

# TTS-Transducer: End-to-End Speech Synthesis with Neural Transducer

Vladimir Bataev\*, Subhankar Ghosh\*, Vitaly Lavrukhin, Jason Li

NVIDIA

{vbataev, subhankarg, vlavrukhin, jasoli}@nvidia.com

**Abstract**—This work introduces TTS-Transducer – a novel architecture for text-to-speech, leveraging the strengths of audio codec models and neural transducers. Transducers, renowned for their superior quality and robustness in speech recognition, are employed to learn monotonic alignments and allow for avoiding using explicit duration predictors. Neural audio codecs efficiently compress audio into discrete codes, revealing the possibility of applying text modeling approaches to speech generation. However, the complexity of predicting multiple tokens per frame from several codebooks, as necessitated by audio codec models with residual quantizers, poses a significant challenge. The proposed system first uses a transducer architecture to learn monotonic alignments between tokenized text and speech codec tokens for the first codebook. Next, a non-autoregressive Transformer predicts the remaining codes using the alignment extracted from transducer loss. The proposed system is trained end-to-end. We show that TTS-Transducer is a competitive and robust alternative to contemporary TTS systems<sup>1</sup>.

**Index Terms**—TTS, RNNT, Neural Transducers

## I. INTRODUCTION

Neural text-to-speech (TTS) is a sequence-to-sequence task where the model learns to generate a speech sequence conditioned on the input text sequence. TTS synthesis is monotonic, preserving the order between input text and output speech. Since speech is produced at the frame level, one phoneme can correspond to multiple frames, and output length varies with the speaker’s style, making text-to-speech alignment challenging. Non-autoregressive (NAR) TTS models [1], [2] use explicit phoneme or text token duration predictor. In autoregressive (AR) encoder-decoder [3] TTS models [4], alignment is learned implicitly, while they produce more natural speech, they also suffer from hallucination, skipping or repeating words [5].

The transducer architecture (RNNT) [6], widely used in automatic speech recognition (ASR), enforces a monotonic alignment constraint. Thus, it could provide a robust solution for this problem. However, direct application of transducers to TTS is challenging since transducers are designed to predict discrete units, but speech is typically represented in continuous form, e.g., with a mel-spectrogram. Recent development in neural audio codecs [7]–[9] allows to transform the audio prediction task into a discrete units prediction task. This approach significantly simplifies TTS pipeline, and many recent

state-of-the-art TTS models follow this approach, e.g., VALLE [10], T5-TTS [11], Bark [12], SpeechX [13]. Using discrete speech codes as a target makes the transducer a natural fit for TTS alignment. However, audio codecs produce *multiple codes per frame*, and directly applying RNNT to predict all the codes sequentially requires a huge amount of memory to compute the loss, since memory complexity depends on the product of the input and target sequence lengths. We address this challenge by introducing a two-component architecture that is trained end-to-end. *The transducer component* predict the first codebook codes. *A residual codebook head (RCH)* part iteratively predicts the remaining codes using the aligned encoder output and the predicted previous codebook codes. The components are conditioned on a speaker embedding, generated from the target speaker’s sample speech using Global Style Tokens (GST) [14]. The model is based on the Transformer [15] architecture.

The key contributions of our work are as follows:

- 1) A novel end-to-end TTS-Transducer model based on the neural transducer architecture that predicts audio codes directly from the tokenized text, solving the problem of producing multiple codes per frame.
- 2) TTS-Transducer achieves 3.94% character error rate (CER) on challenging texts, surpassing larger state-of-the-art TTS models trained on significantly more data.
- 3) We demonstrate that TTS-Transducer is codec agnostic and is able to generalize well across different residual vector quantization (RVQ) codecs, which is a rapidly growing field.

TTS-Transducer generates high-quality speech and achieves zero-shot results comparable to state-of-the-art TTS models without pretraining on large data. We will open-source our code in the NeMo toolkit [16].

## II. BACKGROUND AND RELATED WORK

### A. Neural Transducers

Recurrent neural network transducer [6] is a universal architecture for sequence-to-sequence tasks requiring the monotonic alignment between input and output. Transducer consists of three neural modules: (1) *encoder*, which processes the input sequence (originally audio features) and generates high-level representations, (2) *prediction network (predictor)* - an autoregressive network (originally RNN) that uses previously predicted tokens to generate the next output, starting from

\*Equal contribution

<sup>1</sup>Audio Examples are available at <https://tts-transducer.github.io>

special  $\langle SOS \rangle$  (start-of-sequence) token, and (3) *joint network*, which combines outputs of encoder step  $i$  and prediction network step  $j$  to produce the distribution of the probabilities ( $p_{i,j}$ ) over the vocabulary augmented with special  $\langle blank \rangle$  ( $\langle b \rangle$ ) symbol. RNNT optimizes the total probability of all possible alignments between input and output sequences, where the target sequence includes inserted  $\langle blank \rangle$  symbols, acting as delimiters between frames.

There are several decoding strategies for transducers. Greedy decoding uses nested loop design, where the outer loop iterates over frames of the encoder output, and the inner loop retrieves labels one by one with the maximum probability by combining the encoder output for the current frame and prediction network output using joint network until the  $\langle blank \rangle$  symbol is found. In our experiments, we apply nucleus sampling [17] using predicted probabilities instead of greedy selection.

### B. Neural Transducers for Text-to-Speech Synthesis

Segment-to-Segment Neural Transduction (SSNT) [18] introduced transducer-based monotonic restrictions in TTS but factorized joint probability into alignment transition probability and emission probability for acoustic features. Speech-T model [19] also decouples aligning and mel-spectrogram prediction. The authors use a modified RNNT loss to learn the alignment, but the model requires the external forced aligner to construct diagonal constraints in the probability lattice to help the transducer learn the alignment.

Recently introduced *Transduce-and-Speak* [20] model has two components trained separately. The first component is a neural transducer generating "semantic" tokens (one token per frame). Such tokens are obtained from the clustered output of pretrained Wav2Vec 2.0 [21]. The second component is a modified non-autoregressive (NAR) VITS [22] model synthesizing speech from the semantic tokens. A similar two-stage approach with intermediate "semantic" tokens was used in [23] with the second component predicting audio codes.

More recently, the VALL-T [24] combined transducer with a decoder-only Transformer. The authors combined relative position embeddings with absolute positional embeddings, where a relative position of 0 specifies the current phoneme under synthesis. The  $\langle blank \rangle$  symbol in the output indicates a position shift. During training, the model uses all possible shifts of the positional embeddings, and the output is combined to form the transducer lattice. Transducer loss is used to optimize the model. VALL-T needs a large amount of memory and multiple forward passes during training due to all the relative position shifts.

### C. TTS using Audio Codes

Recently, a new TTS approach has emerged, where TTS is considered a language modeling task that translates text input to discrete audio tokens. Similar to large language models (LLMs), there are two main types of models: (1) decoder-only (AudioLM, VALL-E, UniAudio, Bark, SpeechX [10],

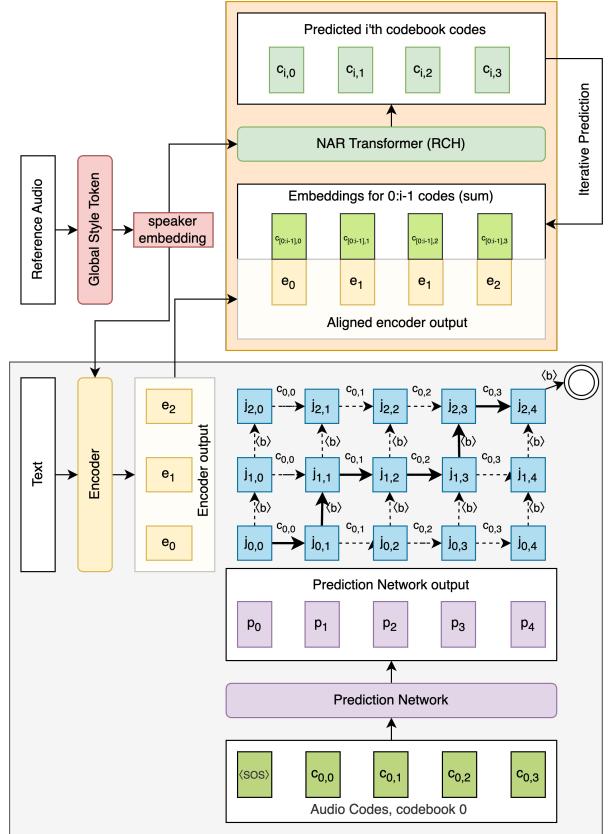


Fig. 1: TTS-Transducer model architecture

[12], [13], [25], [26]) and (2) encoder-decoder (e.g. Speech-T5 [27]). Typically audio codecs use an encoder-decoder architecture, e.g. EnCodec, SoundStream, Descript Audio Codec (DAC) [7], [8], [28]. The encoder converts audio to a hidden representation, which the quantization layer compresses using the RVQ approach. The decoder synthesizes audio in the time domain from the quantized representation. The entire model is trained end-to-end using reconstruction loss and perceptual loss from a discriminator.

## III. TTS-TRANSDUCER MODEL

The architecture of the TTS-Transducer, inspired by the VALL-E [10] system, consists of two components trained separately. The first component of VALL-E is an autoregressive Transformer model that predicts codes of the first codebook given the input text and a prompt. The second one is a non-autoregressive Transformer that predicts codes from all other residual codebooks based on the prediction of the first component, input text, and a prompt.

TTS-Transducer schema is shown in Fig. 1. We use a neural transducer to predict the codes of the first codebook given the text units by learning alignment between text and audio. The second component, residual codebook head (RCH), is a non-autoregressive Transformer. It predicts the remaining audio codes iteratively, given all previously predicted codes along with the aligned encoder output. The encoder of the transducer and the residual codebook head are conditioned on speaker

embeddings. After predicting all the codes, the decoder of the audio codec model is used to produce audio.

*Prediction of the first codebook*  $c_{0,i}$  is learned by a neural transducer. Encoder is a non-autoregressive Transformer [15] model, which transforms tokenized text  $t_i$  to the sequence of vectors  $e_i$ . We train models with Byte-Pair Encoding (BPE) [29] tokenization, and also experiment with phonemes from International Phonetic Alphabet (IPA). We also add speaker embedding conditioning to the encoder using conditional LayerNorm [30]. The prediction network is an autoregressive Transformer-Decoder, which transforms a sequence of audio codes  $c_{0,i}$  with prepended  $\langle S O S \rangle$  symbol to the sequence of vectors  $p_j$ . For each combination of vectors  $e_i, p_j$  the joint network is applied:  $j_{i,j} = \text{Softmax}(\text{Linear}(\text{ReLU}(e_i + p_j)))$ . The output of the joint network is the probability distribution for the tokens of the first codebook augmented with the  $\langle \text{blank} \rangle$  symbol.

*To predict all residual codebooks* from 1 to  $n$ , we use a non-autoregressive Transformer-encoder, which predicts  $i$ -th codebook codes using previously predicted  $[0 \dots i - 1]$  codebooks and the aligned encoder output. The input is the sum of embeddings for previously predicted codes  $c_{0:i-1,j}$  concatenated with the corresponding encoder vector  $e_k$ . We use speaker conditioning similar to the first component of our system.

*To represent speakers*, TTS systems typically use fixed embeddings from a speaker verification model [31], but these embeddings do not generalize well beyond seen speakers. So we use Global Style Tokens (GST) [14] to capture the style of the speaker as used in [32]. In our work, we convert target speaker’s reference speech to mel-spectrogram and feed it to the speaker representation module. The speaker representation module consists of a convolutional recurrent neural network-based encoder that learns the style tokens. A multi-head attention layer combines the learned style tokens to give the speaker embedding.

*We train our system end-to-end.* On each training step, we first perform a forward pass for the first component of our system. We use WFST-based implementation of the RNNT loss [33] in the k2 framework [34]. This allows us to extract the alignment between audio codes and encoder output (corresponding to text units) from the calculated RNNT lattice<sup>2</sup>. We distribute the encoder frames according to the extracted alignment. We also randomly select  $i$  from all residual codebooks  $[1 : n]$  to predict  $i+1$  codebook codes with the second part of our system. We optimize this component by applying cross-entropy loss ( $\lambda_{CE}$ ). The total loss is the weighted sum of the losses from the first and second components of our network:  $\lambda_{total} = (1 - \alpha) * \lambda_{RNNT} + \alpha * \lambda_{CE}$ . We use  $\alpha = 0.4$  in our experiments.

*In decoding*, we first evaluate the RNNT component, getting the predictions for the first codebook. Due to the nature of the decoding algorithm as described in Section II-A, getting

<sup>2</sup>To extract the alignment, we need to find the best path with maximum probability, which can be done with k2.shortest\_path.

the alignment along with the predictions does not imply computational overhead. For efficient decoding, we adopt label-looping [35] greedy decoding algorithm, replacing greedy label selection with nucleus sampling [17] on each step. Since our prediction network is a non-autoregressive Transformer, we use a key-value cache to speed up the decoding. After this step, for the remaining codebooks from  $i$  to  $n$ , we iteratively evaluate our system’s second component, utilizing the aligned encoder output. It is worth noting that the system can be naturally used in streaming mode by using an autoregressive Transformer as the second component, but we leave this for further work.

## IV. EXPERIMENTS

### A. Datasets

We use LibriTTS-R [37] dataset, an improved 24 kHz version of the LibriTTS [38]. The LibriTTS corpus contains a diverse set of speakers reading English audiobooks. We train the model on train-clean-100, train-clean-360, and train-other-500 subsets of the LibriTTS-R. We set aside 1.15 hours of data from the train-clean-100 to test the model on seen speakers. We filter the data by a maximum duration of 15 seconds, which results in 464 hours in the training dataset. To evaluate our system on *unseen speakers*, we randomly choose 0.35 hours of data for unseen speakers with 39 unseen speakers from the dev-clean subset. Additionally, we evaluate our model in *out-of-domain* conditions on a subset of 200 utterances (0.2 hours) from VCTK [39] to test the generalization abilities to multiple acoustic conditions. We use reference audios of length between 3 to 5 seconds.

### B. Model Details

We use pretrained neural audio codecs: EnCodec [7] (8 codebooks, 6 kbps), NeMo-Codec [9] with RVQ (8 codebooks, 6.9 kbps), and Descript Audio Codec [8] model (9 codebooks, 8 kbps). Our TTS-Transducer model has 12 Transformer layers in the encoder, 6 layers in the prediction network, and 12 layers in residual codebook head. The encoder and residual codebook head have 2 attention heads and a feed-forward dimension of 1536, with model dimension of 640 and 512, respectively. The prediction network Transformer-decoder uses 4 attention heads, with a model dimension of 512 and a feed-forward dimension of 2048. This results in *199M* parameters for EnCodec and NeMo-Codec, and *200M* parameters for the DAC model due to a larger embedding table since DAC codec uses 9 codebooks. For the speaker embedding model, we use 1024 640-dimensional GST [14] learnable embeddings.

All TTS-Transducer models are trained in NeMo [16] toolkit with a global batch of 2048 for 200 epochs using 32 NVIDIA A100 GPUs. We use AdamW [40] optimizer with cosine annealing scheduler [41] with 2000 warmup steps and a maximum learning rate of  $1e-3$ .

During inference, we use nucleus sampling [17] ( $p = 0.95$ ) for predicting codes from the first codebook. The second component predicts remaining codebooks greedily.

TABLE I: Automatic evaluation, large models with different codecs and tokenization. *Seen speakers*: LibriTTS-R held-out set from train-clean-100. *Unseen speakers*: 200 utterances from LibriTTS-R dev-clean. *Out-of-domain*: 200 utterances from the VCTK dataset.

Codec	Tokens	Seen Speakers			Unseen Speakers			Out-of-Domain		
		WER,% $\downarrow$	CER,% $\downarrow$	SSIM $\uparrow$	WER,% $\downarrow$	CER,% $\downarrow$	SSIM $\uparrow$	WER,% $\downarrow$	CER,% $\downarrow$	SSIM $\uparrow$
Ground Truth	-	2.70	0.93	-	2.42	0.87	-	1.02	0.44	-
DAC	BPE	6.66	3.15	0.903	7.52	3.77	0.876	3.86	2.13	0.762
NeMo-Codec	BPE	4.86	2.19	<b>0.908</b>	7.19	4.45	<b>0.881</b>	4.81	2.40	0.773
EnCodec	BPE	4.90	2.18	0.903	6.25	3.01	0.868	3.93	1.79	0.759
DAC	IPA	4.04	1.67	0.900	4.89	2.13	0.866	4.41	1.96	<b>0.775</b>
NeMo-Codec	IPA	<b>3.36</b>	<b>1.31</b>	0.906	<b>4.56</b>	<b>2.11</b>	0.871	<b>3.05</b>	<b>1.48</b>	0.765
EnCodec	IPA	3.73	1.43	0.901	4.61	2.50	0.853	3.93	1.75	0.754

TABLE II: Impact of the model size. EnCodec codec, BPE tokens. *Seen speakers*: LibriTTS-R held-out set from train-clean-100. *Unseen speakers*: 200 utterances from dev-clean.

Encoder	Num Layers	Precitor	RCH	Seen Speakers		Unseen Speakers	
				WER,% $\downarrow$	SSIM $\uparrow$	WER,% $\downarrow$	SSIM $\uparrow$
6	3	6	6	6.66	0.900	6.93	0.875
6	6	6	6	6.04	0.899	6.74	0.875
12	6	6	6	5.56	0.900	6.40	0.868
6	6	12	12	5.42	0.896	6.46	<b>0.877</b>
12	6	12	12	<b>4.90</b>	<b>0.903</b>	<b>6.25</b>	0.868

TABLE III: Challenging Texts Evaluation. Comparison of best TTS-Transducer models with external LLM-based TTS models. Naturalness MOS, 95% confidence interval. Randomly selected male and female speakers, 92 utterances for each (184 total).

Model	WER,% $\downarrow$	CER,% $\downarrow$	MOS $\uparrow$
Bark [12]	22.92	11.67	$3.82 \pm 0.04$
VALL-E-X [36]	19.25	7.96	$3.72 \pm 0.04$
SpeechT5 [27]	16.24	6.00	<b>3.84 ± 0.04</b>
Ours, EnCodec, BPE	17.28	5.66	$3.83 \pm 0.04$
Ours, NeMo-Codec, BPE	16.87	5.37	$3.81 \pm 0.04$
Ours, EnCodec, IPA	15.64	4.50	$3.69 \pm 0.04$
Ours, NeMo-Codec, IPA	<b>13.83</b>	<b>3.94</b>	$3.82 \pm 0.04$

### C. Results

We use automatic evaluation metrics – Character Error Rate (CER), Word Error Rate (WER) and Speaker Similarity (SSIM) – to compare our models with different audio codecs and tokenization. To test robustness, we transcribe generated audio using the pretrained Parakeet-CTC-1.1B<sup>3</sup> and compute CER and WER with respect to the ground truth transcriptions. For SSIM, we compute cosine similarity between embeddings of generated speech and ground truth audio obtained from WavLM [42] model<sup>4</sup>. The results are shown in Table I. All the models provide intelligible speech with a relatively low word error rate. This shows that our model can produce robust speech and preserve the target speaker’s acoustic qualities, regardless of the audio codec being used. Most of the BPE-based models show best speaker similarity scores, but all models using IPA show significantly better intelligibility.

We perform ablation studies for BPE-based models varying the number of layers in encoder, predictor and residual code-

book head. The results in Table II show that all the components contribute to the intelligibility of the produced speech.

For comparison with external TTS systems, along with automatic metrics, we use Mean Opinion Score (MOS) evaluations. Each audio was rated by at least 11 independent listeners on Amazon Mechanical Turk platform. Each rater was asked the question "How natural does the speech sound?" for a given audio on a scale of 1 to 5 with 1 point difference, 1 being totally unnatural and 5 extremely natural. We choose following TTS models which use large-scale language modeling approach: VALL-E-X [10], Bark [12] with EnCodec audio codec, and also SpeechT5 [27]<sup>5</sup>. For VALL-E-X, we use unofficial open-source implementation with checkpoint<sup>6</sup> trained on 1739 hours of audio<sup>7</sup>. Bark authors do not disclose<sup>8</sup> the amount of data but claim the work is comparable<sup>9</sup> with related AudioLM [25] and VALL-E [10], both using ~60k hours of audio data. SpeechT5 uses LibriSpeech audio for pretraining (1k hours) and 400M text sentences from the text corpus and is finetuned on the LibriTTS dataset. As shown in Table III, the proposed BPE-based TTS-Transducer outperforms Bark and public VALL-E-X systems in intelligibility and is comparable with best models in naturalness. The IPA-based model with NeMo codec is also comparable in naturalness but shows the best robustness across all the compared models.

### V. CONCLUSION

We presented a novel TTS-Transducer system that predicts audio neural codec tokens directly from phonemes based on a neural transducer. It combines the strength of the neural transducer to learn monotonic alignment between text and audio and the effectiveness of neural audio codecs. The neural transducer predicts the first codebook. The remaining residual codes for the same frame are predicted by a separate block based on the learned alignment. Both components are optimized jointly. We demonstrated that our model can produce high-quality, reliable speech with popular audio codecs. Our experiments showed that the model achieves results comparable to SOTA TTS models in naturalness, and surpasses them in intelligibility on challenging texts without requiring large-scale pretraining.

<sup>5</sup>[https://hf.co/microsoft/speecht5\\_tts](https://hf.co/microsoft/speecht5_tts)

<sup>6</sup><https://github.com/Plachtaa/VALL-E-X>

<sup>7</sup><https://plachtaa.github.io/vallex/>

<sup>8</sup><https://github.com/suno-ai/bark/issues/2>

<sup>9</sup><https://github.com/suno-ai/bark/issues/277>

<sup>3</sup><https://hf.co/nvidia/parakeet-ctc-1.1b>

<sup>4</sup><https://hf.co/microsoft/wavlm-base-plus-sv>

## REFERENCES

- [1] Y. Ren *et al.*, “FastSpeech: Fast, robust and controllable text to speech,” *NeurIPS*, 2019.
- [2] S. Beliaev and B. Ginsburg, “TalkNet: Non-autoregressive depth-wise separable convolutional model for speech synthesis,” in *Interspeech*, 2021.
- [3] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *NIPS*, 2014.
- [4] J. Shen *et al.*, “Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions,” *ICASSP*, 2017.
- [5] W. Ping *et al.*, “Deep Voice 3: Scaling text-to-speech with convolutional sequence learning,” *ICLR*, 2018.
- [6] A. Graves, “Sequence transduction with Recurrent Neural Networks,” in *ICML Workshop on Representation Learning*, 2012.
- [7] A. D’efossez, J. Copet, G. Synnaeve, and Y. Adi, “High fidelity neural audio compression,” *arXiv:2210.13438*, 2022.
- [8] R. Kumar *et al.*, “High-fidelity audio compression with improved RVQGAN,” in *NeurIPS*, 2023.
- [9] R. Langman *et al.*, “Spectral codecs: Spectrogram-based audio codecs for high quality speech synthesis,” *arXiv:2406.05298*, 2024.
- [10] C. Wang *et al.*, “Neural codec language models are zero-shot text to speech synthesizers,” *arXiv:2301.02111*, 2023.
- [11] P. Neekhara, S. Hussain, S. Ghosh, J. Li, and B. Ginsburg, “Improving robustness of LLM-based speech synthesis by learning monotonic alignment,” in *Interspeech 2024*, 2024.
- [12] Suno, “Bark: Text-prompted generative audio model,” 2024, [Online; accessed 2024-06-01]. [Online]. Available: <https://github.com/suno-ai/bark>
- [13] X. Wang *et al.*, “SpeechX: Neural codec language model as a versatile speech transformer,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.
- [14] Y. Wang, D. Stanton, Y. Zhang, R.-S. Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, “Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis,” in *ICML*, 2018.
- [15] A. Vaswani *et al.*, “Attention is all you need,” in *NeurIPS*, 2017.
- [16] O. Kuchaiev, J. Li, H. Nguyen *et al.*, “Nemo: a toolkit for building AI applications using neural modules,” in *NeurIPS Workshop on Systems for ML*, 2019.
- [17] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The curious case of neural text degeneration,” in *ICLR*, 2020.
- [18] Y. Yasuda, X. Wang, and J. Yamagishi, “Initial investigation of encoder-decoder end-to-end TTS using marginalization of monotonic hard alignments,” in *Proc. 10th ISCA Workshop on Speech Synthesis (SSW 10)*, 2019.
- [19] J. Chen *et al.*, “Speech-T: Transducer for text to speech and beyond,” in *NeurIPS*, 2021.
- [20] M. Kim *et al.*, “Transduce and Speak: Neural transducer for text-to-speech with semantic token prediction,” *Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023.
- [21] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “Wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *NeurIPS*, 2020.
- [22] J. Kim, J. Kong, and J. Son, “Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech,” in *ICML*, 2021.
- [23] J. Y. Lee, M. Jeong, M. Kim, J.-H. Lee, H.-Y. Cho, and N. S. Kim, “High fidelity text-to-speech via discrete tokens using token transducer and group masked language model,” in *Interspeech*, 2024.
- [24] C. Du *et al.*, “VALL-T: Decoder-only generative transducer for robust and decoding-controllable text-to-speech,” *arXiv:2401.14321*, 2024.
- [25] Z. Borsos *et al.*, “AudioLM: a language modeling approach to audio generation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.
- [26] D. Yang *et al.*, “UniAudio: An audio foundation model toward universal audio generation,” *arXiv:2310.00704*, 2023.
- [27] J. Ao *et al.*, “SpeechT5: Unified-modal encoder-decoder pre-training for spoken language processing,” in *Proc. of the 60th Annual Meeting of ACL (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022, pp. 5723–5738.
- [28] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, “SoundStream: An end-to-end neural audio codec,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.
- [29] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proc. of the 54th Annual Meeting of ACL*, 2016.
- [30] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv:1607.06450*, 2016.
- [31] Y. Jia *et al.*, “Transfer learning from speaker verification to multispeaker text-to-speech synthesis,” in *NeurIPS*, 2018.
- [32] C.-P. Hsieh, S. Ghosh, and B. Ginsburg, “Adapter-based extension of multi-speaker text-to-speech model for new speakers,” in *Interspeech*, 2023.
- [33] A. Laptev, V. Bataev, I. Gitman, and B. Ginsburg, “Powerful and extensible WFST framework for RNN-Transducer losses,” in *ICASSP*, 2023.
- [34] D. Povey, P. Źelasko, and S. Khudanpur, “Speech recognition with next-generation Kaldi (K2, Lhtose, Icefall),” *Interspeech*, 2021.
- [35] V. Bataev, H. Xu, D. Galvez, V. Lavrukhin, and B. Ginsburg, “Label-looping: Highly efficient decoding for transducers,” *arXiv:2406.06220*, 2024.
- [36] Z.-H. Zhang *et al.*, “Speak foreign languages with your own voice: Cross-lingual neural codec language modeling,” *ArXiv*, vol. abs/2303.03926, 2023.
- [37] Y. Koizumi *et al.*, “LibriTTS-R: A Restored Multi-Speaker Text-to-Speech Corpus,” in *Interspeech*, 2023.
- [38] H. Zen *et al.*, “Libritts: A corpus derived from librispeech for text-to-speech,” in *Interspeech*, 2019.
- [39] J. Yamagishi, C. Veaux, and K. MacDonald, “CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92),” *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2019.
- [40] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *ICLR*, 2019.
- [41] ———, “SGDR: Stochastic gradient descent with warm restarts,” in *ICLR*, 2017.
- [42] S. Chen *et al.*, “WavLM: Large-scale self-supervised pre-training for full stack speech processing,” *IEEE Journal of Selected Topics in Signal Processing*, 2022.