Speak, Read and Prompt: High-Fidelity Text-to-Speech with Minimal Supervision

Eugene Kharitonov[1], **Damien Vincent**[2], **Zalán Borsos**[2], **Raphaël Marinier**[1], **Sertan Girgin**[1], **Olivier Pietquin**[1], **Matt Sharifi**[2], **Marco Tagliasacchi**[2], **Neil Zeghidour**[1]

[1]Google, France [2]Google, Switzerland

{kharitonov,damienv,neilz}@google.com

Abstract

We introduce SPEAR-TTS, a multi-speaker text-to-speech (TTS) system that can be trained with minimal supervision. By combining two types of discrete speech representations, we cast TTS as a composition of two sequence-to-sequence tasks: from text to high-level semantic tokens (akin to ''reading'') and from semantic tokens to low-level acoustic tokens (''speaking''). Decoupling these two tasks enables training of the ''speaking'' module using abundant audio-only data, and unlocks the highly efficient combination of pretraining and backtranslation to reduce the need for parallel data when training the ''reading'' component. To control the speaker identity, we adopt example prompting, which allows SPEAR-TTS to generalize to unseen speakers using only a short sample of 3 seconds, without any explicit speaker representation or speaker labels. Our experiments demonstrate that SPEAR-TTS achieves a character error rate that is competitive with state-of-the-art methods using only 15 minutes of parallel data, while matching ground-truth speech in naturalness and acoustic quality.

1 Introduction

Training a text-to-speech (TTS) system typically requires hundreds of hours of parallel data in the form of transcribed utterances. As a consequence, high-quality TTS is only available for a few dozen languages. Moreover, the audio generated by these systems is only as diverse as the parallel data that they are trained on, which should contain many speakers, with various accents, of diverse demographics, and in heterogeneous recording conditions. At the same time, extremely diverse audio-only data can be relatively abundant online, present in the forms of podcasts, radio, and TV shows. This makes it attractive to use such audio-only data for building realistically sounding TTS systems while minimizing the need for diverse parallel data.

In this paper, we investigate how audio-only data can be leveraged to reduce the need for supervision in training TTS systems. We introduce SPEAR-TTS,[1] a multi-speaker TTS system that can be trained with as little as 15 minutes of parallel data from a single speaker. Moreover, SPEAR-TTS can synthesize a new voice using only 3 seconds of speech, without any speaker labels or explicit speaker representation. SPEAR-TTS leverages recent advances in the ''textless'' modeling of spoken language (Lakhotia et al., 2021; Polyak et al., 2021; Kreuk et al., 2021; Kharitonov et al., 2022; Borsos et al., 2023). These methods represent continuous audio waveforms as sequences of tokens from a finite vocabulary, casting speech generation as a language modeling task. In particular, AudioLM (Borsos et al., 2023) combines two types of discrete tokens: high-level semantic tokens and low-level acoustic tokens, which can be mapped to audio. Using these representations, we cast the TTS problem as a ''translation'' from text transcripts to acoustic tokens with semantic token representations serving as a pivot ''language'' (Utiyama and Isahara, 2007). This way, TTS is a composition of two sequence-to-sequence (seq2seq) tasks: translating text to semantic tokens, and translating semantic to acoustic tokens.

The key benefit of such an approach is that the supervision needed to learn how to map text into the intermediate semantic token representation (''reading'') and how to produce speech from it (''speaking'') become decoupled. While the ''reading'' stage relies on parallel text-audio data, the audio tokens used to train the ''speaking'' component are produced by self-supervised audio models and therefore can be extracted from a

[1]SPEAR stands for ''**sp**eak, r**ea**d and p**r**ompt''.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 1703–1718, 2023. https://doi.org/10.1162/tacl_a_00618
Action Editor: Karen Livescu. Submission batch: 4/2023; Revision batch: 7/2023; Published 12/2023.

massive amount of unlabeled speech data. As a result, the quality and diversity of the generated speech become independent from the available parallel data.

Casting each stage of SPEAR-TTS as a seq2seq problem allows us to use standard Transformer models (Vaswani et al., 2017) and makes it easy to tap into the vast pool of ideas developed by the machine translation community to reduce the need for supervision. Specifically, we combine BART-style pretraining (Lewis et al., 2020) with backtranslation (Sennrich et al., 2016) to significantly reduce the amount of parallel supervision required to train SPEAR-TTS.

To control the voice used by SPEAR-TTS when producing an utterance, we leverage an example prompting mechanism that is closely related to prompting in textual language models (Brown et al., 2020). Here we condition the ''speaking'' model with an audio clip representing the target voice, steering it to use this voice when generating the utterance. This feature can simplify building controllable multi-speaker TTS systems for languages where only single-speaker parallel data is available.

Our experimental study on English speech shows that, by combining pretraining and backtranslation over a large dataset (551 hours) with just 15 minutes of parallel data from a single speaker, SPEAR-TTS:

- Generates speech with high fidelity to the input transcript—CER 1.92% on LibriSpeech test-clean (Panayotov et al., 2015);
- reliably reproduces the voice of an unseen speaker, when using a 3-second example from the target speaker;
- achieves high acoustic quality, comparable to that of the ground-truth utterances (MOS 4.96 vs. 4.92).[2]

Overall, our approach to building TTS using massive self-supervised pretraining and backtranslation of discrete speech representations considerably differs from how existing TTS systems are implemented (Shen et al., 2018; Kong et al., 2020; Ren et al., 2020; Kim et al., 2021; Ao et al., 2022; Wang et al., 2023), significantly reducing the costs related to data collection and potentially providing high-quality multi-speaker TTS for languages that are not well covered today.

[2]Samples can be found on the demo site: `https://google-research.github.io/seanet/speartts/examples/`.

2 Related Work

Our work relates to several research directions that we overview in this Section.

Discretized Speech Processing The work of Lakhotia et al. (2021) on generative spoken language modeling and generation pioneered the use of language models on discretized speech representations. Their work became a foundation for a range of extensions, including emotion transfer (Kreuk et al., 2021), prosody (Kharitonov et al., 2022), and dialog (Nguyen et al., 2023) modeling.

SPEAR-TTS is based on AudioLM (Borsos et al., 2023), a recent development in this line of work that achieves a superior quality in spoken language modeling and a high audio quality. SPEAR-TTS extends AudioLM along several dimensions. We adapt it to the text-to-speech scenario by conditioning it with a transcript input and show that by combining pretraining and backtranslation, we can dramatically reduce the amount of supervision required to train a high-fidelity TTS system. Finally, SPEAR-TTS explicitly incorporates a prompt-like mechanism for a voice control.

Low- and Semi-supervised TTS Guided-TTS (Kim et al., 2021) is another TTS system that leverages audio-only data. It combines (a) a denoising diffusion probabilistic model (DDPM) that learns to model audio-only data, and (b) a phoneme classifier that guides the generative process towards producing an utterance with a desired transcript. Guided-TTS 2 (Kim et al., 2022) extends it by allowing speaker adaptability either via finetuning or in a zero-shot manner, using a 10-second speech sample processed by a dedicated speaker embedding module. Another adaptable DDPM-based TTS system was proposed by Levkovitch et al. (2022), which uses the classifier guidance mechanism to steer generation towards a particular voice in a zero-shot manner.

In contrast to SPEAR-TTS, the above works rely on stronger supervision: a phoneme classifier that is trained on LibriSpeech 960 and a pretrained speaker verification system. Conversely, SPEAR-TTS uses a parameter-less prompting

mechanism which does not require any speaker labels.

Liu et al. (2020) combine a sequential auto-encoder with vector quantization and temporal segmentation mechanisms to learn a phoneme-like discrete speech representation, along with a seq2seq model that maps these representations to phonemes. Similarly to SPEAR-TTS, this system can be trained with almost no supervision, however the generated speech is single-speaker only and of much lower quality than ground-truth audio (2.33 vs 4.81 in their experiments).

Prompted Audio Generation When a sentence is prepended by an emotional prompt, expressed in plain English, e.g., [*I am really sad,*] Tortoise TTS (Betker, 2022) synthesizes text in a sad voice.

Wang et al. (2023) propose VALL-E, a TTS system that allows prompt-based conditioning of the synthesized voice and emotion. In contrast to the two-stage architecture of SPEAR-TTS, VALL-E predicts an equivalent of acoustic tokens directly from a phoneme representation of a text. This is unlike SPEAR-TTS which only prompts the model with self-supervised audio tokens. Another difference is the amount of the parallel training data used: VALL-E is trained on the 60k hours of ASR-transcribed LibriLight (Kahn et al., 2020).

3 Discrete Speech Representations

Below is a brief overview of the two self-supervised audio representations which we adapt from AudioLM. These representations occupy the opposite ends of the reconstruction quality vs. bitrate trade-off: Acoustic tokens are typically high-bitrate, allowing high-fidelity audio generation, while semantic tokens are low-bitrate, thus making it easier to achieve a long-span coherence. A detailed discussion can be found in Borsos et al. (2023)[Section III-B].

Semantic Tokens The role of semantic tokens is to provide a coarse, high-level conditioning to subsequently produce acoustic tokens. Thus, they should provide a representation of speech where linguistic content—from phonetics to semantics—is salient, while paralinguistic information such as speaker identity and acoustic details are removed. To obtain such a representation, we train a self-supervised speech representation model based on w2v-BERT (Chung et al., 2021). After its training, we run a k-means clustering on the mean-variance normalized outputs of a specific layer. We use the centroid indices as discrete tokens.

Acoustic Tokens Acoustic tokens are discrete audio representations that provide high-fidelity reconstruction of the acoustic details. We train a SoundStream (Zeghidour et al., 2021) codec to reconstruct speech while compressing it into few discrete units using a residual quantizer. To represent the hierarchy of residual quantizers in a sequence, we flatten the tokens corresponding to the different levels by interleaving them (Borsos et al., 2023). We use the SoundStream codec to convert audio to acoustic tokens and synthesize audio back from acoustic tokens.

4 SPEAR-TTS Overview

SPEAR-TTS extends AudioLM (Borsos et al., 2023) by enabling text as a form of conditioning. Similarly to AudioLM, SPEAR-TTS is organized in two main stages, as illustrated in Figure 1. In the first stage ($\mathcal{S}_1$), text inputs are translated into a sequence of discrete semantic tokens. The second stage ($\mathcal{S}_2$) maps semantic tokens into acoustic tokens, which are decoded to speech by the SoundStream decoder (Zeghidour et al., 2021). This way, $\mathcal{S}_1$ learns to map text to the internal representation provided by semantic tokens (‘‘reading’’), while $\mathcal{S}_2$ handles the production of speech from this intermediate internal representation (‘‘speaking’’).

By using semantic tokens as an intermediate representation, we achieve two goals. First, semantic tokens provide us with a high-level representation of speech. Thus, it is easier to learn a mapping from text transcripts to semantic tokens than directly between text and acoustic tokens. Second, as both semantic and acoustic tokens are derived from self-supervised models, the second stage $\mathcal{S}_2$ can be trained using audio-only data. This turns out to be extremely beneficial for training $\mathcal{S}_2$, as the typical scale of available audio-only data is considerably larger than that of parallel data.

In turn, separating $\mathcal{S}_1$ from $\mathcal{S}_2$ allows us to pretrain the former with a denoising pretext task operating on succinct semantic tokens, further harnessing audio-only data.

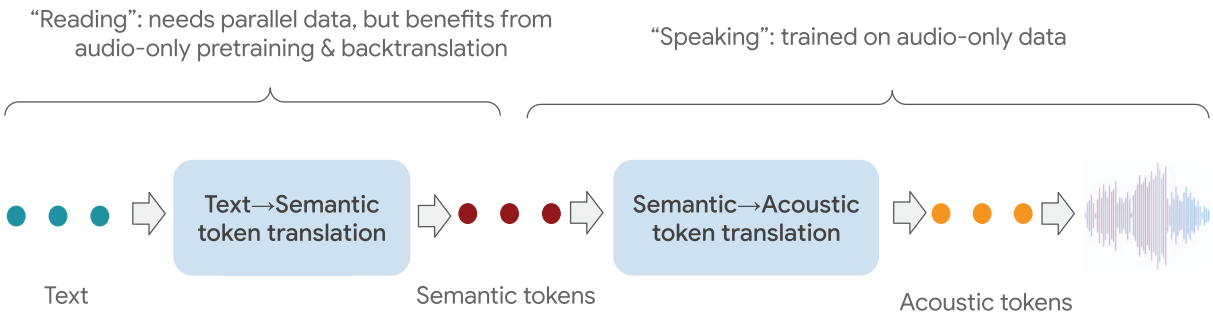

Figure 1: **SPEAR-TTS.** The first stage $\mathcal{S}_1$ (''reading'') maps tokenized text to semantic tokens. The second stage $\mathcal{S}_2$ (''speaking'') maps semantic tokens to acoustic tokens. Acoustic tokens are decoded to audio waveforms using the SoundStream decoder.

5 $\mathcal{S}_1$: Improving Supervision Efficiency

The first stage $\mathcal{S}_1$ maps tokenized text into semantic tokens. We use parallel text-semantic tokens data to learn this mapping. We start with a text-audio TTS dataset and extract semantic tokens from audio. As a result, $\mathcal{S}_1$ is reduced to a seq2seq task that can be implemented by encoder-decoder or decoder-only Transformer architectures (Vaswani et al., 2017).

Training Transformer seq2seq models can require substantial amounts of parallel data, which can be extremely scarce for some languages. In the following, we discuss two approaches used to alleviate this limitation: target domain pretraining (§ 5.1) and backtranslation (§ 5.2).

5.1 Pretraining

We take inspiration from BART and pretrain an encoder-decoder Transformer on a denoising task (Lewis et al., 2020). In this task, the model is provided with a corrupted version of an original semantic token sequence and the goal is to produce the corresponding uncorrupted token sequence. Such pretraining does not require parallel data and we carry it out using a large audio-only dataset.

Typical corruption methods include random substitution, deletion and masking of individual tokens or entire spans of tokens (Lewis et al., 2020; Raffel et al., 2020). In preliminary studies, we observed that deleting individual tokens independently at random works better than other alternatives.

After pretraining the model $\mathcal{P}$, we finetune it for the $\mathcal{S}_1$ task. To achieve this, we freeze the upper layers of the encoder and all parameters of the decoder, excluding the parameters used in the decoder-encoder cross-attention layers, and update the lower layers of the encoder. The exact number of layers to tune is a hyperparameter.

5.2 Backtranslation

The same text sequence can be rendered as audio in multiple ways, with varying voice, accent, prosody, emotional content, and recording conditions. This one-to-many relationship makes the text-to-speech problem highly asymmetric—unlike text translation, where, for example, English-French translation is roughly equally hard in either direction. Thus, it is very attractive to use backtranslation (Sennrich et al., 2016; Edunov et al., 2018), i.e., to use the available parallel data to train a speech-to-text model and use it to generate synthetic parallel data from an audio-only corpus.

The two-stage architecture of SPEAR-TTS is particularly suitable for backtranslation as it can be implemented as translation between semantic tokens and text. The benefits are two-fold: (a) a reduction in the computational complexity due to never dealing with raw audio or long acoustic token sequences,[3] and (b) the ability to leverage the same semantic token-level pretraining (§ 5.1) when training the ''backward''-direction model.

In order to obtain a backtranslation model, we start from the same pretrained model $\mathcal{P}$ as above. However, this time we freeze the encoder and only finetune the decoder. Afterwards, we transcribe audio-only data using this model. Next, we use the synthetically generated parallel data to train the first stage of the TTS system, which, in turn, is also obtained via finetuning another copy of $\mathcal{P}$ (see § 5.1). After finetuning on the synthetic

[3] An acoustic representation is $\geq 6\times$ longer than the semantic one.

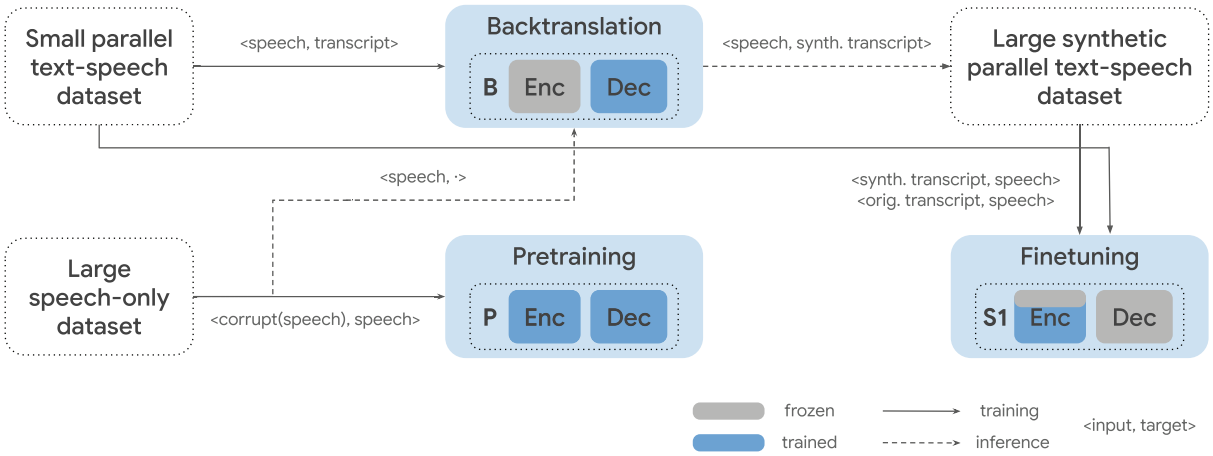

Figure 2: **Training $\mathcal{S}_1$, combining pretraining and backtranslation**. We start with pretraining an encoder-decoder model $\mathcal{P}$ on a denoising task, using a semantic-token representation of speech-only data. Next, we finetune its decoder to backtranslate (semantic tokens to transcripts) using a small parallel dataset. Then, we use this model to transcribe the speech-only dataset and obtain a synthetic parallel dataset. In turn, this synthetic dataset is used to finetune the encoder of $\mathcal{P}$ for ''translation'' in the forward direction (text transcripts to semantic tokens), along with the original small parallel dataset.

data, we continue finetuning on the original parallel data. Figure 2 illustrates this process.

6 $\mathcal{S}_2$: Controlling the Generation Process

The second stage model $\mathcal{S}_2$ maps semantic tokens into acoustic tokens. To train it, we extract pairs of semantic and acoustic token sequences from each utterance in an audio-only dataset. Next, we train a Transformer model to perform seq2seq translation between the two token sequences. The second stage generates utterances with randomly varying voice, tempo, and recording conditions, reproducing the distribution of the characteristics observed in the training data. As $\mathcal{S}_1$ and $\mathcal{S}_2$ are trained independently, this diversity of speech is also preserved when $\mathcal{S}_1$ is trained on a single-speaker dataset.

To control the characteristics of the speaker's voice, we combine two findings from AudioLM (Borsos et al., 2023): (a) whenever the speech prefix is represented solely by semantic tokens, AudioLM generates continuations by sampling a different random voice each time; (b) however, when conditioning also includes acoustic tokens, AudioLM maintains the voice characteristics captured by the acoustic tokens when generating the continuation. In contrast to AudioLM, we explicitly incorporate this ability during training, as illustrated in Figure 3. During training, we randomly select two non-overlapping windows of speech from each training example, from which we compute sequences of semantic and acoustic tokens. We consider one of the windows as the prompt and the other as the target output. Next, we concatenate the sequences in the following order: (a) semantic tokens from the prompt, (b) semantic tokens from the target, (c) acoustic tokens from the prompt, and (d) acoustic tokens from the target. During training of $\mathcal{S}_2$, (a)–(c) are used as prefix and the model learns to generate the target acoustic tokens (d), preserving the speaker identity captured by the acoustic tokens from the prompt. At inference time, (a)–(c) are provided as input, and (d) is generated autoregressively. We also add a special separator token to inform the model about the expected discontinuity. This prevents boundary artifacts, which sometimes occur otherwise. Note that the text transcript of the prompt is not needed.

The speech samples generated by $\mathcal{S}_2$ might contain some background noise, since this is typically present in the training data. We consider two methods to control the noise level in the synthesized speech at inference time. First, in the case of prompted generation, it is possible to select prompts containing cleaner speech. Second, we can use a stochastic sampling (e.g., temperature sampling), generate multiple sequences for the same input and then use a no-reference audio quality metric to select the samples containing the least amount of noise. To this end, we use a MOS estimator model similar to DNSMOS (Reddy et al., 2021).

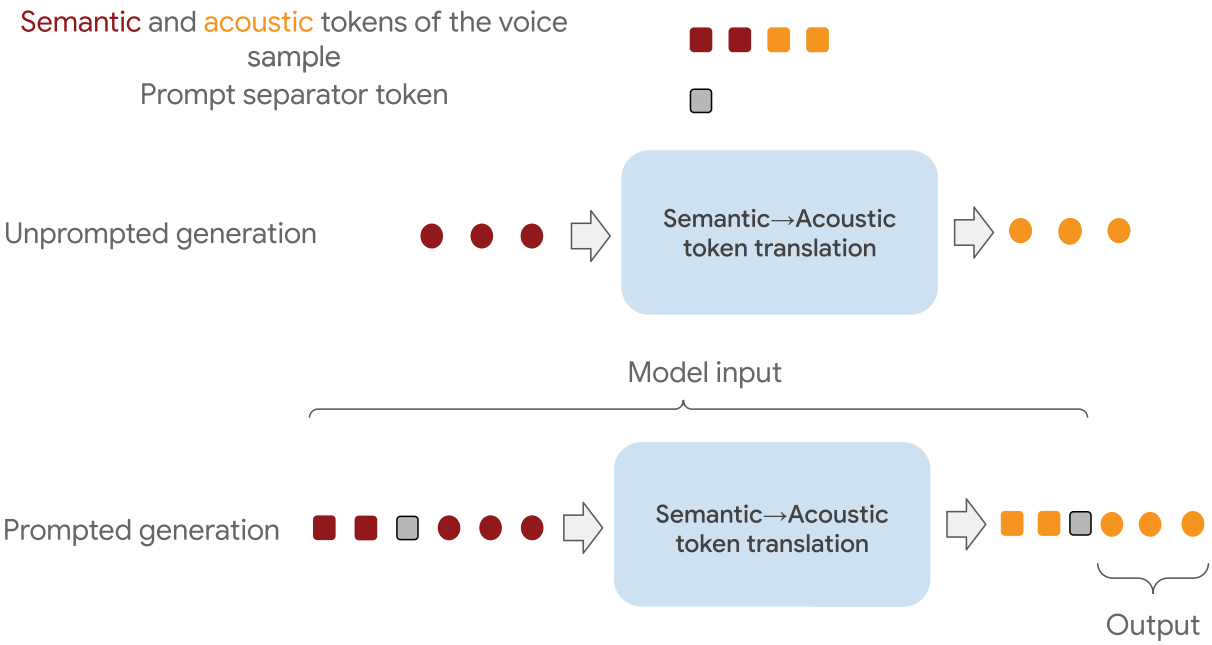

Figure 3: **Controlling generation with example prompting in $\mathcal{S}_2$.** For prompted generation, we concatenate token sequences in the following order: semantic tokens from the prompt, semantic tokens from the target, acoustic tokens from the prompt. Then, the model generates acoustic tokens corresponding to the semantic tokens from the target, while preserving the voice and speaking conditions in the acoustic tokens from the prompt. Squares correspond to the parts of the prompt.

Name	Size, hours	Transcripts used?	Used for
LibriLight	60k	✗	Acoustic & semantic tokens, pretraining $\mathcal{S}_1$, and training $\mathcal{S}_2$
LJSpeech	0.25..24	✓	Finetuning $\mathcal{S}_1$ for backtranslation
LibriTTS train	551	✗	Source of backtranslated data
LibriSpeech test-clean (shorter than 10s)	3	✓	Intelligibility evaluation
LibriSpeech train-clean + test-clean	105	✗	Training the voice classifier used in the evaluation

Table 1: **Datasets used in the paper.** For each dataset, we highlight its size, use, and whether textual transcripts are used.

7 Experimental Setup

7.1 Training and Validation Data

In this Section, we describe datasets that are used in this paper for training and evaluation. We provide an outline in Table 1.

Acoustic and Semantic Tokens: We use LibriLight (Kahn et al., 2020) to train the self-supervised representation models (SoundStream and w2v-BERT) as well as the k-means used to discretize w2v-BERT embeddings into semantic tokens. We use the largest unlab-60k split of LibriLight that contains around 60,000 hours of English audiobooks read by more than 7,000 speakers.

First Stage $\mathcal{S}_1$: To experiment in the low-supervision regime, we train $\mathcal{S}_1$ on LJSpeech (Ito and Johnson, 2017), a single-speaker dataset containing 24 hours of parallel data. By using LJSpeech as the only source of parallel data, we also show that our method generalizes to multiple speakers, even if the parallel training data itself contains only a single speaker. Since LJSpeech

does not specify a canonical train/dev/test split, we follow Liu et al. (2022, 2020) and randomly select 300 utterances as development and another 300 utterances as test set (30 minutes each), using the rest as training data. To simulate scenarios in which very limited data is available, we uniformly sample subsets of 3 hours, 1 hour, 30 minutes, and 15 minutes from the training set. As an indicative figure, the 15 minute subset contains around 21k semantic tokens and 2k words.

Pretraining: To pretrain a model on the sequence corruption task (§ 5.1), we extract semantic tokens from LibriLight (Kahn et al., 2020), since pre-training only requires audio data.

Backtranslation: In our experiments with backtranslation, we use LibriTTS (Zen et al., 2019) as a source of unlabelled speech (ignoring transcripts). We pool all training subsets of LibriTTS to obtain an audio-only dataset containing 551 hours of speech. Using LibriTTS as a source for audio-only data for backtranslation allows us to compare SPEAR-TTS with $\mathcal{S}_1$ trained on original and backtranslated LibriTTS transcripts.

Second Stage $\mathcal{S}_2$: To train $\mathcal{S}_2$, we extract pairs of semantic and acoustic token sequences from LibriLight (Kahn et al., 2020).[4]

7.2 Evaluation Data

We use LibriSpeech test-clean (Panayotov et al., 2015) to measure the character error rate (CER) (see § 7.4). As LJSpeech only contains sequences shorter than 10 seconds, we filter out sequences longer than that from LibriSpeech test-clean. As a result, we obtain 2,007 utterances, with a total duration of approximately 3 hours. Importantly, LibriSpeech test-clean has no intersection with any training or validation data we used.

7.3 Preprocessing

To prepare the data for training, we unroll standard abbreviations used in LJSpeech. Next, we apply the G2p_en phonemizer (Park and Kim, 2019). After removing the lexical stress information from its output, we obtain a string representation in a vocabulary of 47 tokens (39 phonemes from the CMU Dictionary, whitespace, and punctuation).

[4]In Appendix A, we study the impact of reducing the amount of data used for training $\mathcal{S}_2$ on the overall performance.

7.4 Metrics

Character Error Rate (CER) We transcribe the synthesized utterances using an in-house ASR system and we evaluate the faithfulness to the input transcript by measuring the character error rate (CER). We use the LibriSpeech test-clean dataset (Panayotov et al., 2015) to calculate CER, since it requires minimal postprocessing to be compared to the output of the adopted ASR system. On the original audio, CER is equal to 0.98%.

Voice Preservation When prompting the model with a short utterance, we evaluate the consistency of the speaker voice between the prompt and the generated speech. We use the same speaker classifier as Borsos et al. (2023), which is trained on a union of LibriSpeech train-clean-100 and test-clean (251 and 40 speakers, respectively). It computes predictions over a set of 291 speaker classes (see Appendix C for more details). We measure how often the speaker label predicted from the generated speech matches the one predicted from the prompt.

Quality We rely on human judgments to evaluate the perceived quality of SPEAR-TTS by collecting Mean Opinion Scores (MOS). In this context, human raters listen to individual audio segments and rate their audio quality and speech naturalness on a scale from Poor (1) to Excellent (5).

7.5 Baselines

As our main baseline, we consider a system explicitly trained to target the low-supervision scenario. Namely, we use a modification of FastSpeech2 (Ren et al., 2020), which is a non-autoregressive model that uses auxiliary duration, pitch, and energy predictors. Specifically, in our experiments we consider the adaptation to the low-supervision setting by Pine et al. (2022). The model takes as input the phoneme representation of the text and predicts a spectrogram, which is then vocoded with HiFi-GAN (Kong et al., 2020). We denote this modification as FastSpeech2-LR. In a subjective evaluation reported by Pine et al. (2022), FastSpeech2-LR trained on 1 (3) hour(s) of parallel data performed on par with an open-source implementation of Tacotron2 (Shen et al., 2018) trained with 10 (24) hours of parallel data. We use checkpoints trained on 15 minutes, 30 minutes,

1 hour, 3 hours, and 24 hours that were shared by the authors.[5]

We also compare SPEAR-TTS to VALL-E (Wang et al., 2023), a recent TTS system that demonstrates state-of-the-art results in zero-shot voice adaptation. Similarly to SPEAR-TTS, it is capable of voice transfer using a 3-second voice prompt. VALL-E maps the input text to coarse acoustic tokens, and uses a non-autoregressive refinement stage to predict fine-grained acoustic tokens. VALL-E is trained on an ASR-transcribed version of LibriLight (Kahn et al., 2020), containing roughly 60,000 hours of parallel data. Since the model is not publicly available, the comparison is based on the samples provided on its demo page.

8 Hyperparameters & Training Details

8.1 Discrete Speech Representations

We follow the setup of AudioLM (Borsos et al., 2023) to compute both semantic and acoustic tokens, with a few differences. The semantic tokens are obtained by quantizing the embeddings returned by the 7th layer of w2v-BERT using a codebook of size 512. As a result, 1 second of audio is represented by 25 semantic tokens with a vocabulary size of 512, resulting in a bitrate of 225 bit/s. We remove sequentially repeated semantic tokens, as done in Lakhotia et al. (2021) and Borsos et al. (2023).

We extract acoustic tokens from a SoundStream neural codec (Zeghidour et al., 2021) with 3 quantization levels, each with a codebook of size 1024. We use a vocabulary with 3×1024 unique tokens and represent each frame as a flat sequence of tokens, interleaving the first, second, and third quantization layers, respectively. As a result, 1 second of audio is represented by 50 Hz $\times$ $3 = 150$ acoustic tokens, a bitrate of 1500 bit/s.

8.2 First Stage ($\mathcal{S}_1$)

In all experiments, we use the Adafactor optimizer (Shazeer and Stern, 2018) with inverse square-root learning rate decay. As a regularization method, we use label smoothing with the smoothing parameter set to 0.1, except in the case of pretraining, when a large amount of data is available.

[5] https://github.com/roedoejet/FastSpeech2_ACL2022_reproducibility.

	Embed. dim.	FFN dim.	Head dim.	# heads
T5-small	256	512	64	6
T5-base	768	2048	64	12
T5-large	1024	2816	64	16

Table 2: **Architecture details.** We report details for T5-small, T5-base, and T5-large layers. The number of layers used is defined by a grid search (Section 8).

Pretraining The pretraining task is configured so that the probability of deleting individual tokens is set to 0.6. This parameter was selected via grid search inspecting the validation accuracy of $\mathcal{S}_1$ after finetuning. We apply dropout with probability equal to 0.5 and set the batch size to 256. We ran the pretraining for 1M updates and used the resulting checkpoint $\mathcal{P}$ in all our experiments. As the architecture, we use T5-Large (Raffel et al., 2020), which is a 24 layer encoder-decoder model (see Table 2).

Finetuning The same pretrained checkpoint $\mathcal{P}$ is finetuned for different purposes (Figure 2). In all cases we perform a grid search on the dropout rate ($\{0.1, 0.3, 0.5\}$) and the number of layers to finetune, selecting the combination with the highest validation accuracy (with teacher-forcing). More specifically, when finetuning on ground-truth parallel data (as an ablation), we freeze both the upper layers of the encoder and the entire decoder, while updating the weights of the encoder embeddings and the lower layers. The number of the lower layers to tune is searched in $\{4, 6, 8\}$. When finetuning on synthetic parallel data, we search over the number of the encoder's lower layers to be finetuned in $\{4, 6, 8, 10, 12, 24\}$. Next, we finetune the lower 4 layers of the encoder on the original parallel data (to avoid overfitting when very little data is available). Finally, when finetuning the decoder for backtranslation, we finetune N top and N bottom layers, with $N \in \{2, 3, 4, 12\}$. During finetuning, we select the checkpoint with the best validation accuracy.

Training from Scratch As an ablation experiment, we train $\mathcal{S}_1$ from scratch, experimenting with different variants of T5 architectures (Raffel et al., 2020), depending on the amount of data

Parallel training data	FastSpeech2-LR	SPEAR-TTS			
		Training from scratch (a)	Pretraining (b)	Backtranslation from scratch (c)	Backtranslation pretraining (d)
24 h	$1.99_{\pm 0.20}$	$3.67_{\pm 0.21}$	$2.38_{\pm 0.13}$	$2.26_{\pm 0.14}$	$2.06_{\pm 0.12}$
3 h	$2.52_{\pm 0.25}$	$20.1_{\pm 0.74}$	$3.07_{\pm 0.15}$	$2.21_{\pm 0.12}$	$2.01_{\pm 0.12}$
1 h	$2.74_{\pm 0.27}$	$\times$	$5.51_{\pm 0.21}$	$2.23_{\pm 0.13}$	$2.16_{\pm 0.13}$
30 min	$3.18_{\pm 0.28}$	$\times$	$21.3_{\pm 0.43}$	$2.52_{\pm 0.15}$	$2.20_{\pm 0.12}$
15 min	$4.90_{\pm 0.34}$	$\times$	$\times$	$2.88_{\pm 0.19}$	$2.21_{\pm 0.12}$

Table 3: **Intelligibility** of SPEAR-TTS and our baselines, depending on the training scenario and the amount of parallel data available from LJSpeech. We measure CER (%, lower is better) on LibriSpeech test-clean. $\pm$ indicates 95% CI obtained by bootstrap. ''$\times$'' indicates models that produce unintelligible speech.

available. We adopt a decoder-only model without causal masking on the input sequence (Raffel et al., 2020), which led to better results in our preliminary experiments. We perform a grid-search on the following hyperparameters: dropout probability $\{0.1, 0.3, 0.5\}$; architecture size (T5-small or T5-base, see Table 2); the number of layers (T5-small: 2, 4, 6, 8; T5-base: 4, 6, 8, 12).

8.3 Second Stage ($\mathcal{S}_2$)

For $\mathcal{S}_2$, we use a 12-layer decoder-only Transformer model, with each layer having 12 heads with dimensionality 64, embedding dimensionality of 768, and FFN size of 2048. The optimizer and the learning rate schedule are the same as for $\mathcal{S}_1$.

8.4 Inference

We use beam search to sample from $\mathcal{S}_1$ and temperature sampling to sample from $\mathcal{S}_2$. This combination ensures faithfulness to the transcript while enabling more diverse and natural sounding speech. We use a beam size equal to 10, as larger values do not lead to improvements in CER but are more computationally expensive. When generating backtranslation data we re-use the settings of $\mathcal{S}_1$, without running any additional hyperparameter search. For $\mathcal{S}_2$, we experiment with sampling temperatures $T \in \{0.50, 0.55, \ldots, 0.95, 1.0\}$ and select $T = 0.75$ which minimizes the CER on the LJSpeech validation dataset. In this case, the $\mathcal{S}_1$ model is trained on synthetically generated parallel data obtained by backtranslation, with the backtranslation model trained on the 15 minute split of LJSpeech.

To control the noise levels in the synthesized speech, we employ the sampling technique (§ 6) where we sample $n_s = 3$ audio utterances corresponding to the input and select the one that has highest quality according to a no-reference audio quality model similar to DNSMOS (Reddy et al., 2021).

9 Experiments

We evaluate SPEAR-TTS along several dimensions. First, we measure the faithfulness of the generated speech to the input transcript, for different training scenarios and amounts of parallel data available (§ 9.1). Then, we show that SPEAR-TTS is able to successfully control the speaker voice, without any degradation in terms of fidelity to the transcript (§ 9.2).

9.1 Intelligibility and Supervision Efficiency

When evaluating SPEAR-TTS, we consider the following training settings for $\mathcal{S}_1$: *(a)* training from scratch using parallel data; *(b)* finetuning the pretrained checkpoint $\mathcal{P}$ using parallel data; *(c)* finetuning the pretrained checkpoint $\mathcal{P}$ to obtain the backtranslation model and then training the forward model from scratch on the synthetically generated data; *(d)* same as (c), but both the backward and the forward models are obtained by finetuning $\mathcal{P}$ with an additional finetuning of the forward model on the original parallel data.

Table 3 reports the main results in terms of CER, as a proxy for the intelligibility of the generated speech. We observe that when decreasing the amount of parallel data, training from scratch (a) results in very high error rates. Conversely, thanks to pretraining (b), SPEAR-TTS maintains a relatively low CER ($\leq 4\%$), when using as little as 3 hours of parallel data. This is similar to the CER

CER (%)	Speaker accuracy (%)		Voice diversity (bits)
	top-1	top-3	
1.92	92.4	98.1	0.41

Table 4: **Voice preservation in prompted generation.** $\mathcal{S}_1$ is trained on 15 min of parallel data.

Model	Parallel training data	Cosine similarity
YourTTS	$\sim$ 600 h	0.34
VALL-E	60,000 h	0.58
SPEAR-TTS	15 min	0.56

Table 5: **Comparing voice preservation with baselines (cosine similarity).** Results for YourTTS and VALL-E are taken from (Wang et al., 2023, Table 2).

achieved with 24 hours, but without pretraining. Backtranslation (c) has a general positive impact, especially when the amount of parallel data is reduced, achieving a CER of 2.88% with only 15 minutes. By combining backtranslation with pretraining (d), the CER is further decreased to 2.21% with the same amount of parallel data. This indicates that having a fixed decoder is useful to cope with the noisy nature of the synthetically generated training data obtained via backtranslation.

We also compare SPEAR-TTS to FastSpeech2-LR, observing that when using 24 hours of parallel data, both systems perform approximately on par (FastSpeech2-LR: 1.99% vs. SPEAR-TTS: 2.06%). However, as the amount of parallel data is reduced, CER of FastSpeech2-LR increases very rapidly. As a result, there is a significant gap when only 15 minutes are available, that is, FastSpeech2-LR: 4.90% vs. SPEAR-TTS: 2.21%.

In conclusion, the combination of pretraining and backtranslation allows SPEAR-TTS to synthesize speech that adheres to the input transcript, even with as little as 15 minutes of parallel data.

9.2 Prompted Generation

SPEAR-TTS is able to control the speaker voice via example prompting, as described in § 9.2. We evaluate SPEAR-TTS in a *zero-shot* scenario, in which the voice used for prompting was never seen by $\mathcal{S}_1$ or $\mathcal{S}_2$ at training and $\mathcal{S}_2$ has to reproduce its characteristics from a single prompt example. Specifically, we fix $\mathcal{S}_1$, using the model trained on 15-minutes of LJSpeech and we consider all 40 speakers from LibriSpeech test-clean as target speakers. For each speaker, we randomly select 5 speech prompts with duration of 3 seconds each and transcripts from the same dataset. For each speech prompt and text transcript, we repeat the synthesis 5 times and average the metrics.

Table 4 reports the speaker accuracy, that is, how often the same voice is detected in both the prompt and the generated speech. We provide details on the classifier's architecture and training in Appendix C. We observe top-1 accuracy equal to 92.4% showing that the prompting allows SPEAR-TTS to preserve the speaker voice. Also, the synthesized voice is stable when repeating inference, as captured by a low value of voice variability (0.41 bits). Moreover, we observe that with prompted generation SPEAR-TTS achieves a CER equal to 1.92%, which is lower than without prompting (2.21%). We believe that this improvement is due to using cleaner recordings for prompts, which steers the $\mathcal{S}_2$ model to produce cleaner speech and consequently reduce ASR errors.

We also compare the voice preservation abilities of SPEAR-TTS with those of VALL-E (Wang et al., 2023). Following the methodology of Wang et al. (2023) we compute the cosine similarity between embeddings computed from the prompt (encoded and decoded with SoundStream) and from the generated speech, using a publicly available speaker verification system based on WavLM (Chen et al., 2022). This is the same model as used by Wang et al. (2023), which makes our measurements directly comparable with scores reported in their paper. From the results reported in Table 5, we observe that SPEAR-TTS significantly outperforms YourTTS (Casanova et al., 2022) and almost matches the speaker similarity of VALL-E, despite being trained with 240,000$\times$ less parallel data.

10 Subjective Evaluation

Ultimately, we resort to subjective tests with human raters to compare the quality of SPEAR-TTS with the baselines and with ground-truth natural speech. We focus on the scenario with minimal supervision and use the $\mathcal{S}_1$ model that is trained with the 15 minute LJSpeech (Ito and Johnson, 2017) subset. As baselines, we use the FastSpeech2-LR models (Ren et al., 2020; Pine et al., 2022) trained

	FastSpeech2-LR			SPEAR-TTS	Ground-truth
Parallel training data	15 min	1 h	24 h	15 min	–
MOS	$1.72_{\pm 0.04}$	$2.08_{\pm 0.04}$	$2.11_{\pm 0.04}$	$\mathbf{4.96}_{\pm 0.02}$	$4.92_{\pm 0.04}$

Table 6: **Mean Opinion Score (MOS) evaluation.** All compared systems are trained on subsets of LJSpeech (Ito and Johnson, 2017). ± indicates 95% CI obtained by bootstrap.

on 15 minutes, 1 hour, and 24 hour subsets of LJSpeech.

To ensure that the evaluation sentences are not part of the training set of SPEAR-TTS or the FastSpeech2-LR models, we extract sentences from an audiobook chapter released in 2022, read by the same voice as in LJSpeech.[6] This chapter was published later than any of the datasets we use. We extract 20 sentences from it, each with duration between 3 and 11 seconds, for a total of 133 seconds. We take transcripts for those sentences in the text of the corresponding book.

The baselines are TTS systems trained to generate a single voice. To ensure a fair comparison, we prompt $\mathcal{S}_2$ with utterances extracted from the LJSpeech dataset, so that SPEAR-TTS generates speech with the same voice. To this end, we randomly select 3 second speech samples from LJSpeech and filter out samples that have more than 1 second of silence, using the remaining as prompts.

We recruited raters using an internal crowdsourcing platform. The raters took an English proficiency test. Samples are presented to raters one-by-one, and raters are asked to judge the overall quality on a scale from Poor (1) to Excellent (5). Before starting, the raters were provided with example utterances for each grade. Those reference samples represent clean natural speech, speech resynthesized by SoundStream, and speech synthesized from partially corrupted SoundStream token sequences. We additionally included some examples representing speech degradations common in online communication. Each audio sample is evaluated by 20 raters. For each treatment, we average all scores to compute the Mean Opinion Score (MOS).

Table 6 reports the results of the subjective tests. We observe that SPEAR-TTS achieves considerably higher quality than the baselines, even when the latter use more parallel data during training. The MOS score achieved by SPEAR-TTS (4.96) is comparable to the one obtained for the ground-truth speech (4.92), confirming the high quality of the generated speech, despite the fact that the model was trained only on 15 minutes of parallel data.

System	VALL-E	SPEAR-TTS (15 min)
MOS	$3.35_{\pm 0.12}$	$\mathbf{4.75}_{\pm 0.06}$

Table 7: **Mean Opinion Score (MOS) evaluation for prompted generation.** Prompts for both systems and samples for VALL-E are taken from the demo page of VALL-E. ± indicates 95% CI obtained by bootstrap.

We also compare SPEAR-TTS and VALL-E (Wang et al., 2023) in a small-scale subjective test using the examples provided on its demo page.[7] These examples are generated by combining 8 transcripts with 3 prompts each, resulting in 24 speech utterances. Using the same instance of SPEAR-TTS described above (with $\mathcal{S}_1$ trained with 15 minutes of single-speaker LJSpeech), we synthesize 24 utterances using the same transcripts and prompts and conduct a subjective test with the same protocol described above. Table 7 shows that, on these examples, SPEAR-TTS achieves considerably better naturalness and higher speech quality (MOS 4.75) than VALL-E (3.35), despite using considerably less supervision (15 min of parallel data & 1 speaker vs. approximately 60,000 hours of parallel data spoken by over 7,000 speakers).

11 Limitations and Broader Impact

While our motivation is to enable high-quality, diverse, and controllable TTS for under-served languages, we started our investigations with English, which allowed us to address the problem

[6] https://librivox.org/predecessors-of-cleopatra-by-leigh-north/, §10.

[7] https://www.microsoft.com/en-us/research/project/vall-e-x/vall-e/, "More Samples".

using a collection of well-studied datasets. This leaves some unknowns on whether our proposed approach would be applicable for languages sufficiently different from English, e.g., for tonal languages. For instance, would we be able to get a useful discretized representation of speech for such languages? Would they require a dramatically larger token vocabulary size and would it allow effective speech modeling? However, we are hopeful due to an increasing amount of progress in related research. For example, it was demonstrated that the underlying AudioLM framework successfully works across many languages and even for music (Borsos et al., 2023; Agostinelli et al., 2023; Rubenstein et al., 2023). However, this remains circumstantial evidence and a direct investigation on SPEAR-TTS is in order.

Another potential caveat is that while training the $\mathcal{S}_2$ does not require parallel text-audio data, it still needs a considerable amount of data, in the order of a few thousands hours (see Appendix A). That can become a limitation for low-resource languages. A similar problem can be faced when training a w2v-BERT model. Here, a natural potential solution would be leveraging cross-lingual transfer, particularly from phonetically close languages, which was shown to work remarkably well for some self-supervised models (Riviere et al., 2020).

We also acknowledge that the ability to mimic a voice can have numerous malicious applications, including bypassing biometric identification and for the purpose of impersonation (Delgado et al., 2021; Casanova et al., 2022). Thus it is crucial to put in place safeguards against the misuse and, as an initial step, we verify that speech produced by SPEAR-TTS can be reliably detected by a classifier with an accuracy of 82.5% on a balanced dataset, using the same protocol and classifier as Borsos et al. (2023).

Due to the concerns about a potential malicious use, we decided not to release checkpoints and training code for our models publicly.

12 Conclusions & Future Work

We introduced SPEAR-TTS, a multi-speaker TTS system that has two features setting it apart. First, it only requires a minimal amount of parallel data to be trained, i.e., it can synthesize speech with high fidelity when trained on as little as 15 minutes of parallel data coming from a single speaker. Second, SPEAR-TTS is able to synthesize speech maintaining voice characteristics of a previously unseen speaker using a 3-second long voice example.

These capabilities originate from harnessing abundant audio-only data. The key component that unlocks the usage of such data is the hierarchical discrete representation of speech that combines high-level semantic tokens with low-level acoustic tokens. Using these representations, SPEAR-TTS casts the TTS problem as a composition of two sequence-to-sequence tasks, ‘‘reading’’ (from tokenized text to semantic tokens) and ‘‘speaking’’ (from semantic tokens to acoustic tokens).

SPEAR-TTS uses audio-only data in three ways: (a) to train the ‘‘speaking’’ model, such that the hard task of speech generation benefits from large-scale data, (b) as a domain for pretraining, and (c) to generate synthetic parallel data for backtranslation.

Our experimental study on English data (§ 9) shows that by combining audio-only data from LibriTTS (Zen et al., 2019) with 15 minutes of parallel data sampled from LJSpeech (Ito and Johnson, 2017), SPEAR-TTS achieves intelligibility comparable to that of an adapted FastSpeech2-LR (Pine et al., 2022) trained on 24 hours of LJSpeech.

Next, our experiments show that SPEAR-TTS can maintain voice characteristics of a previously unseen speaker, in a zero-shot manner, with high accuracy. Indeed, our measurements indicate that by taking a 3-second-long voice example for a speaker from LibriSpeech test-clean, SPEAR-TTS achieves 92.4% accuracy on maintaining the voice when synthesizing held-out text transcripts, according to our speaker classifier. Moreover, when measuring speaker similarity between prompts and generated speech, SPEAR-TTS obtains a cosine similarity close to the score reported for VALL-E (Wang et al., 2023) and significantly higher than the score of YourTTS (Casanova et al., 2022).

Subjective evaluations of speech naturalness show that SPEAR-TTS has significantly higher quality than a strong single-voice baseline even when trained with 96× less parallel data. Moreover, the MOS score of SPEAR-TTS is on par with the natural speech. When comparing quality of the speech synthesized in a zero-shot voice transfer task, SPEAR-TTS obtains a MOS that is

considerably higher than VALL-E, with 240,000× less data.

We believe that applying our findings to building a TTS system for truly low-resource languages is the main direction for further work.

Acknowledgments

The authors are grateful to Ron Weiss and Matthieu Geist for their feedback on a draft of this paper. We also thank Aidan Pine for helping us to obtain and run checkpoints from Pine et al. (2022). We would also like to thank the TACL reviewers and editors for their thorough reviews and constructive feedback.

References

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. MusicLM: Generating music from text. *arXiv preprint arXiv:2301.11325*.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. 2022. SpeechT5: Unified-modal encoder-decoder pre-training for spoken language processing. In *TACL*. https://doi.org/10.18653/v1/2022.acl-long.393

James Betker. 2022. TorToiSe text-to-speech.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. 2023. AudioLM: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. https://doi.org/10.1109/TASLP.2023.3288409

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *NeurIPS*.

Edresson Casanova, Julian Weber, Christopher D. Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A. Ponti. 2022. YourTTS: Towards zero-shot multi-speaker TTS and zero-shot voice conversion for everyone. In *ICML*.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. 2022. WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE JSTSP*, 16(6). https://doi.org/10.1109/JSTSP.2022.3188113

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. 2021. W2V-Bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *ASRU*. https://doi.org/10.1109/ASRU51503.2021.9688253

Héctor Delgado, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, Xuechen Liu, Andreas Nautsch, Jose Patino, Md Sahidullah, Massimiliano Todisco, Xin Wang, Junichi Yamagishi. 2021. Asvspoof 2021: Automatic speaker verification spoofing and countermeasures challenge evaluation plan. *arXiv preprint arXiv:2109.00535*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*. https://doi.org/10.18653/v1/D18-1045

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Keith Ito and Linda Johnson. 2017. The LJ speech dataset. https://keithito.com/LJ-Speech-Dataset/.

Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A. Patterson. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Annual International Symposium on Computer Architecture*. https://doi.org/10.1145/3579371.3589350

Jacob Kahn, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, Tatiana Likhomanenko, Gabriel Synnaeve, Armand Joulin, Abdelrahman Mohamed, and Emmanuel Dupoux. 2020. Libri-Light: A benchmark for ASR with limited or no supervision. In *IEEE ICASSP*. https://doi.org/10.1109/ICASSP40776.2020.9052942

Eugene Kharitonov, Ann Lee, Adam Polyak, Yossi Adi, Jade Copet, Kushal Lakhotia, Tu Anh Nguyen, Morgane Riviere, Abdelrahman Mohamed, Emmanuel Dupoux, and Wei-Ning Hsu. 2022. Text-free prosody-aware generative spoken language modeling. In *ACL*. https://doi.org/10.18653/v1/2022.acl-long.593

Heeseung Kim, Sungwon Kim, and Sungroh Yoon. 2021. Guided-TTS: Text-to-speech with untranscribed speech. *arXiv preprint arXiv: 2111.11755*.

Sungwon Kim, Heeseung Kim, and Sungroh Yoon. 2022. Guided-TTS 2: A diffusion model for high-quality adaptive text-to-speech with untranscribed data. *arXiv preprint arXiv:2205 .15370*.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *NeurIPS*.

Felix Kreuk, Adam Polyak, Jade Copet, Eugene Kharitonov, Tu-Anh Nguyen, Morgane Rivière, Wei-Ning Hsu, Abdelrahman Mohamed, Emmanuel Dupoux, and Yossi Adi. 2021. Textless speech emotion conversion using decomposed and discrete representations. *arXiv preprint arXiv:2111.07402*. https://doi.org/10.18653/v1/2022.emnlp-main.769

Kushal Lakhotia, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, Emmanuel Dupoux. 2021. On generative spoken language modeling from raw audio. *TACL*.

Alon Levkovitch, Eliya Nachmani, and Lior Wolf. 2022. Zero-shot voice conditioning for denoising diffusion TTS models. *arXiv preprint arXiv:2206.02246*. https://doi.org/10.21437/Interspeech.2022-10045

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*. https://doi.org/10.18653/v1/2020.acl-main.703

Alexander H. Liu, Cheng-I Jeff Lai, Wei-Ning Hsu, Michael Auli, Alexei Baevski, and James Glass. 2022. Simple and effective unsupervised speech synthesis. *arXiv preprint arXiv: 2204.02524*.

Alexander H. Liu, Tao Tu, Hung-yi Lee, and Lin-shan Lee. 2020. Towards unsupervised speech recognition and synthesis with quantized speech representation learning. In *ICASSP*.

Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoît Sagot, Abdelrahman Mohamed, and Emmanuel Dupoux. 2023. Generative spoken dialogue language modeling. *TACL*. https://doi.org/10.1162/tacl_a_00545

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. LibriSpeech: An ASR corpus based on public domain audio books. In *ICASSP*. https://doi.org/10.1109/ICASSP.2015.7178964

Kyubyong Park and Jongseok Kim. 2019. g2p. https://github.com/Kyubyong/g2p.

Aidan Pine, Dan Wells, Nathan Brinklow, Patrick Littell, and Korin Richmond. 2022. Requirements and motivations of low-resource speech synthesis for language revitalization. In

ACL. https://doi.org/10.18653/v1/2022.acl-long.507

Adam Polyak, Yossi Adi, Jade Copet, Eugene Kharitonov, Kushal Lakhotia, Wei-Ning Hsu, Abdelrahman Mohamed, and Emmanuel Dupoux. 2021. Speech resynthesis from discrete disentangled self-supervised representations. In *Interspeech*. https://doi.org/10.21437/Interspeech.2021-475

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21.

Chandan K. A. Reddy, Vishak Gopal, and Ross Cutler. 2021. DNSMOS: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP*.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. FastSpeech 2: Fast and high-quality end-to-end text to speech. In *ICLR*.

Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. 2020. Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7414–7418. IEEE. https://doi.org/10.1109/ICASSP40776.2020.9054548

Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirn, Dirk Padfield, James Qin, Danny Rozenberg, Tara Sainath, Johan Schalkwyk, Matt Sharifi, Michelle Tadmor, Ramanovich, Marco Tagliasacchi, Alexandru Tudor, Mihajlo Velimirovic, Damien Vincent, Jiahui Yu, Yongqiang Wang, Vicky Zayats, Neil Zeghidour, Yu Zhang, Zhishuai Zhang, Lukas Zilka, and Christian Frank. 2023. AudioPaLM: A large language model that can speak and listen. *arXiv preprint arXiv: arXiv:2306.12925*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL*. https://doi.org/10.18653/v1/P16-1009

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*.

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu. 2018. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In *ICASSP*. https://doi.org/10.1109/ICASSP.2018.8461368

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *NAACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS*.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. https://doi.org/10.1109/TASLP.2021.3129994

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. 2019. LibriTTS: A corpus derived from LibriSpeech for text-to-speech. *arXiv preprint arXiv:1904.02882*. https://doi.org/10.21437/Interspeech.2019-2441

Downsample factor	1	2	5	10
CER (%)	1.99	1.99	2.36	2.92

Table 8: **CER of SPEAR-TTS on LibriSpeech dev-clean vs. $\mathcal{S}_2$ training data size.** We measure how downsampling LibriLight (Kahn et al., 2020) before training $\mathcal{S}_2$ affects the CER (%).

Appendices

A Influence of the Data Size on $\mathcal{S}_2$

In this experiment, we measure how sensitive $\mathcal{S}_2$ is to the amount of data used to train it. To this end, we downsample LibriLight (Kahn et al., 2020) by factors of 1, 2, 5, and 10 before training $\mathcal{S}_2$ models. All models share the same architecture and are trained for the same number of updates and we select the checkpoint with the highest validation accuracy. Next, we combine the selected checkpoints with $\mathcal{S}_1$ trained on LibriTTS (Zen et al., 2019) (with pretraining) and measure intelligibility of SPEAR-TTS on LibriSpeech dev-clean. We report results in Table 8. We notice that reducing the data size 5x starts to affect the performance.

B Computational Cost

The cost of training SPEAR-TTS is dominated by the pretraining phase: Fitting the pretrained model took approximately 100 TPU-days with TPUv4 (Jouppi et al., 2023). For a comparison, the second most computationally demanding job, finetuning on the backtranslated data, required 20 TPU-hours. At the same time, the high computational cost of the pretraining can be offset by using the prepared model for multiple tasks.

C Speaker Classifier

We use the same speaker classifier as Borsos et al. (2023), which is a convolutional network that takes log-mel spectrograms as its input. The spectrograms are calculated with a window size of 25ms, a hop length of 10ms, and have 64 mel bins. The network contains 6 blocks, each cascading convolutions with kernels of 3×1 and 1×3. Each block is followed by a ReLU non-linearity and batch normalization (Ioffe and Szegedy, 2015). The per-block numbers of channels are [64, 128, 256, 256, 512, 512]. The classifier has an input span of 1 second and, to classify a longer utterance, we run a sliding window with a hop length of 250 ms and average predictions across the windows.