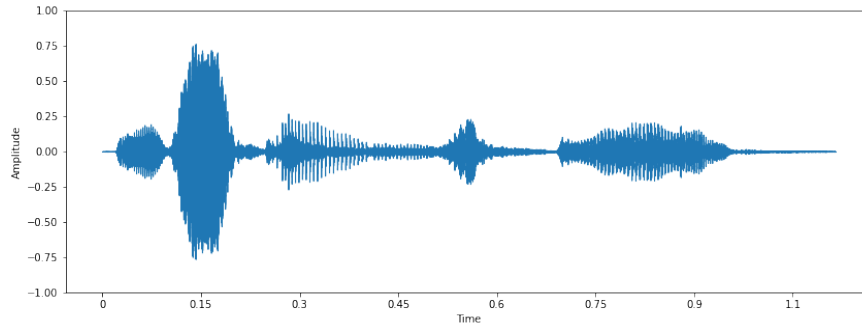


Figure 7.19: Waveplot for 5c

5d

URL: <https://drive.google.com/file/d/1mR31IJk27E2pF4X0lad8K-OKTPAdhZej/view?usp=sharing>

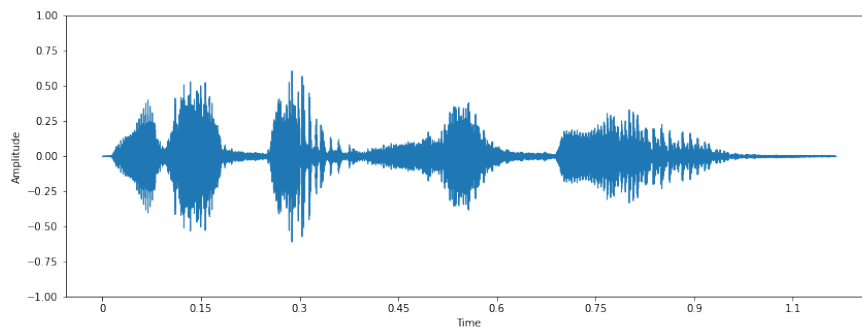


Figure 7.20: Waveplot for 5d

7.2 Algorithms

function REVERSEDIFFUSIONPPO(z , $mask$, μ , $nTimesteps$, $advantages$, $baseLogProbs$, $baseLatents$)

$$h \leftarrow \frac{1}{nTimesteps}$$

$$x_t \leftarrow z * mask$$

▷ Use the noise tensor z to get the current state x_t

for each t in $nTimesteps$ **do**

$$steps \leftarrow steps + 1$$

$$noise_t \leftarrow get_noise(h, t, z)$$

$$dx_t_det \leftarrow 0.5 * (\mu - x_t) - estimator(x_t, mask, \mu, t)$$

$$dx_t_det \leftarrow dx_t_det * noise_t * h$$


```

sdvt ← sqrt(noiset * h)
dxt ← baseLatents[t]

log_prob ← log_normal_pdf(x = dxt_det, μ = dxt, σ = sdvt)
log_prob ← mean_all_but_batch_dim(log_prob)

advantages ← clamp(advantages, clamp_boundaries)
ratio ← exp(log_prob − baseLogProbs[t])
unclipped_loss ← −advantages * ratio
clipped_loss ← −advantages * clamp(ratio, 1.0 − clip_range, 1.0 + clip_range)

loss ← mean(element_wise_maximum(unclipped_loss, clipped_loss))

```

Update current state x_t

Accumulate Gradients

```

if steps % accumulation_steps = 0 then
    Clip Gradient Norm
    Step the Optimizer
    Zero Out Gradients
end if
end for
end function

```