13th edition of the Speech Synthesis Workshop
24-27 August 2025, Leeuwarden, The Netherlands

Interpolating Speaker Identities for Synthetic Voice Generation

Juliana Francis[1], *Joakim Gustafsson*[1], *Éva Székely*[1]

[1] Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Sweden

jfrancis@kth.se, jkgu@kth.se, szekely@kth.se

Abstract

We introduce a method for blending speaker embeddings to generate synthetic speaker identities for text-to-speech (TTS) systems. Traditional TTS models rely on real recorded voices, which limits their ability to synthesize speech tailored to specific applications without sourcing a suitable speaker. Our approach enables interpolation between existing speaker embeddings in embedding-based TTS models, allowing the creation of new voices without requiring additional data collection. We evaluate how well the generated speech retains characteristics of the source voices and assess the consistency of blended outputs in relation to the training data. Furthermore, we explore embedding space manipulation by performing controlled blending along reduced embedding dimensions, enabling more targeted voice synthesis. As a secondary contribution, we provide empirical evidence that this zero-shot TTS model generalizes less effectively to specific inputs when training data does not sufficiently approximate those points in the embedding space.

Index Terms: Speaker Generation, Text-to-Speech, Speaker Embeddings

1. Introduction

Zero-shot Text-to-Speech (TTS) systems have provided the ability to clone input audio from few second long speech samples and generate natural-sounding human speech, differing from typical TTS systems which can clone only a single voice or multiple voices from within its training data. However, these models are limited by the need for high quality real input speech samples for the model to replicate and synthesize new speech from the input. Further, if the speaker used to create these samples vocal qualities are not adequately represented within the training dataset, the recreation of the voice is often of poorer quality [1, 2].

This can cause issues, especially when it comes to attempts to utilize these systems in production environments. For example, currently users of speech generating devices are limited to available voices within their devices. Zero-shot TTS could provide an opportunity to diversify the voices available to these individuals without requiring large amounts of training data [3]. Applications such as voice assistants and characters within media could also benefit by allowing these to have procedurally generated content and text with voice attached to it. However, with zero-shot TTS synthesis currently limited to voices that have been produced by other people, it may also be difficult to find suitable voices from individuals willing to donate their voice for these purposes.

In this paper we present a method to create both blended and original voices through interpolation of speaker embeddings for synthesis using zero-shot TTS. The paper provides the following contributions:

- Methodology and creation of a method for generation of voice blends between different speakers by utilizing a fine-tuned zero-shot TTS model.
- Analysis of these methods through similarity to input blended speech embeddings and consistency of generated outputs.
- A method for blending of speech over individual components of speech through dimensionality reduced embeddings.

2. Related Work

2.1. Zero-Shot Text-to-Speech

VALL-E introduced the concept of using neural codecs as an intermediate representation using a GPT model, treating the process of performing TTS similarly to how GPT models predict text, modeling acoustic tokens with autoregressive transformers [4]. Voicecraft also makes use of a neural codec based approach to speech synthesis, using both past and future context using causal masking and delayed stacking of tokens, enabling it to learn to synthesize audio within a sequence by placing tokens in the central "missing" sections [5].

Other models have focused on disentangling speech representations to enhance quality and control of outputs. Naturalspeech 3 uses a neural codec approach to convert an input waveform into content, prosody, timbre, and other relevant acoustic information, which are then used within separate diffusion models to generate these different attributes for its corresponding output. Through these disentangled subspaces, the model is able to modify these qualities separately, enabling a form of voice creation with similarities to those used within this paper [6]. Mega-TTS 2 also disentangles different components of speech, in this case prosody and timbre by utilizing an autoencoder, as well as the use of language models to model prosody to extract useful information from input audio, which is employed to enable controllable prosody [7].

XTTS also uses a GPT based encoder that takes as input text tokens as for a conditioning encoder that provides high-dimensional embeddings of the input mel spectrograms. Further, the model utilizes the latent space of the encoder, conditioned on speaker embeddings to decode the mel spectrograms into synthesized audio. [2]

Flow matching has also emerged as a powerful framework in zero-shot TTS. This technique learns to transform simple base distributions into complex output distributions. Voicebox uses a non-autoregressive flow matching model that has been adapted for a variety of tasks, including conditioning on future content. However, to account for the difference in duration between text and speech, it requires a duration prediction mechanism [8]. E2 TTS also uses flow matching for use in generating

10.21437/SSW.2025-26

output mel-spectrograms, and aims to improve upon Voicebox by simplifying the model used. By replacing the duration prediction mechanism used within Voicebox with a filler token instead, reducing the complexity of the model and simplifying its training process [9].

Cosyvoice2 approaches the task with a causal flow matching model that conditions the synthesized outputs using both text and speech embeddings. The causal nature of the model enables it to operate in both streaming and non-streaming modes, enabling processing of inputs before receiving the full input sequence [10].

2.2. Speaker Generation

Speaker generation is the process of generating novel speakers to synthesize voice. Tacospawn approached speaker generation by utilizing a modified Tacotron model that enabled the use of speaker embeddings. The distribution of speaker embeddings is modelled using a mixture of gaussian priors trained using maximum likelihood estimation. The model is conditioned on accent and gender information, which can be provided when sampling from the distribution to guide the synthesized voice [11]. A modified Tacotron model was used to generate gender-ambiguous voices by analyzing principal components of speaker embeddings strongly associated with gender. Gender-ambiguous embeddings were then generated by sampling the PCA space between gendered speaker distributions, and projecting these samples into the full embedding space before synthesizing speech [12]. Similarly, a modified two-speaker Tacotron model interpolated between male and female speaker embeddings, with the use of utterance-level prosody control to reduce gendered cues and maintain ambiguity across expressive styles [13].

Another method was used based on normalizing flows, mapping input mel spectrograms to a latent representation through the invertible transforms within a flow-based model. The latent representations are used to generate outputs using a gaussian mixture model [14].

NANSY++ uses disentangled pitch, timbre, and linguistic representations of speech to synthesize audio conditioned on continuous gender and age values. The model used is able to generate new voices by modification of the input gender and age values [15].

Another method for generating new speakers is by using text prompts describing the desired voice that the synthesized text should be generated in. PromptTTS++ uses a manually annotated dataset with voice descriptions to train a diffusion-based model to model speaker qualities within the training data. The model is able to produce outputs based on a combination of acoustic features and the input voice descriptions [16]. PromptSpeaker uses a Glow model rather than a diffusion model to perform a similar task of converting a text representation to a a speaker representation, which is then used as input to a zero-shot TTS system to generate new speakers for this model [17].

3. Methods

3.1. Blending Method

To blend between multiple speakers, we linearly interpolate between speaker embeddings and conditioning latents generated using XTTS [2], by using a modified version of an open-source implementation of the model [18]. XTTS synthesizes speech based on both the speaker embedding and a GPT-based conditioning latent, which it predicts using input audio files, and as such we blend over both of these components to synthesize output audio.

Prior to any experiments described below, we fine-tuned the XTTS model for 100,000 steps on the VCTK dataset, which comprises 44 hours of read speech by 110 English speakers with various accents over 44282 audio samples [19]. This was performed such that we knew approximately what speakers were used within the training data for this research.

Given N speakers to blend, we denote their speaker embeddings as $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N$ and their corresponding GPT conditioning latents as $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_N$. The user specifies a set of blend weights $\mathbf{b} = [b_1, b_2, \ldots, b_N]$, which are normalized to ensure they sum to one:

$$\mathbf{b} \leftarrow \frac{\mathbf{b}}{\sum_{i=1}^{N} b_i} \tag{1}$$

Using these normalized weights, we compute the final blended representations as weighted sums:

$$(\mathbf{e}, \mathbf{l}) = \left(\sum_{i=1}^{N} b_i \cdot \mathbf{e}_i, \sum_{i=1}^{N} b_i \cdot \mathbf{l}_i \right) \tag{2}$$

Here, $\mathbf{e}$ is the final blended speaker embedding used for synthesis, and $\mathbf{l}$ is the corresponding blended GPT conditioning latent. These are then passed into the XTTS model to synthesize the final audio output.

For generation, let $S = \{s_1, s_2, \ldots, s_{20}\}$ be the set of the first 20 speakers from the VCTK dataset. For each speaker $s_i \in S$, we use two matched utterances to generate 3 sentences, resulting in 6 generated sentences per blend for each unique speaker pair. Let P be the set of all unique speaker pairs, i.e., $P = \{(s_i, s_j) \mid i \neq j, s_i, s_j \in S\}$. We then create blends for each pair $(s_i, s_j) \in P$, generating outputs using 7 blend weights, evenly spaced along a linear interpolation between the two speakers. These blend ratios were computed as convex combinations of the form:

$$(\alpha, 1-\alpha), \quad \alpha \in \{1.0, 0.83, 0.66, 0.5, 0.33, 0.17, 0.0\},$$

to assess how blending transitions between speaker identities.

Although our method supports blending across more than two speakers, this experiment focuses solely on pairwise blending in order to analyze the characteristics of each speaker transition.

For example of blended audios see: https://jandfranc.github.io/VoiceBlendingDemo/

3.2. Embedding Generation

To assess the performance of the system, we gather speaker embeddings from two sources: ECAPA-TDNN speaker embeddings [20] and the speaker embeddings generated by the fine-tuned XTTS model. The ECAPA-TDNN embeddings serve as a more robust reference due to their established reliability in speaker representation, while the internal speaker embeddings are directly used in the model to generate the blended outputs, so we believe are relevant to investigate also. These embeddings are collected from the entire VCTK corpus used for fine-tuning, as well as from all generated outputs. We interpolate between these embeddings as well as the conditioning latents to obtain intermediary speaker embeddings, using the above described blending method.

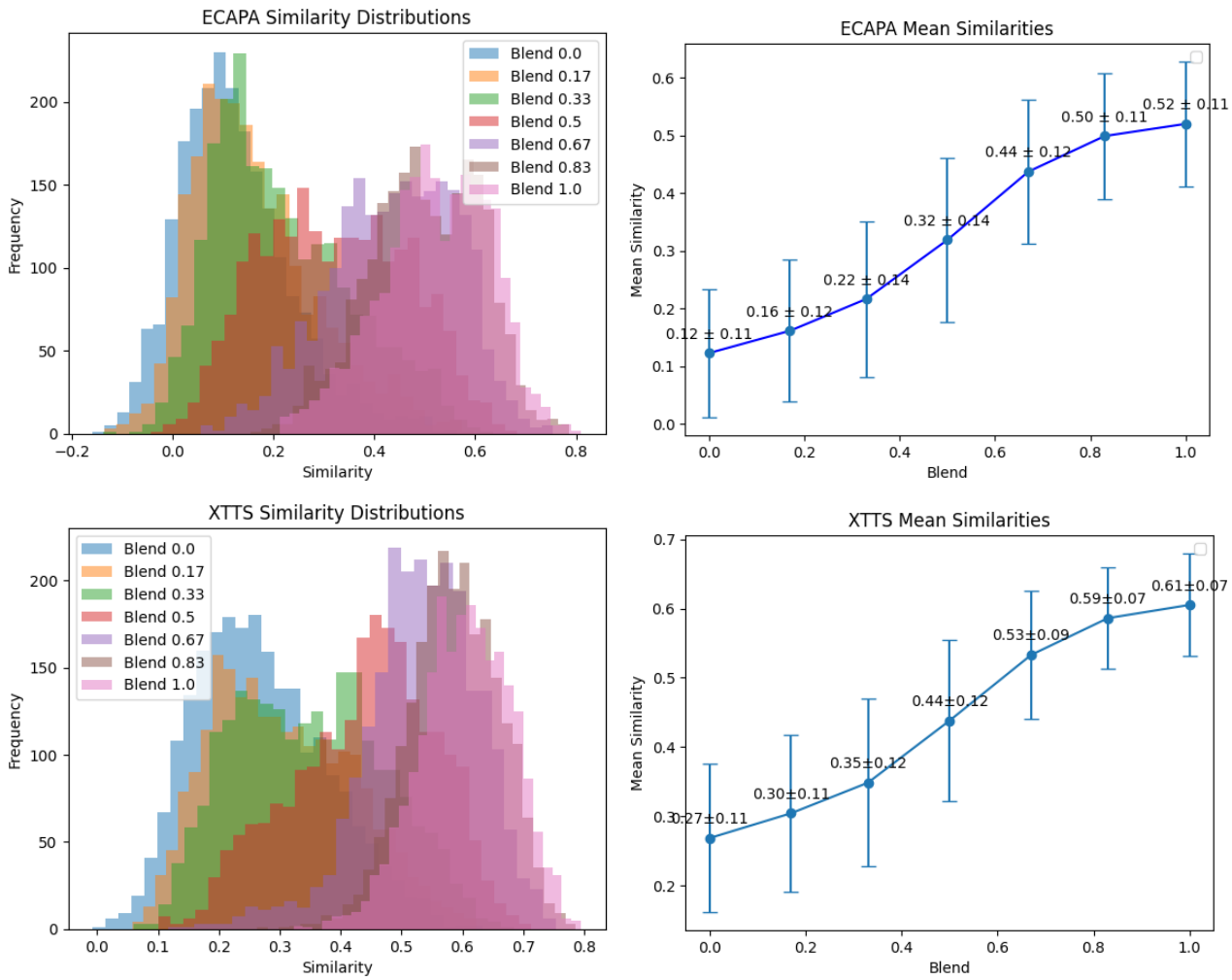

Figure 1: *Distribution of Cosine Similarity between Input and Output Embeddings. Top Row: ECAPA Embeddings. Bottom Row: XTTS Internal Encoder*

We further evaluate the system by calculating the cosine similarity between each blended embedding and all embeddings in the VCTK dataset [19]. Specifically, we measure how similar the generated blended embeddings are to the embeddings of "nearby" speakers in the training data. We class a "nearby" speaker as those with a cosine similarity of 0.6 or higher. The threshold for identifying "nearby" embeddings was chosen based on past work [21] and the need for a reasonable cutoff to distinguish between embeddings that are close enough to be considered similar. This allows us to assess how far each blended embedding diverges from speakers within the training data.

To evaluate the performance of the generated audio samples, we compute the cosine similarity between the output speaker embeddings and the corresponding input embeddings. We conduct four types of comparisons, each statistically evaluated using pairwise t-tests:

- **Proximity of embeddings to input (blended) embeddings:** We measure how many embeddings in the training data are close to the input embeddings, where the input embeddings are obtained by averaging between the two original speaker embeddings.
- **Similarity between blend-in and blend-out embeddings:** We assess the similarity between the blend-in and blend-out embeddings and their relation to the number of nearby embeddings in the training data.
- **Consistency of generated outputs:** We evaluate the consistency of the generated outputs by comparing generated samples for each speaker blend, and whether this consistency is affected by count of nearby speaker embeddings
- **Closest embedding count vs. average generated output similarity:** We compare the count of nearby embeddings to the input to the average similarity of each of the generated outputs to the training embeddings.

3.3. Dimensionality-Reduced Blending for Speaker Conditioning

To facilitate more flexible and interpretable blending methods in speaker conditioning, we introduce a technique for generating speaker embeddings and conditioning latents from reduced-dimensional inputs. These reduced inputs may be specified either without reference audio or through interpolation between two reference samples. In our target architecture (XTTS),

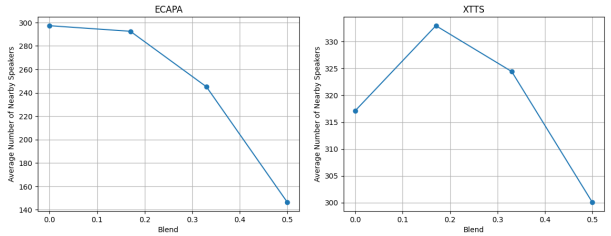

Figure 2: *Average Number of Nearby Speakers (Cosine Similarity greater than 0.6 within the fine-tuning data) by Blend, Blend Values of 1.0, 0.83, and 0.66 are reported along with 0.0, 0.17, and 0.34 respectively. Left: ECAPA Embeddings. Right: XTTS Internal Encoder*

speaker embeddings are 512-dimensional vectors, and conditioning latents are 32×1024 matrices.

3.3.1. Latent Compression via IPCA

We begin by flattening the conditioning latents and applying Incremental Principal Component Analysis (IPCA) to samples drawn from the VCTK corpus. This procedure yields 1000 principal components, which account for 99% of the variance. However, exposing this many dimensions to end users would be impractical for interactive manipulation.

To address this, we hypothesize that the conditioning latents can be predicted directly from the 512-dimensional speaker embeddings, since both representations are derived from the same reference audio. We frame this as a regression task and explore several model architectures, including fully connected feedforward networks and transformer-based models.

No matter the architecture used, cosine similarity between the predicted and ground-truth latents on the validation set plateaued at 0.64. We selected a transformer-based regression model that achieved this performance level while maintaining architectural simplicity.

3.3.2. Transformer Regression Architecture

Our final model comprises a transformer encoder with 8 attention heads, followed by a feedforward network composed of fully connected layers. Each layer uses batch normalization, GELU activation, and dropout for regularization. A final layer normalization step is applied to the output. This architecture achieves a cosine similarity of 0.64 on both the validation and test sets when predicting the 1000-dimensional latent space from the 512-dimensional embeddings.

3.3.3. Speaker Embedding Reduction for User Control

We also apply PCA to the 512-dimensional speaker embeddings. This analysis reveals that 312 components are required to explain 99% of the variance. However, for interface usability, we limit user interaction to the top 26 principal components, which account for approximately 50% of the total variance. The remaining components are either set to zero (for new samples) or preserved during interpolation (for blended inputs).

3.3.4. Audio Synthesis and Reconstruction Pipeline

To synthesize audio from edited latent representations, we proceed as follows:

1. Apply inverse PCA to reconstruct the full 512-dimensional speaker embedding from the 26 editable components and the remaining unedited components.
2. Use the transformer regression model to map the reconstructed embedding to a 1000-dimensional latent representation.
3. Apply inverse PCA to recover the original 32×1024 conditioning latent structure.

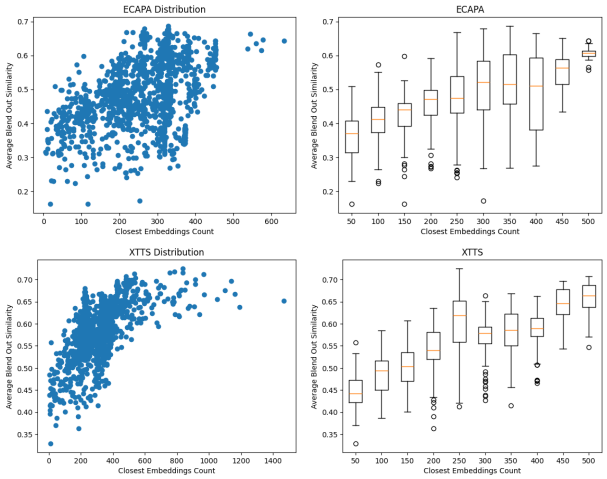

Figure 3: *Comparison of number of "close" embeddings within the fine-tuning data (Cosine Similarity greater than 0.6) to input embeddings, to the average cosine similarity to output blended values. Top Row: ECAPA Embeddings. Bottom Row: XTTS Internal Encoder*

3.3.5. Evaluation

To evaluate the proposed approach, we compute the cosine similarity between speaker embeddings extracted using ECAPA-TDNN from the reference audio and those reconstructed through our dimensionality-reduction and regression pipeline. This metric is used to provide an estimate of how much speaker identity information is preserved after dimensionality reduction and reconstruction using the above described method.

4. Results

4.1. Similarity by Blend

Cosine similarity between each input embedding and the blended embeddings were calculated to determine efficacy of the blending method used within the paper. For both types of embedding, cosine similarity to the input embeddings increased as the blend percentage increased, and all differences between blends were found to be statistically significant ($p < 0.001$). The distribution and change in similarity by blend can be seen in Figure 1.

4.2. Nearby Speakers By Blend

To investigate the distribution of speakers within the fine-tuned XTTS model the number of nearby speakers to each of the input embeddings used for speaker blending was calculated by computing the cosine similarity between these, cutting off nearby speakers at a minimum cosine similarity of 0.6, the average number of nearby speakers by blend can be seen in Figure 2. For the ECAPA embeddings, the average number of nearby speak-

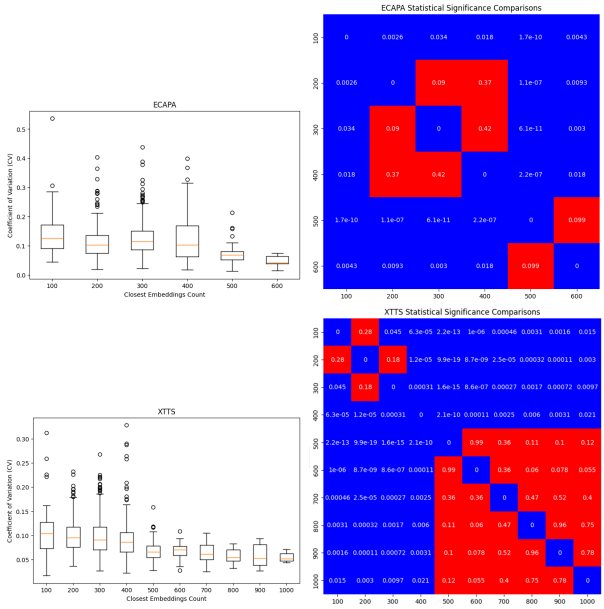

Figure 4: *Number of close embeddings by coefficient of variation. Top Row: ECAPA Embeddings. Bottom Row: XTTS Internal Encoder*

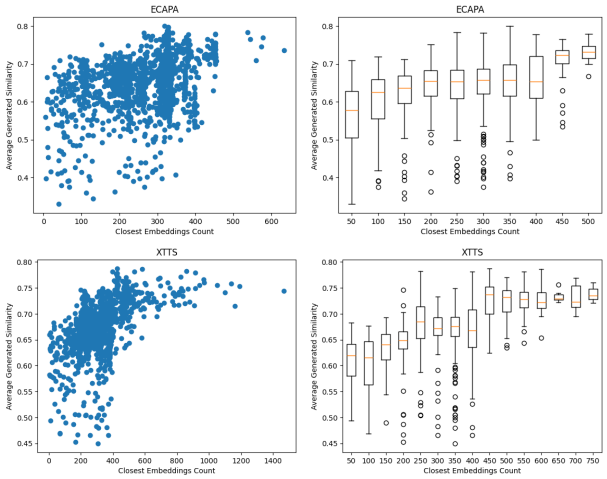

Figure 5: *Average Cosine Similarity for each Speaker/Blend Combination in comparison to number of close embeddings. Top Row: ECAPA Embeddings. Bottom Row: XTTS Internal Encoder*

ers decreased as the blend percentage increased, and significant differences were found between each blend ($p < 0.001$), except for the 0.0 and 0.17 embedding where there was no significant difference found ($p = 0.56$). For the internal XTTS embeddings, the range between the average number of speakers was more compressed than for ECAPA embeddings, potentially due to the fine-tuning process. Further, the only blends which were found to have significantly different amounts of nearby speakers was between the 0.17 and 0.5 blend values ($p < 0.05$). These results provide evidence that the latent speaker space in the fine-tuned XTTS model reflects the distribution of the fine-tuning data, even as embedding blend proportions are varied.

4.3. Nearby Embedding Count compared to Output Similarity

The cosine similarity between the blended inputs similarity to the synthesized blended outputs was computed. As the number of nearby embeddings increased, so did average cosine similarity, however the overall effect of this appears to decrease as the number of nearby embeddings increased. Significant differences were found between each "group" of embeddings except for the XTTS encoder between the 50-100 and 100-150 counts ($p = 0.18$) and the 300-350 to the 350-400 counts ($p = 0.097$), and for the ECAPA embeddings between the 50-100 and 100-150 counts ($p = 0.14$), the 200-250 and 350-400 counts ($p = 0.13$), the 250-300 and 300-350 counts ($p = 0.81$) and the 250-300 and 350-400 counts ($p = 0.05$). The distribution of this data can be seen in Figure 3.

4.4. Variation of Embedding Similarity compared to Number of Nearby Embeddings

To determine the effects of nearby embeddings on the variance of the blended outputs similarity, the coefficient of variation was calculated on the blended outputs similarity to its inputs. This was assessed to see whether performance was more "stable" when the inputs had more nearby "reference embeddings" within the training data to draw from. For the XTTS internal embeddings, the coefficient of variation decreases as the number of nearby embeddings increases until approximately 400-500 nearby count where there are no longer significant differences found between the coefficient of variations at this point. This suggests that the density of nearby embeddings does have an effect on the output qualities consistency up to a certain point where this effect appears to decrease/disappear. Similar effects are not seen within the ECAPA embeddings, with no particular clear pattern, except for a significant decrease in the coefficient of variation at the 100-200 embeddings count and the 300-400 embeddings count. To assess whether differences in output similarity across nearby embedding bins were statistically significant, we computed pairwise p-values using T-tests between bins. These are visualized in Figure 4 as a heatmap, where lower values marked in blue indicate statistical significance ($p < 0.05$).

4.5. Consistency of Generated Outputs

To further assess how the consistency of outputs changes in relation to the number of nearby embeddings, the cosine similarity between all generated outputs (for a specific blend value per speaker pair) and calculated the mean to get an average value for the similarity between generated outputs. There is a gradual increase across both similarities as the number of nearby embeddings increased, with this increase appearing to plateau for the XTTS internal embeddings at approximately the 400-450 nearby embeddings group where there are no longer significant differences as the number of nearby embeddings increase. In relation to this model and the dataset it was fine-tuned on, there appears to be an upper limit to the amount of necessary training data to achieve good consistency for a specific speaker, blended or otherwise. The distribution of the outputs can be seen in Figure 5 and heatmaps of statistical significance, where significant values ($p < 0.05$) are represented in blue, can be seen in Figure 6.

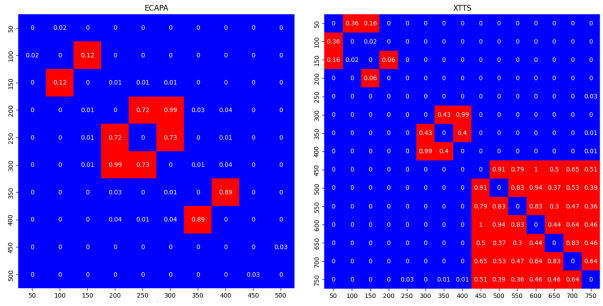

Figure 6: *Statistical significance heatmaps between bins of nearby speakers by cosine similarity of output embeddings. Top Row: ECAPA, Bottom Row: XTTS Internal Encoder*

4.6. Blended Embedding Proximity

To evaluate how well the voice generation method preserves the original voice characteristics, we measured the proximity between the input embeddings and the generated outputs. Since each processing step may introduce information loss, this analysis focused on whether the reconstructed voices remained close to the original embeddings.

We used a held-out test set from VCTK, comprising 15% of the total utterances, for this evaluation. The remaining data had been used to train the embedding-to-reduced-latent regression model.

As shown in Figure 7, there is a clear difference in similarity scores when comparing reconstructions based on full original embeddings versus those based on reduced embeddings. An independent t-test confirmed that this difference is statistically significant ($t = 50.8$, $p = 0$), suggesting that the embedding reduction introduces a deviation from the original representation.

5. Discussion

In this paper we have provided evidence that blending between speaker embeddings can be used to produce intermediate voices that gradually transition from one speaker to another as blend percentage increases. Further, we discovered conditions under which this performance performs worse, in terms of that the variation decreases and consistency of outputs increases as the number of nearby speakers to the input embedding increases, up to a plateau. It is a possibility this plateau could be potentially caused by an absence of data points within these regions, as there were fewer input audio samples that had larger values of nearby speaker embeddings.

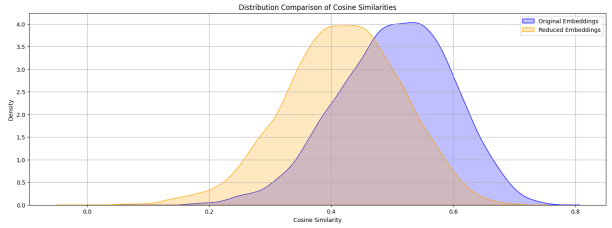

Figure 7: *A graph showing the difference distribution of similarities between the original embeddings and the reduced embeddings after running through the Voice Creation method.*

The fact that the blended audio retains similarity to both the averaged input embeddings and the original speaker identities indicates that this method effectively generates intermediate voices. While the voice creation method does result in some information loss-evidenced by a shift in the distribution toward lower overall similarity scores, this does not necessarily indicate a failure of the method. Although the reduced similarity suggests the method is not optimal for accurately recreating the original embeddings, this was never the primary objective. Instead, our aim was to develop a method that allows for controlled modification of certain voice characteristics via reduced PCA embeddings. In this context, perfect reconstruction is not essential. The fact that the generated voices still retain some degree of similarity to the original inputs demonstrates that the blending process preserves important aspects of the voice, even after dimensionality reduction. Future research will be needed to assess how effectively this method can alter specific voice features in practice, particularly with feedback and guidance from relevant user groups such as individuals using speech-generating devices or creators of virtual agents. Nevertheless, we consider the current results, showing retained similarity despite embedding reduction, a meaningful and encouraging finding.

This research has provided evidence that blending speaker embeddings is a viable approach for generating intermediate voices between speakers, allowing for controlled transitions between speaker identities, which could allow for better creation of generated voices for a variety of applications. Future work should focus on subjective evaluations of voice quality and identity perception across the blend ratios, as well as further research into the ability to blend between different voice qualities from the reduced embeddings provided by the voice creation method.

6. Acknowledgements

This work is funded by the WASP funded project Real-time context-aware speech prosthesis for conversational interaction.

7. References

[1] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," *arXiv e-prints*, pp. arXiv–2106, 2021.

[2] E. Casanova, K. Davis, E. Gölge, G. Göknar, I. Gulea, L. Hart, A. Aljafari, J. Meyer, R. Morais, S. Olayemi, and J. Weber, "Xtts: a massively multilingual zero-shot text-to-speech model," arXiv.org, 06 2024. [Online]. Available: https://arxiv.org/abs/2406.04904

[3] J. Francis, É. Székely, and J. Gustafson, "Connectone: A modular aac system prototype with contextual generative text prediction and style-adaptive conversational tts," pp. 1001–1002, 09 2024.

[4] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," *arXiv preprint arXiv:2301.02111*, 2023.

[5] P. Peng, P.-Y. Huang, A. Mohamed, and D. Harwath, "Voicecraft: Zero-shot speech editing and text-to-speech in the wild," *arXiv*, 2024.

[6] Z. Ju, Y. Wang, K. Shen, X. Tan, D. Xin, D. Yang, Y. Liu, Y. Leng, K. Song, S. Tang *et al.*, "Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models," *arXiv preprint arXiv:2403.03100*, 2024.

[7] Z. Jiang, J. Liu, Y. Ren, J. He, Z. Ye, S. Ji, Q. Yang, C. Zhang, P. Wei, C. Wang *et al.*, "Mega-tts 2: Boosting prompting mechanisms for zero-shot speech synthesis," *arXiv preprint arXiv:2307.07218*, 2023.

[8] M. Le, A. Vyas, B. Shi, B. Karrer, L. Sari, R. Moritz, M. Williamson, V. Manohar, Y. Adi, J. Mahadeokar *et al.*, "Voicebox: Text-guided multilingual universal speech generation at scale," *Advances in neural information processing systems*, vol. 36, pp. 14 005–14 034, 2023.

[9] S. E. Eskimez, X. Wang, M. Thakker, C. Li, C.-H. Tsai, Z. Xiao, H. Yang, Z. Zhu, M. Tang, X. Tan *et al.*, "E2 tts: Embarrassingly easy fully non-autoregressive zero-shot tts," in *2024 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2024, pp. 682–689.

[10] Z. Du, Y. Wang, Q. Chen, X. Shi, X. Lv, T. Zhao, Z. Gao, Y. Yang, C. Gao, H. Wang *et al.*, "Cosyvoice 2: Scalable streaming speech synthesis with large language models," *arXiv preprint arXiv:2412.10117*, 2024.

[11] D. Stanton, M. Shannon, S. Mariooryad, R. Skerry-Ryan, E. Battenberg, T. Bagby, and D. Kao, "Speaker generation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7897–7901.

[12] K. Markopoulos, G. Maniati, G. Vamvoukakis, N. Ellinas, G. Vardaxoglou, P. Kakoulidis, J. Oh, G. Jho, I. Hwang, A. Chalamandaris, P. Tsiakoulis, and S. Raptis, "Generating multilingual gender-ambiguous text-to-speech voices," in *Interspeech 2023*, 2023, pp. 621–625.

[13] Éva Székely, J. Gustafson, and I. Torre, "Prosody-controllable gender-ambiguous speech synthesis: A tool for investigating implicit bias in speech perception," in *Interspeech 2023*, 2023, pp. 1234–1238.

[14] P. Bilinski, T. Merritt, A. Ezzerg, K. Pokora, S. Cygert, K. Yanagisawa, R. Barra-Chicote, and D. Korzekwa, "Creating new voices using normalizing flows," in *Interspeech 2022*, 2022, pp. 2958–2962.

[15] H.-S. Choi, J. Yang, J. Lee, and H. Kim, "Nansy++: Unified voice synthesis with neural analysis and synthesis," *arXiv preprint arXiv:2211.09407*, 2022.

[16] R. Shimizu, R. Yamamoto, M. Kawamura, Y. Shirahata, H. Doi, T. Komatsu, and K. Tachibana, "Prompttts++: Controlling speaker identity in prompt-based text-to-speech using natural language descriptions," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 12 672–12 676.

[17] Y. Zhang, G. Liu, Y. Lei, Y. Chen, H. Yin, L. Xie, and Z. Li, "Promptspeaker: Speaker generation based on text descriptions," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2023, pp. 1–7.

[18] G. Eren and The Coqui TTS Team, "Coqui TTS," Jan. 2021. [Online]. Available: https://github.com/idiap/coqui-ai-TTS

[19] C. Veaux, J. Yamagishi, and K. MacDonald, "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit," https://datashare.ed.ac.uk/handle/10283/2651, The Centre for Speech Technology Research (CSTR), University of Edinburgh, 2017.

[20] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," in *Proc. Interspeech*, 2020, pp. 3830–3834.

[21] S. Wang and É. Székely, "Evaluating text-to-speech synthesis from a large discrete token-based speech language model," in *Proc.LREC-COLING*, 2024, pp. 6464–6474.