TensorTalk: A Voice Cloning Framework for Text-to-Speech Synthesis Using WavLM Features and Rhythm Modeling

Alexandre Dalban
Boston College
dalban@bc.edu

Robert Richenburg
Boston College
richenbr@bc.edu

Arun Raja
Boston College
rajaar@bc.edu

Alexander Sharpe
Boston College
sharpeal@bc.edu

Abstract

Voice cloning technology has traditionally required extensive training data from target speakers to produce natural-sounding speech. While recent zero-shot approaches have reduced this data requirement, they often fail to capture the natural rhythm and timing patterns that make speech sound authentic. We present a system that addresses these challenges by incorporating rhythm modeling into zero-shot voice cloning.

Building on recent work in SSL-TTS (Hajal et al., 2024) and rhythm modeling (van Niekerk et al., 2023), our system introduces two key innovations: (1) integration of explicit rhythm modeling for more natural speech synthesis, and (2) a computationally efficient alternative to GlowTTS using Google Text-to-Speech combined with WavLM feature extraction. Our approach achieves a UTMOS score of 4.27 on the LJSpeech dataset, outperforming previous models while reducing training time. Demo samples and code are available at https://github.com/robrich07/TensorTalk.

1. Introduction

While recent zero-shot voice cloning approaches have made significant strides in reducing the amount of training data needed, two key challenges remain: capturing natural speech rhythms and managing computational costs. Our work tackles both issues by building on recent advances in self-supervised learning and rhythm modeling for speech synthesis.

The rhythmic patterns in speech–including pauses, emphasis, and timing–are crucial elements of a speaker's identity that current zero-shot systems often fail to capture. The Urhythmic framework [4] demonstrated the importance and feasibility of modeling these patterns, but incorporating rhythm modeling into zero-shot voice cloning brings new technical challenges.

Additionally, state-of-the-art systems like SSL-TTS [1] rely on computationally intensive components such as GlowTTS, which requires approximately 1,600 training epochs. This computational burden creates a significant barrier to entry for researchers and developers looking to build upon or modify these systems.

We evaluate our system using standard metrics including Speaker-Encoder Cosine Similarity (SECS), UTokyo-SaruLab mean opinion score (UTMOS), word error rate (WER) and phoneme error rate (PER). Our experiments demonstrate that high-quality zero-shot voice cloning with rhythm modeling is achievable with reduced computational requirements, making voice cloning research more accessible to the broader research community.

2. Related Works

Recent advances in voice conversion and text-to-speech synthesis have laid the groundwork for our research. We examine several key works that inform our approach to zero-shot voice cloning with rhythm modeling.

WavLM (Chen et al., 2022) demonstrated the effectiveness of self-supervised learning for speech processing tasks. Their model, trained on 94k hours of diverse audio data, showed strong performance across various speech tasks including speaker verification and speech separation. WavLM's ability to capture both speaker characteristics and content information makes it particularly valuable for voice conversion applications. We leverage WavLM's robust feature extraction capabilities in our system, using it to generate speaker embeddings from both source and target audio.

kNN-VC (Baas et al., 2023) introduced a simpler yet effective approach to voice conversion using the k nearest neighbors matching of self-supervised speech representations. Their work showed that complex neural architectures are not always necessary for high-quality voice conversion - sometimes a straightforward nearest neighbor search in the feature space is sufficient. This insight influenced our decision to use k-NN matching rather than more complex approaches for speaker similarity. However, like many voice conversion systems, kNN-VC did not address the challenge of preserving speaker-specific rhythm patterns.

The Urhythmic framework (van Niekerk et al.,

2023) specifically tackled the rhythm modeling problem in voice conversion. They proposed an unsupervised method that first segments speech into sonorants, obstruents, and silences, then models rhythm by estimating speaking rate and duration distributions. While effective, their approach required integration with existing voice conversion systems. Our work builds on Urhythmic by incorporating its rhythm modeling principles into a zero-shot voice cloning pipeline.

SSL-TTS (Hajal et al., 2024) presented a framework for zero-shot multi-speaker text-to-speech synthesis using self-supervised learning. Their approach used GlowTTS for text-to-SSL (text-to-representations) conversion and demonstrated strong performance in speaker similarity. However, the computational requirements of training GlowTTS (approximately 1,600 epochs) presented a significant barrier. Our work adapts their framework but replaces GlowTTS with a more computationally efficient approach using Google Text-to-Speech and WavLM feature extraction.

Together, these works highlight both the progress and remaining challenges in zero-shot voice cloning. Although self-supervised learning and k-NN matching have proven effective for voice conversion and separate solutions exist for rhythm modeling, combining these approaches efficiently remains a challenge. Our work seeks to bridge these gaps while reducing computational requirements for improved accessibility.

3. Methods

Our system comprises several key components.

3.1. Speech Representation with WavLM

We use WavLM-Large [3] to extract speech representations, specifically from its 6th transformer layer. This choice is motivated by SSL-TTS's [1] findings that layer 6 provides an optimal balance between speaker identity and content information. Later layers tend to focus more on content and linguistic features while losing speaker characteristics, and earlier layers have not yet built up sufficient high-level representations. The sixth layer outputs 1024-dimensional feature vectors that capture both speaker identity, phonetic content, and context-related information, making them ideal for voice cloning tasks.

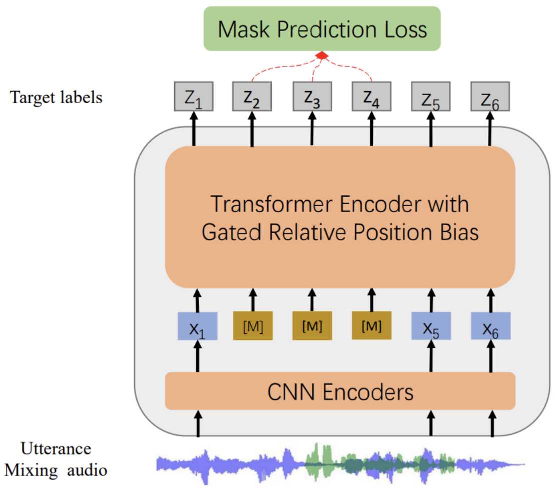

Figure 1. Diagram of WavLM-Large Architecture

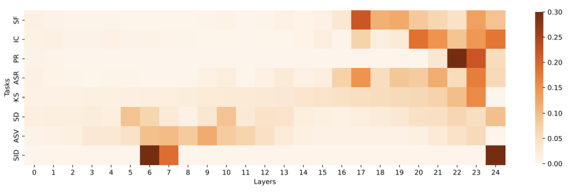

Figure 2. Diagram of weight analysis of the attention model for WavLM-Large per task. Our application focuses on the bottom three categories: ASR (Automatic Speech Recognition), SID (Speaker Identification), and SD (Speaker Diarization).

WavLM [3] processes audio using a sliding-window approach, where each feature vector represents a 25ms window of audio with a stride of 20ms between windows. This means that for a 1-second audio clip, WavLM generates 50 feature vectors (1000ms/20ms), each being 1024-dimensional. The 5ms overlap between consecutive windows helps ensure smooth transitions between frames while maintaining temporal consistency in the extracted features.

WavLM features are particularly suitable for voice conversion because they are trained through self-supervised learning (to do masked speech prediction and de-noising) on massive amounts of speech data (94k hours) [3]. This training approach allows WavLM to learn robust speech representations that generalize well across different speakers and acoustic conditions. Notably, the features exhibit a useful property where vectors that are linearly close to each other tend to share phonetic content while preserving speaker characteristics.

3.2. k-NN Retrieval with Cosine Similarity

We employ k-nearest neighbors ($k = 4$) with cosine similarity to match the characteristics of the source and target speech features. The choice of k-NN is motivated by several factors. First, as demonstrated by kNN-VC [2], simple nearest neighbor matching can be as effective as more complex neural approaches for voice conversion. Second, k-NN is non-parametric and requires no training, making it computationally efficient and easy to implement. Cosine similarity is used as our distance metric because it focuses on the angle between vectors rather than their magnitude:

$$\cos(a, b) = \frac{a \cdot b}{||a|| \cdot ||b||} \tag{1}$$

This property is particularly important for speech features where the direction of the vector often encodes more meaningful information than its magnitude. As shown in kNN-VC [2], cosine similarity effectively captures phonetic similarity while being invariant to speaker-dependent variations in feature magnitude.

```python
def KNN(self, source_features, target_features):
    synth_set = target_features

    dists = cosine_dist(source_features, target_features)
    best = dists.topk(k=4, largest=False, dim=-1)
    selected_features = synth_set[best.indices].mean(dim=1)

    return selected_features
```

Figure 3. Python Code of KNN Implementation

3.3. Linear Interpolation

We incorporate linear interpolation to provide fine-grained control over the influence of the target speaker's characteristics. Given source features and their k-nearest neighbor matched target features, we compute the converted features using:

$$y_{converted} = \lambda y_{selected} + (1 - \lambda) y_{source} \tag{2}$$

where $\lambda \in [0, 1]$ is an interpolation parameter that controls the balance between source and target characteristics. When $\lambda = 1$, the output maintains full target speaker characteristics, while $\lambda = 0$ preserves the original source features.

This interpolation serves multiple purposes. First, it provides a mechanism to control the strength of voice conversion, allowing for subtle to dramatic transformations. Second, it helps maintain intelligibility by allowing source speech characteristics to partially remain when needed. Finally, as demonstrated in kNN-VC [2], this approach can help smooth potential artifacts that might arise from direct feature replacement, particularly in cases where the k-NN matching might be imperfect.

In our implementation, we typically use $\lambda = 1$ for full voice conversion, but the parameter can be adjusted based on specific requirements or to achieve different voice mixing effects. This flexibility is particularly valuable in cases where preserving certain aspects of the source speech while adopting target speaker characteristics is desired.

3.4. Modified TTS-SSL Pipeline

While SSL-TTS uses GlowTTS to convert text directly to WavLM features, we introduce a more computationally efficient approach. Our pipeline first uses Google Text-to-Speech to generate initial audio, then processes this audio through WavLM to obtain features. This approach eliminates the need for extensive GlowTTS (Text-to-SSL) training while still providing high-quality features for matching. We named this approach TensorTalk.

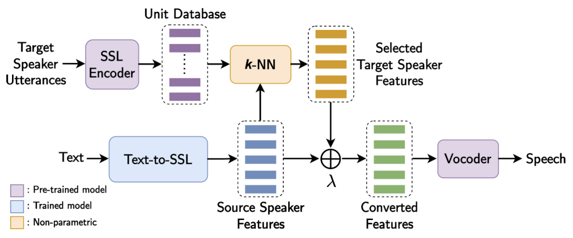

Figure 4. Original Architecture of zero-shot multi-speaker text-to-speech synthesis using self-supervised learning detailed in Hajal et al, 2024

3.5. Rhythm Modeling Integration

We incorporate rhythm modeling following the Urhythmic framework's [4] approach. The rhythm modeling process begins by analyzing Mel-Spectrograms of the target speaker's speech. These spectrograms are processed through a HuBERT Encoder [5], which uses a transformer architecture to segment the speech and generate initial feature vectors.

The system employs a dual encoding approach that combines global and local rhythm analysis. A global encoder captures sample-wide rhythmic patterns, while a local encoder focuses on temporal patterns within specific time intervals. The local feature vectors are further processed through a specialized Rhythm Encoder consisting of convolutional layers followed by dilated convolutions. This architecture allows the model to capture rhythmic patterns across increasingly broader time intervals.

The core processing steps involve segmenting speech into sonorants, obstruents, and silences, followed by modeling the duration distribution of each segment type [4]. These segments are then processed through the dual encoding system to generate rhythm-aware features through the specialized encoder.

The final stage employs a decoder that combines global rhythm features, local rhythm features, and target speaker features. This decoder, implementing both convolutional and attention mechanisms, generates a rhythm-aware Mel-spectrogram. The rhythm modeling acts as a post-processing step on the converted features before they are passed to the vocoder, ensuring that the final synthesized speech matches not only the target speaker's voice characteristics but also their temporal speaking patterns.

The rhythm-aware pipeline processes speech in five stages. First, input text is converted to speech using Google TTS. Next, WavLM [3] features are extracted from the TTS-generated speech and target speaker samples. These features are matched using k-NN with cosine similarity, which identifies the four closest target speaker feature vectors for each 25 ms segment of the source data. The k-NN replaces the source features with the average of these closest vectors, creating selected features. Linear interpolation is then applied to blend the source and selected features, producing a mix of characteristics from both speakers. The blended features are passed through a HiFi-GAN vocoder that reconstructs WavLM features into a waveform, which is further encoded into HuBERT [5] features for rhythm modeling. Temporal patterns are adjusted using the rhythm modeling, and the final HiFi-GAN vocoder reconstructs the waveform from the HuBERT features, generating output audio that incorporates both the voice and rhythm characteristics of the target speaker.

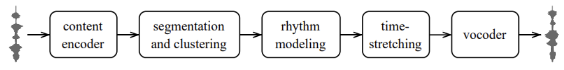

Figure 5. Urhythmic Model architecture detailed in van Niekerk, Benjamin, et al., 2023

4. Experiments

In our evaluation phase, we test two model architectures on two datasets, giving us a total of four models to analyze. The two architectures are as follows: TensorTalk (as mentioned in 3.4), and TensorTalk with rhythm modeling. The two datasets are LJSpeech and VCTK. LJSpeech consists of a singular speaker who utters 13,100 different sentences. VCTK consists of 110 English speakers who utter 400 different sentences. We use these two datasets because the urythmic component was trained using them (in the case of VCTK, it was only trained on six specific speakers out of the 110). In order to test the models, we run the forward call with one sentence from the FLoRes+ dataset and 30 wav files from the target speaker, for a total of 75 sentences. The output of each call is a generated example of the input sentence in the target speaker's voice. This process is done for each of the 7 target speakers. All tests are performed with $\lambda = 1$. After obtaining the generated examples for each speaker, we calculate the following metrics to determine the performance of the models. See Table 1 for the numerical results.

4.1. Naturalness

We evaluate the model's performance in terms of naturalness, i.e. how natural does the generated audio sound in relation to a human, using UTMOS [6]. Important to understanding UTMOS is to first understand MOS, or mean opinion score. MOS is a measure of the human-judged overall quality of an event or experience. For this project, a MOS is a measure of the quality of speech utterances. Most often judged on a scale of 1 (bad) to 5 (excellent), MOS's are the average of a number of other human-scored individual parameters. Although MOS's were originally derived from surveys of expert observers, today a MOS is often produced by an Objective Measurement Method, approximating a human ranking (like UTMOS). A value of 4.3-4.5 is considered an excellent target to shoot for due to human tendency to rarely give out perfect 5's, and below 3.5 is generally unacceptable. Thus, UTMOS is a mechanical method of calculating how natural an utterance is. In Figure 6 below, one can see a comparison of WER to UTMOS for both of our models on the LJSpeech data. Each data point represents a test of the model with a different duration of target reference utterance (how much audio was input to the model).

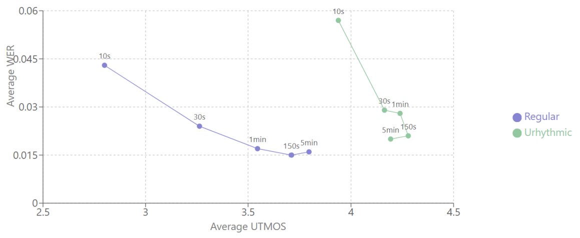

Figure 6. WER vs. UTMOS of TensorTalk on LJSpeech

4.2. Intelligibility

We evaluate the model's performance in terms of intelligibility, i.e. how accurate the generated sentence was in relation to the input sentence, using WER and PER (word and phoneme error rate). WER is the ratio of word errors in a transcript to the total words spoken. A lower WER in speech-to-text means better accuracy in recognizing speech. In our case, this would be calculated with the formula $\frac{S+D+I}{N}$, where S is the number of substitutions (instances where a word in the synthesized sentence vector would need to be substituted to match the truth vector), D is the number of deletions (instances where a word in the synthesized sentence vector would need to be deleted to match the truth vector), I is the number of insertions (instances where a new word would need to be inserted to match the truth vector), and N is the total number of words. The numerator is also known as the edit distance because it represents "how far away" two sentences are. PER is the ratio of phoneme errors in a transcript to the total phonemes spoken. As above, a lower PER means better accuracy. The formula is the same except in the context of comparing phonemes instead of words. Both of these are calculated using the Whisper-

Table 1. Comparison of models in terms of performance metrics. The **Baselines** include all of the models and their corresponding metrics from the original paper.

Model	WER ↓	PER ↓	UTMOS ↑	SECS ↑
Ground Truth	2.91 ± 0.31	0.92 ± 0.15	4.09 ± 0.01	0.87 ± 0.003
Baselines:				
YourTTS	6.09 ± 0.32	2.24 ± 0.12	3.65 ± 0.01	0.54 ± 0.003
XTTS	2.76 ± 0.21	0.84 ± 0.09	4.07 ± 0.01	0.40 ± 0.003
HierSpeech++	3.36 ± 0.23	0.78 ± 0.06	$\mathbf{4.44 \pm 0.01}$	0.67 ± 0.003
GlowTTS-SSL	3.71 ± 0.24	0.98 ± 0.07	4.02 ± 0.01	$\mathbf{0.72 \pm 0.002}$
GradTTS-SSL	4.32 ± 0.25	1.44 ± 0.09	4.16 ± 0.01	0.71 ± 0.003
Proposed:				
TensorTalk on LJSpeech	0.02 ± 0.004	0.01 ± 0.01	3.79 ± 0.03	0.57 ± 0.002
TensorTalk Urythmic on LJSpeech	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	4.27 ± 0.02	0.65 ± 0.002
TensorTalk on VCTK	0.02 ± 0.002	0.01 ± 0.002	3.87 ± 0.02	0.51 ± 0.003
TensorTalk Urythmic on VCTK	0.02 ± 0.002	0.01 ± 0.002	3.68 ± 0.016	0.53 ± 0.002

Large v3 model. For our project specifically, it is very important to note how our models have substantially lower WER and PER values than all of the other models. This is due to the fact that we used gTTS in our pipeline, which is designed to have no errors in the output sentence, at the cost of having a very robotic-sounding voice.

4.3. Speaker Similarity

We evaluate the model's performance in terms of intelligibility, i.e. how similar the target speaker's voice and output voice are, using SECS (speaker encoder cosine similarity) [6]. SECS is the cosine similarity between the embeddings of two audio samples, which in our case are a ground truth sample from one speaker and the synthesized sample for that same speaker. We use ECAPA2 to get the embeddings and the sklearn library to calculate the similarity. Cosine similarity outputs a value in $[-1, 1]$, but the original paper [6] normalized to $[0, 1]$, so we do the same. A value of 0 means the speakers are completely different, and a value of 1 means the speakers are perfectly alike. One can see that the ground truth value is 0.87 (this is obtained by comparing the embeddings of two audio clips from the same speaker). In Figure 7, one can see a comparison of WER to SECS for both of our models on the LJSpeech data. Each data point represents a test of the model with a different duration of target reference utterance.

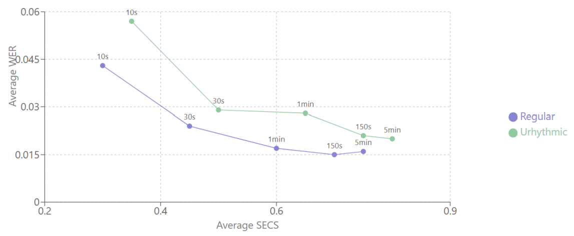

Figure 7. WER vs. SECS of TensorTalk on LJSpeech

5. Conclusions

Overall, our models had fairly differing results. For example, our model with urythmic on LJSpeech performed very competitively with the baseline models. Most notably, it achieved the second highest UTMOS score, placing it ahead of the proposed models from the original paper, GlowTTS-SSL and GradTTS-SSL. Additionally, it had the third highest SECS value. On the other hand, our model with urythmic on VCTK did not perform nearly as well, with the second lowest UTMOS score, and a lower-middle SECS value. Given our time and resources, we believe that this was ultimately a great success, as we were able to achieve very competitive results in at least one model. One of the advantages of our models is the vast decrease in computational complexity through our implementation of gTTS. While this may have lowered the performance of some metrics, it allowed us to achieve a high enough level of success with limited computing power due to Google Colab usage limits. This paves the way for future work: researchers with greater access to powerful GPUs can implement a similar technique of rhythm modeling while also using a more complex TTS model that may improve performance. Also, other students with similar resources to us can develop models using gTTS, thus allowing them to improve the pipeline in additional ways without having to worry about computational limits. One final takeaway is the urythmic component does seem to have a beneficial effect on the quality of the generated audio. We see that in all but one case (UTMOS of VCTK models), the urythmic models have higher metrics than the same model without urythmic.

6. Contributions

Alexandre Dalban wrote sections 3.1, 3.2, 3.3, and 3.4. Robert Richenburg wrote sections 2 and 5. Arun Raja wrote the abstract as well as sections 1 and 3.5. Alexander Sharpe wrote section 4. Demo samples of generated speech and code are avaliable at `https://github.com/robrich07/TensorTalk`.

References

[1] K. El Hajal, A. Kulkarni, E. Hermann, and M. Magimai-Doss, “SSL-TTS: Leveraging Self-Supervised Embeddings and kNN Retrieval for Zero-Shot Multi-speaker TTS,” *arXiv preprint arXiv:2408.10771*, 2024. [Online]. Available: https://arxiv.org/pdf/2408.10771

[2] M. Baas, B. van Niekerk, and H. Kamper, “Voice Conversion With Just Nearest Neighbors,” in *Proc. Interspeech*, 2023, pp. 2053–2057. [Online]. Available: https://arxiv.org/pdf/2305.18975

[3] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, *et al.*, “WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022. [Online]. Available: https://arxiv.org/pdf/2110.13900

[4] B. van Niekerk, M.-A. Carbonneau, and H. Kamper, “Rhythm Modeling for Voice Conversion,” *IEEE Signal Processing Letters*, vol. 30, pp. 1297–1301, 2023. [Online]. Available: https://arxiv.org/pdf/2307.06040

[5] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021. [Online]. Available: https://arxiv.org/pdf/2106.07447

[6] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K.-t. Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H.-y. Lee, “SUPERB: Speech Processing Universal PERformance Benchmark,” in *Proc. Interspeech*, 2021, pp. 1194–1198. [Online]. Available: https://arxiv.org/pdf/2105.01051