Text to Speech

Vișan Ionuț

Abstract

Text-to-Speech (TTS) models are key to generating natural and expressive speech in applications like voice assistants and screen readers. This study explores the use of SpeechT5 for personalized voice synthesis using a dual-dataset strategy: one general corpus and one speaker-specific set [1]. Since standard metrics are limited in capturing voice quality and speaker similarity, our evaluation relies mainly on subjective listening tests [2]. Results show improved naturalness and a closer match to the target voice, highlighting the importance of perceptual evaluation in TTS model adaptation.

1. Introduction

In recent years, Text-to-Speech (TTS) technology has undergone significant advancements, driven by the rapid development of deep learning and sequence-to-sequence architectures. Modern TTS systems are now capable of generating speech that is not only intelligible but also expressive and natural-sounding. These models have become foundational components in a wide array of applications, including voice assistants, audiobooks, screen readers, virtual avatars, and other systems that rely on synthetic speech to communicate with users.

Among recent architectures, SpeechT5 [3] stands out as a unified framework capable of handling both speech and text generation tasks. Based on the transformer architecture, SpeechT5 is pretrained on diverse modalities, enabling it to generalize across tasks such as automatic speech recognition, speech synthesis, and even voice conversion. This versatility makes it an appealing choice for building high-quality TTS systems, especially when the goal is to adapt the synthetic voice to specific speakers or user preferences.

However, achieving speaker-adaptive TTS remains a challenging task, particularly in scenarios with limited target speaker data or acoustic variability. In such cases, balancing generalization and personalization becomes critical. To address this, we will explore a two-stage fine-tuning strategy.

2. Task Description

The objective of this work is to develop a speaker-adaptive Text-to-Speech (TTS) system capable of generating high-quality, natural-sounding speech that closely matches the vocal characteristics of a specific speaker. To achieve this, we build upon the SpeechT5 architecture and investigate a dual-stage fine-tuning approach [1] aimed at improving both the generalization and personalization capabilities of the model.

The task is structured as follows:

- General adaptation: The base SpeechT5 model is fine-tuned on a large, diverse speech dataset to reinforce its ability to handle a wide range of linguistic and prosodic patterns. This step ensures that the model maintains intelligibility and fluency across various inputs.
- Speaker-specific adaptation: A smaller dataset containing recordings of a single target speaker is used to further fine-tune the model. The goal of this stage is to align the synthetic voice with the speaker's unique vocal traits, such as timbre, pitch, and speaking style.

Due to the limited effectiveness of existing quantitative metrics in capturing the nuances of speaker identity and expressiveness, we rely primarily on subjective auditory evaluations to assess model performance. This task setup reflects real-world needs where human perception remains the gold standard for evaluating TTS quality, especially in applications involving personalized or emotionally expressive voices.

3. Algorithm Description

We will provide an overview of the implementation details and components used in our speaker-adaptive Text-to-Speech system. The model is built upon the SpeechT5 architecture and leverages pre-trained modules for both text-to-speech synthesis and speaker embedding extraction. The training pipeline integrates custom data loading, audio preprocessing, and optimization strategies tailored to fine-tune the model on h-quality, speaker-conditioned speech data.

3.1. Base Model: SpeechT5

SpeechT5 is a unified encoder-decoder architecture based on the Transformer model, designed to handle a variety of speech and text-related tasks within a single framework. Developed by Microsoft, it generalizes the T5 [4] (Text-to-Text Transfer Transformer) paradigm to support multimodal inputs and outputs, including both textual tokens and acoustic representations. In the context of Text-to-Speech (TTS), SpeechT5 functions by converting text inputs into mel-spectrogram sequences, which are subsequently transformed into waveform audio using an external vocoder.

The model consists of two primary components:

- a **text encoder** that processes the input text sequence into contextual embeddings
- a **spectrogram decoder** that autoregressively generates mel-spectrogram frames, conditioned on the encoder outputs and, optionally, a speaker embedding vector.

Transformer Encoder

The encoder processes tokenized text inputs into contextual embeddings using a multi-layer Transformer architecture with self-attention, normalization, and feedforward layers. These embeddings capture the semantic and syntactic content of the input text.

Transformer Decoder

The decoder is responsible for generating mel-spectrogram frames in an autoregressive manner. It attends to both previously generated frames (via masked self-attention) and the encoder output (via cross-attention). The decoder operates over a continuous output space, predicting mel-spectrogram frames of fixed dimensionality, commonly 80 bins per frame.

In practice, the decoder can also utilize a reduction factor, predicting multiple spectrogram frames at each time step to speed up training and inference. This is crucial for long sequences, as it improves efficiency without significantly degrading audio quality.

Speaker Embedding Integration

Speaker-specific synthesis is achieved by conditioning the decoder on a 512-dimensional speaker embedding, extracted using a pretrained x-vector model (e.g., speechbrain/spkrec-xvect-voxceleb [5]). This vector modulates the decoding process to match the desired voice identity.

Output Representation

The output of SpeechT5 is a sequence of mel-spectrogram frames, each representing the spectral energy distribution across frequency bands at a given time. These frames are used as an intermediate representation and are converted to waveform audio using a vocoder such as HiFi-GAN [6].

Pretraining and Initialization

SpeechT5 benefits from large-scale pretraining on multimodal datasets, which enhances its generalization capabilities. The pretrained checkpoint microsoft/speecht5_tts serves as the starting point for fine-tuning. It has been trained on large and diverse corpora of text and speech, enabling it to perform well across multiple languages and speakers with minimal adaptation.

3.2. Architecture Overview

The SpeechT5 architecture follows a standard encoder-decoder Transformer design. The encoder processes tokenized input text and transforms it into contextual embeddings. In parallel, a speaker embedding is extracted from reference audio using a pretrained x-vector model (e.g., SpeechBrain's spkrec-xvect). These speaker features are integrated into the decoder, which then autoregressively generates mel-spectrogram frames. The decoder uses both self-attention over previously generated frames and cross-attention over the encoder output. The final spectrogram sequence is designed to be converted into waveform audio via a neural vocoder such as HiFi-GAN. This design allows the model to synthesize speech that is both intelligible and speaker-specific.

3.3. Dataset Preparation

To fine-tune the SpeechT5 model for personalized speech synthesis, we constructed a dataset loader that processes paired audio-transcript examples and generates corresponding training inputs. The dataset is defined through a CSV file containing paths to audio files and their textual transcripts. Each audio sample is loaded and resampled to 16 kHz to match the expected input format of both the SpeechT5 processor and the speaker embedding model.

To condition the speech generation on speaker identity, we use the speechbrain/spkrec-xvect-voxceleb model to extract x-vector embeddings from the raw waveform. These embeddings, normalized to unit norm, serve as fixed-length speaker representations injected into the decoder.

The SpeechT5Processor is then used to tokenize the text input and compute mel-spectrogram targets from the waveform. The resulting inputs are:

- input_ids – tokenized text,
- labels – mel-spectrograms as training targets,
- speaker_embeddings – 512-dimensional speaker vectors.

Since mel-spectrogram sequences can vary in length, all outputs are padded to a consistent shape during collation. Spectrograms are padded or trimmed to ensure dimensionality consistency (e.g., 80 bins per frame), and a reduction factor is applied when necessary to adjust sequence lengths.

This pipeline ensures that the model receives coherent and speaker-specific supervision during training, with all preprocessing handled dynamically through the PyTorch Dataset and DataLoader abstractions.

3.4. Training Procedure

The training pipeline uses a standard supervised learning approach, fine-tuning the model on text–audio pairs with speaker embeddings to guide vocal characteristics. During each epoch, the model predicts mel-spectrograms from input text, computes loss, and updates weights. A learning rate scheduler and early stopping help maintain stability and prevent overfitting. The best-performing model on the validation set is saved for inference, enabling gradual adaptation to both language and speaker traits.

3.5. Saving and Visualization

Throughout the training process, key metrics such as training loss and validation loss are recorded at each epoch. The model checkpoints are automatically saved whenever the validation performance improves, ensuring that the best version is retained. This allows for safe recovery and evaluation after training.

To monitor progress, loss curves are plotted and saved, offering a visual summary of the model's convergence behavior and generalization over time. These visualizations help identify underfitting, overfitting, or plateaus, and provide insight into training dynamics that can guide future adjustments.

4. Datasets Description

Our approach relies on a two-stage fine-tuning process using two datasets: a general multi-speaker corpus to reinforce overall synthesis quality, and a small speaker-specific set to adapt the voice to a particular individual. This setup ensures both naturalness and personalization in the generated speech.

4.1. Common Voices 20 [7]

common_voice_ro_25791627.wav	Aceste informaţii trebuie totuşi difuzate mai repede.
common_voice_ro_21393813.wav	Acești oameni sunt uciși unul după altul.
common_voice_ro_28931923.wav	A fost introdus un număr de îmbunătăţiri concrete.
common_voice_ro_23976671.wav	De fapt, chiar am rezerve în acest sens.
common_voice_ro_21279992.wav	Trebuie să ne ghidăm după fapte.

Fig. 1 – Examples from Common Voices 20 Dataset

Common Voices is an open-source dataset of speech recordings created by Mozilla to improve speech recognition technologies. It consists of crowdsourced voice samples in multiple languages, contributed by volunteers worldwide.

The raw dataset initially contained numerous recordings with inaccurate transcriptions and inconsistencies in format. To ensure optimal compatibility with modern TTS models, all audio files were carefully reviewed, manually corrected, and converted to a standardized format, 16 kHz sampling rate, mono channel, and .wav file type. This preprocessing step was essential to guarantee high-quality alignment between audio and text.

Dataset Summary

Total number of audio segments: 41,431.

Total duration of all audio segments combined: approximately 47 hours.

The file contains 2 columns:

- *audio_file*: File names of the processed audio segments.
- *transcript*: Corresponding text transcriptions for each audio.

To split the dataset, we performed an 80-20 split into training and test sets using a seed value of 42, resulting in:

- *train_common_voices20.csv*: It contains 33,144 of the audio segments.
- *test_common_voices20.csv*: It contains 8,287 of the audio segments.

To ensure consistent and model-friendly text input, we normalized all transcripts by replacing romanian diacritics and other accented characters with their plain latin equivalents. This preprocessing step is especially important because the SpeechT5 tokenizer does not natively support romanian-specific characters, which could otherwise lead to tokenization errors or misalignment during training.

4.2. Own Dataset

audio_file	valence	arousal	dominance	tone - first	tone - secc	transcript	speaker	category
1_vad_left_	0.64	0.61	0.58	Positive	Enthusiast	Am si eu ur	Trandafir	Operator
3_vad_left_	0.39	0.48	0.57	Neutral	Informative	E corect, c	Trandafir	Operator
4_vad_left_	0.56	0.59	0.58	Negative	Frustrated	Aaa, client	Trandafir	Operator
2_vad_righ	0.41	0.52	0.48	Neutral	Curious	Ati trecut s	Tatiana Sib	CSR
3_vad_righ	0.42	0.41	0.36	Neutral	Curious	la mergeti	Tatiana Sib	CSR
2_vad_left_	0.48	0.71	0.69	Neutral	Informative	Prima data	Rodica	Operator
3_vad_left_	0.56	0.64	0.59	Negative	Frustrated	Ulterior a r	Rodica	Operator
4_vad_left_	0.48	0.53	0.55	Negative	Sad	Eu am sun	Rodica	Operator

Fig. 2 – Examples from Own Dataset

The dataset was derived from segmented natural conversations between customers and customer support agents. Each audio segment was carefully processed to meet model specifications (16 kHz sampling rate, mono channel, and .wav format) and annotated for various use cases. For the purpose of our work, we utilize only the transcript and the associated speaker identity.

Dataset Summary

- Total number of audio segments: 2000
- Total duration: approximately 2.5 hours

To split the dataset, we performed an 80-20 split into training and test sets using a seed value of 42, resulting in:

- *train_own_dataset.csv*: It contains 1600 of the audio segments.
- *test_own_dataset.csv*: It contains 400 of the audio segments.

To ensure consistent and model-friendly text input, we normalized all transcripts by replacing romanian diacritics and other accented characters with their plain Latin equivalents. This preprocessing step is especially important because the SpeechT5 tokenizer does not natively support romanian-specific characters, which could otherwise lead to tokenization errors or misalignment during training.

5. Evaluation

In the absence of standard, quantitative evaluation metrics tailored specifically for Text-to-Speech models like SpeechT5, assessing model quality remains a nuanced challenge. While the training and validation loss provide a general indication of convergence and learning stability, they do not fully capture perceptual attributes such as naturalness, intelligibility, or speaker similarity.

Therefore, in this study, we rely primarily on loss values as a baseline reference, while placing significant emphasis on subjective listening tests. By generating audio samples from the fine-tuned model and evaluating them qualitatively, we gain a more meaningful understanding of how well the synthesized speech aligns with the intended voice characteristics. This combined evaluation approach ensures both technical monitoring and perceptual validation of model performance.

For waveform reconstruction, we employ HiFi-GAN, a generative adversarial network-based vocoder known for producing high-fidelity and natural-sounding audio. It reconstructs raw waveform signals from predicted mel-spectrograms, enabling fast and high-quality speech synthesis with low computational overhead.

The objective of this study is to adapt the model to the distinct voices of customer support agents found in Own Dataset. To determine the most effective strategy for achieving this, we evaluate the fine-tuned models across three ideas, aiming to identify the approach that best captures the target vocal identity.

5.1. Fine-Tune Common Voices 20

The SpeechT5 model will be fine-tuned with various configurations on the Common Voices 20 dataset to establish a robust understanding of romanian linguistic patterns and improve its overall adaptability to the language.

Configuration 1:

Model = “microsoft/speecht5_tts”

Batch_size = 4

Learning rate = 1e-5

Num Epochs = 10

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 1:

epoch	train_loss	val_loss	lr
1	0.571882	0.491433	9.00E-06
2	0.520797	0.475184	8.00E-06
3	0.506629	0.465962	7.00E-06
4	0.498153	0.461554	6.00E-06
5	0.492415	0.4594	5.00E-06
6	0.487312	0.45479	4.00E-06
7	0.483693	0.45342	3.00E-06
8	0.480446	0.451312	2.00E-06
9	0.478447	0.44903	1.00E-06
10	0.476937	0.44939	0

Fig. 3 – Metrics over all epochs for Configuration 1 (5.1.)

Fig. 4 – Training vs Validation Loss Plot for Configuration 1 (5.1.)

Model saved at epoch 9 with **Val Loss: 0.4490**.

Configuration 2:

Model = “microsoft/speecht5_tts”

Batch_size = 4

Learning rate = 5e-5

Num Epochs = 10

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 2:

epoch	train_loss	val_loss	lr
1	0.564699	0.495717	4.50E-05
2	0.521763	0.4803	4.00E-05
3	0.504016	0.471917	3.50E-05
4	0.493364	0.459865	3.00E-05
5	0.484204	0.451394	2.50E-05
6	0.475762	0.449382	2.00E-05
7	0.469162	0.444753	1.50E-05
8	0.462713	0.440329	1.00E-05
9	0.457522	0.436667	5.00E-06
10	0.453095	0.435574	0

Fig. 5 – Metrics over all epochs for Configuration 2 (5.1.)

Fig. 6 – Training vs Validation Loss Plot for Configuration 2 (5.1.)

Model saved at epoch 10 with **Val Loss: 0.4356**.

Configuration 3:

Model = “microsoft/speecht5_tts”

Batch_size = 4

Learning rate = 1e-4

Num Epochs = 10

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 3:

epoch	train_loss	val_loss	lr
1	0.612953	0.532008	0.0001
2	0.570926	0.510933	8.89E-05
3	0.539307	0.495305	7.78E-05
4	0.521182	0.483905	6.67E-05
5	0.507552	0.473164	5.56E-05
6	0.494464	0.464771	4.44E-05
7	0.483533	0.457328	3.33E-05
8	0.47362	0.447119	2.22E-05
9	0.464954	0.441961	1.11E-05
10	0.457518	0.438486	0

Fig. 7 – Metrics over all epochs for Configuration 3 (5.1.)

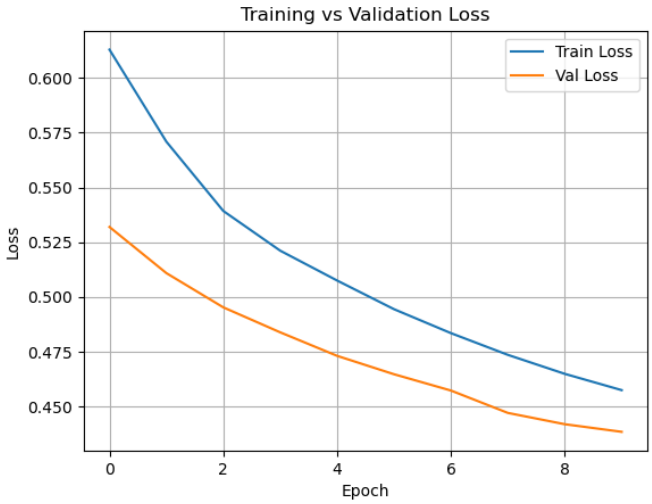

Fig. 8 – Training vs Validation Loss Plot for Configuration 3 (5.1.)

Model saved at epoch 10 with **Val Loss: 0.4385**.

Configuration 4:

Model = “microsoft/speecht5_tts”

Batch_size = 4

Learning rate = 5e-5

Num Epochs = 40

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 4:

epoch	train_loss	val_loss	lr
1	0.66165	0.50306	1.25E-05
2	0.53416	0.4921	2.50E-05
3	0.52031	0.48272	3.75E-05
4	0.5167	0.48336	5.00E-05
5	0.5109	0.46903	4.86E-05
6	0.50135	0.46631	4.72E-05
7	0.49462	0.46405	4.58E-05
8	0.48936	0.45733	4.44E-05
9	0.48483	0.45869	4.31E-05
10	0.4802	0.45401	4.17E-05
11	0.47648	0.45363	4.03E-05
12	0.47307	0.45454	3.89E-05
13	0.47069	0.45099	3.75E-05
14	0.46756	0.45282	3.61E-05
15	0.46518	0.44788	3.47E-05
16	0.46241	0.44627	3.33E-05
17	0.45937	0.4457	3.19E-05
18	0.45679	0.44452	3.06E-05
19	0.45388	0.4448	2.92E-05
20	0.45271	0.4421	2.78E-05
21	0.45011	0.43865	2.64E-05
22	0.44646	0.4379	2.50E-05
23	0.44481	0.43854	2.36E-05
24	0.44397	0.43633	2.22E-05
25	0.44035	0.43796	2.08E-05
26	0.43822	0.4486	1.94E-05
27	0.43661	0.43669	1.81E-05
28	0.4351	0.43512	1.67E-05
29	0.43339	0.43769	1.53E-05
30	0.43095	0.43839	1.39E-05
31	0.42942	0.43741	1.25E-05
32	0.42753	0.43709	1.11E-05
33	0.42593	0.43612	9.72E-06

Fig. 9 – Metrics over all epochs for Configuration 4 (5.1.)

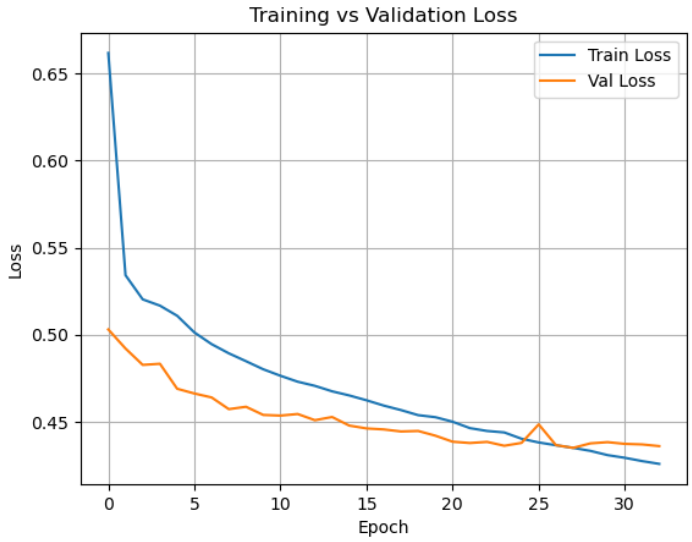

Fig. 10 – Training vs Validation Loss Plot for Configuration 4 (5.1.)

Model saved at epoch 28 with **Val Loss: 0.4351**.

5.2. Fine-Tune Own Dataset

The SpeechT5 model will be fine-tuned on the Own Dataset to assess its ability to adapt when trained on a limited amount of speaker-specific data.

Configuration 1:

Model = “microsoft/speecht5_tts”

Batch_size = 4

Learning rate = 5e-5

Num Epochs = 10

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 1:

epoch	train_loss	val_loss	lr
1	0.896584	0.75031	5.00E-05
2	0.754731	0.696055	4.44E-05
3	0.726461	0.65818	3.89E-05
4	0.691058	0.645016	3.33E-05
5	0.67512	0.635962	2.78E-05
6	0.662442	0.628272	2.22E-05
7	0.657467	0.622195	1.67E-05
8	0.649986	0.618487	1.11E-05
9	0.639537	0.615163	5.56E-06
10	0.633067	0.611905	0

Fig. 11 – Metrics over all epochs for Configuration 1 (5.2.)

Fig. 12 – Training vs Validation Loss Plot for Configuration 1 (5.2.)

Model saved at epoch 10 with **Val Loss: 0.6119**.

Configuration 2:

Model = “microsoft/speecht5_tts”

Batch_size = 4

Learning rate = 5e-5

Num Epochs = 40

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 2:

epoch	train_loss	val_loss	lr
1	0.977298	0.796906	1.25E-05
2	0.801122	0.719521	2.50E-05
3	0.757356	0.682575	3.75E-05
4	0.722095	0.673346	5.00E-05
5	0.700182	0.651037	4.86E-05
6	0.681544	0.644461	4.72E-05
7	0.676661	0.629738	4.58E-05
8	0.673461	0.639929	4.44E-05
9	0.657693	0.626761	4.31E-05
10	0.646926	0.622895	4.17E-05
11	0.648283	0.628496	4.03E-05
12	0.6365	0.620715	3.89E-05
13	0.630259	0.618121	3.75E-05
14	0.626193	0.623426	3.61E-05
15	0.626144	0.626392	3.47E-05
16	0.618667	0.612439	3.33E-05
17	0.619695	0.615454	3.19E-05
18	0.608701	0.608222	3.06E-05
19	0.608432	0.61367	2.92E-05
20	0.605658	0.613664	2.78E-05
21	0.600466	0.609071	2.64E-05

Fig. 13 – Metrics over all epochs for Configuration 2 (5.2.)

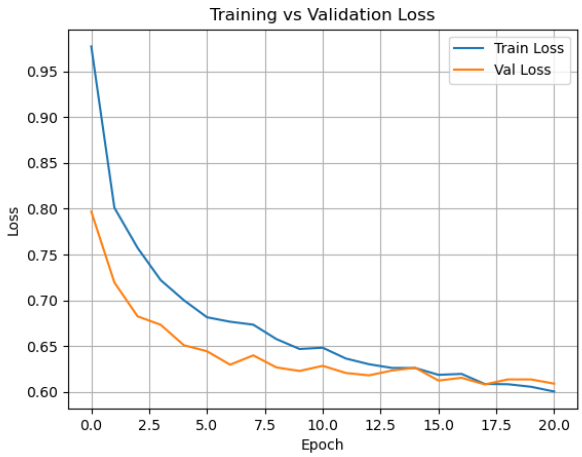

Fig. 14 – Training vs Validation Loss Plot for Configuration 2 (5.2.)

Model saved at epoch 10 with **Val Loss: 0.6082**.

5.3. Fine-Tune model obtained at 5.1. on Own Dataset

We proceed to apply the two-step fine-tuning strategy with the goal of further lowering the validation loss on the Own Dataset below the 0.6082. To do so, we build upon the model obtained in Section 5.1., configuration 4, leveraging its prior adaptation to the romanian language and refining it further using our speaker-specific dataset.

Configuration 1:

Model = "speecht5_finetuned"

Batch_size = 4

Learning rate = 5e-5

Num Epochs = 10

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 1:

epoch	train_loss	val_loss	lr
1	0.765864	0.671306	5.00E-05
2	0.695541	0.633098	4.44E-05
3	0.671459	0.62193	3.89E-05
4	0.652855	0.617118	3.33E-05
5	0.64119	0.607676	2.78E-05
6	0.630832	0.60595	2.22E-05
7	0.627003	0.602945	1.67E-05
8	0.620135	0.601301	1.11E-05
9	0.613042	0.598031	5.56E-06
10	0.608042	0.596855	0

Fig. 15 – Metrics over all epochs for Configuration 1 (5.3.)

Fig. 16 – Training vs Validation Loss Plot for Configuration 1 (5.3.)

Model saved at epoch 10 with **Val Loss: 0.5969**.

Configuration 2:

Model = "speecht5_finetuned"

Batch_size = 4

Learning rate = 5e-5

Num Epochs = 40

Optimizer = AdamW

Scheduler = Linear (with warmup 0.1)

Results 2:

epoch	train_loss	val_loss	lr
1	0.818732	0.706103	1.25E-05
2	0.72281	0.658595	2.50E-05
3	0.693456	0.641433	3.75E-05
4	0.675207	0.635681	5.00E-05
5	0.661065	0.623276	4.86E-05
6	0.645514	0.616221	4.72E-05
7	0.641934	0.610611	4.58E-05
8	0.631626	0.609427	4.44E-05
9	0.622737	0.605854	4.31E-05
10	0.614943	0.604035	4.17E-05
11	0.615677	0.612498	4.03E-05
12	0.605122	0.601509	3.89E-05
13	0.600943	0.600894	3.75E-05
14	0.598143	0.60301	3.61E-05
15	0.597347	0.602579	3.47E-05
16	0.591276	0.597991	3.33E-05
17	0.590525	0.599869	3.19E-05
18	0.585014	0.59803	3.06E-05
19	0.583918	0.597291	2.92E-05
20	0.580026	0.598254	2.78E-05
21	0.576456	0.602811	2.64E-05

Fig. 17 – Metrics over all epochs for Configuration 2 (5.3.)

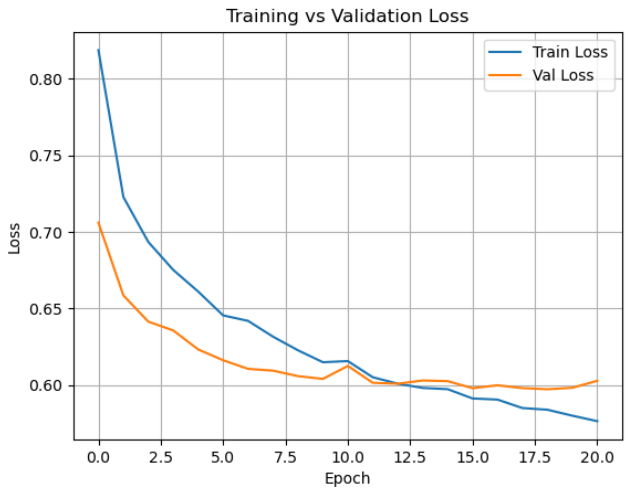

Fig. 18 – Training vs Validation Loss Plot for Configuration 2 (5.3.)

Model saved at epoch 19 with Val **Loss: 0.5973**.

6. Discussion

The experimental results presented in Section 5 highlight the impact of different fine-tuning strategies on the model's ability to generate personalized, speaker-adaptive speech.

Initially, fine-tuning the base SpeechT5 model on the Common Voices 20 dataset (Section 5.1.) proved effective for learning general romanian language structure and phonetics. Among the configurations tested, a learning rate of 5e-5 and 40 training epochs (Configuration 4) yielded the best performance, achieving a validation loss of 0.4351. This confirmed that extended training on a large and diverse dataset leads to more stable convergence and improved linguistic coverage.

When directly fine-tuning the model on the Own Dataset (Section 5.2.), which consists of limited, speaker-specific data, the results were notably less effective, with the best validation loss reaching 0.6082. This reflects the challenge of adapting TTS models when only a small amount of target speaker data is available. The limited exposure to speaker-specific patterns may restrict the model's ability to generalize well, especially without prior adaptation.

To address this, a two-stage fine-tuning approach was applied (Section 5.3.), using the best model from Section 5.1. as a starting point. By transferring linguistic knowledge learned from the larger corpus and further adapting the model on the Own Dataset, we achieved the best validation loss of 0.5969. Although the improvement is modest, it demonstrates the benefit of leveraging pretrained language knowledge before personalizing on limited speaker data.

Furthermore, based on auditory evaluation using the test phrase "Salut! Acesta este un test.", the model from Section 5.1., Configuration 4, demonstrates strong performance, accurately reproducing both the content and vocal characteristics for various speakers from the original dataset. In contrast, the model from Section 5.3., Configuration 2, delivers less precise results, while the speaker identity and intended phrase remain somewhat recognizable, the output lacks the clarity and fidelity required for a fully satisfactory synthesis.

7. Conclusions

This study demonstrates the potential of the SpeechT5 architecture for high-quality, speaker-adaptive Text-to-Speech synthesis. Through a series of fine-tuning experiments, we observed that the model benefits greatly from its pretraining on diverse multimodal data, enabling strong generalization across both linguistic content and acoustic structure.

The results confirm that with the right architectural choices and training configurations, such as optimal learning rates, batch sizes, and early stopping criteria, SpeechT5 can achieve effective convergence and generate intelligible, natural-sounding speech. Notably, the two-stage fine-tuning approach, which transfers linguistic knowledge from a large general dataset before adapting to a specific speaker, provided measurable improvements in validation loss and perceptual quality.

However, the quality of the output remains closely tied to the quantity and diversity of the training data [8]. While the architecture is powerful and flexible, data scarcity in speaker-specific adaptation limits the model's ability to fully capture the nuances of individual voices. Thus, the availability of high-quality, labeled audio remains a key factor in achieving robust, speaker-specific synthesis.

In conclusion, SpeechT5 proves to be a strong foundation for personalized TTS applications, but the effectiveness of adaptation hinges on both strategic training and the richness of available data. Future work could explore data augmentation, semi-supervised training, or speaker embedding enhancements to mitigate data constraints in speaker-specific scenarios.

8. References

[1] – “IMPROVING LANGUAGE MODEL-BASED ZERO-SHOT TEXT-TO-SPEECH SYNTHESIS WITH MULTI-SCALE ACOUSTIC PROMPTS” by Shun Lei et al. - https://arxiv.org/pdf/2309.11977

[2] – “Rethinking MUSHRA: Addressing Modern Challenges in Text-to-Speech Evaluation” by Praveen Srinivasa Varadhan et al. - https://arxiv.org/pdf/2411.12719

[3] – “SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing” by Junyi Ao et al. - https://arxiv.org/pdf/2110.07205

[4] – “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer” by Coli Raffel et al. - https://arxiv.org/pdf/1910.10683

[5] – “SpeechBrain: A General-Purpose Speech Toolkit” by Mirco Ravanelli et al. - https://arxiv.org/pdf/2106.04624

[6] – “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis” by Jungil Kong et al. - https://arxiv.org/pdf/2010.05646

[7] - https://huggingface.co/datasets/TransferRapid/CommonVoices20_ro

[8] – “An Exhaustive Evaluation of TTS- and VC-based Data Augmentation for ASR” by Sewade Ogun et al. - https://arxiv.org/html/2503.08954