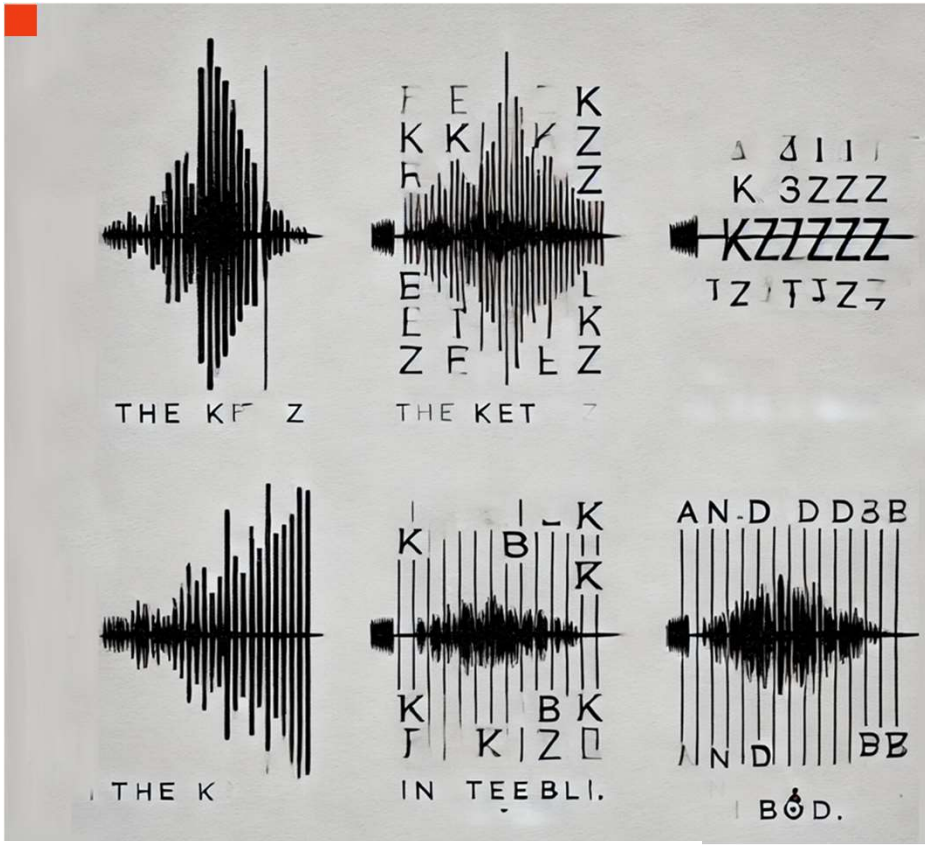


RIVUS VOX EDITOR

Near-live zero-shot adaptive speech editing

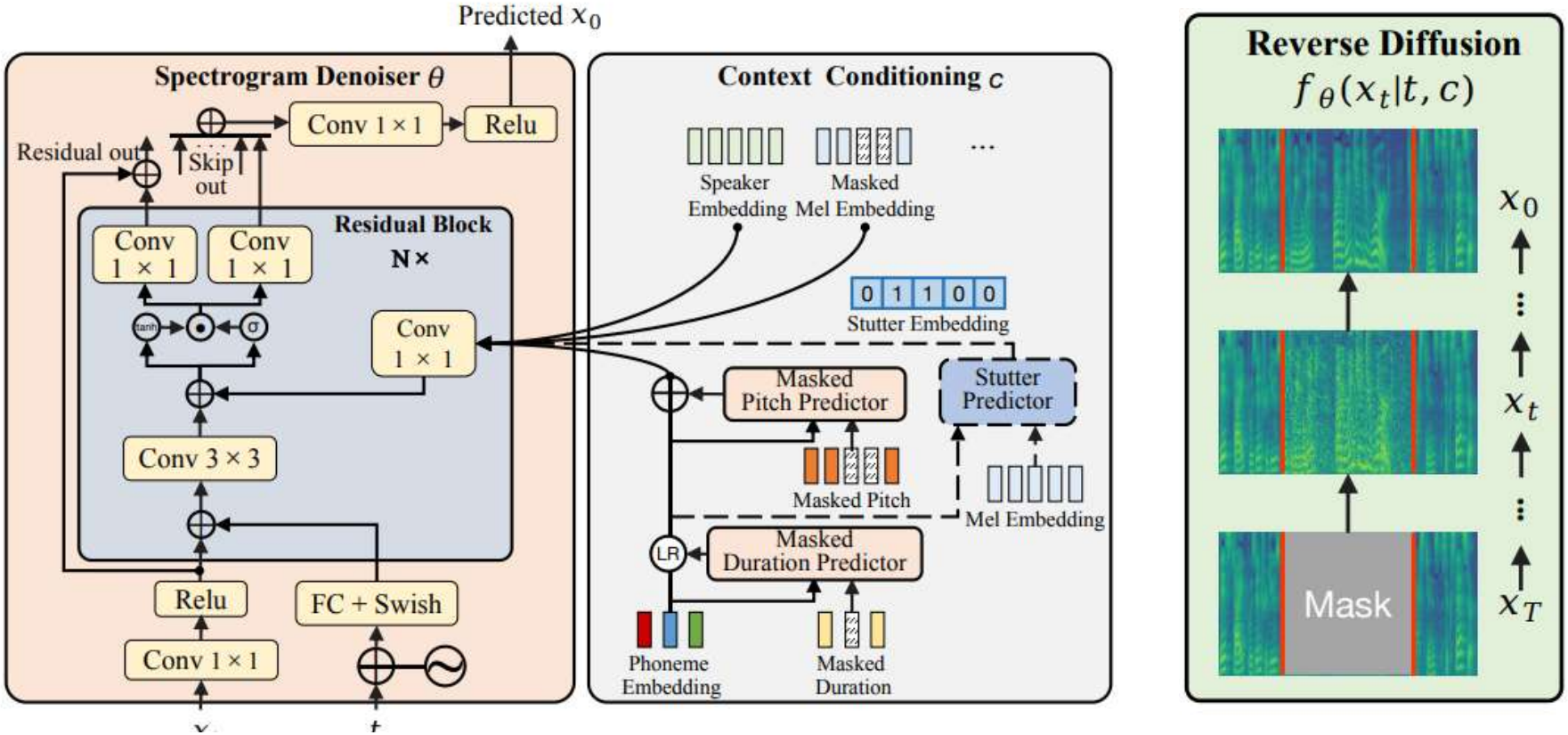
ZACK BEZEMEK
FRANCESCA BALESTRIERI

THE ERDŐS INSTITUTE
Helping PhDs get jobs they love.
Helping you hire the PhDs you need.

MOTIVATION AND MAIN OBJECTIVES

- The main goal behind our project is to be able to edit speech on the fly, by removing key words and replacing them with other words with a generated voice as similar to the speaker's voice as possible, all while having no prior knowledge about the speaker.
- This would be of interest to anyone involved in television, radio, and live streaming, as well as video and sound editors.
- The current publicly available state of the art of zero-shot voice cloning/speech editing – VoiceCraft (2024) and FluentSpeech (2023) – produce high-quality results, but suffer from slow inference times (in our experience, over 90 seconds for inference on a 3 second clip).
- We decided to raise the bar by optimizing one of these models (FluentSpeech) to be able to run inference in real-time (~0.5 seconds for 3 seconds of audio), creating the world's first zero-shot voice-cloning speech editor which runs quickly enough for live stream applications: RivusVox Editor.

THE BASE MODEL: FluentSpeech



OUR CHANGES

- CHANGE 1: modifying the way that short silences are handled. We found that in some cases this alleviates a 'silent phoneme problem.'
- CHANGE 2: replacing MFA with WhisperX and custom wav2vec model for alignment, and substituting out some other vanilla tools being used at different stages of inference to ones that can keep things on GPU.
- CHANGE 3: introducing a hyperparameter which changes how the inferred audio is inserted into the original audio.
- CHANGE 4: having the inferred region automatically adjust until the silent phoneme problem no longer happens.

Loss functions	f0 (fund. frequency)	l1_coarse (l1)	pdur (phoneme duration)	ssim_coarse (ssim)	uv (unvoiced)	wdur (word duration)	total_loss (total loss)
Baseline – MFA GT	0.2075	0.1956	0.0603	0.2001	0.7872	0.2286	1.6791
Change 1 – MFA GT	0.2075	0.1955	0.0596	0.1999	0.7891	0.2238	1.6756
Changes 1 + 2 – MFA GT	0.1942	0.1661	0.1012	0.1793	0.8771	0.6287	2.1466
Changes 1 + 2 – whisper GT	0.1568	0.1663	0.0669	0.1778	0.5023	0.1427	1.2128

Ablation study: validation results on a subset of the LibriTTS Corpus (38 speakers, 3013 utterances)

FINE-TUNING ON INDIVIDUAL SPEAKERS

- If the speech editor is to be used by a specific person, it makes sense to fine-tune some of the components of the model to that specific person.
- The base model was trained on the LibriTTS corpus, consisting mainly of American speakers.
- We decided to fine-tune the model on a female British speaker with a quite distinctive voice – the amazing Amelia Tyler, voicing the Narrator in Baldur's Gate III. Having a speaker with a voice and an accent quite distant from the bulk of the voices used in training was a deliberate choice, in order to test the limits in tailoring the model to such more "extreme" speakers.

THE DATASET FOR FINE-TUNING

- We found a 4 hours video on YouTube of background-noise-free Narrator's lines. We downloaded the audio and pre-processed it by cutting it into chunks based on silences of a certain minimal length and by trimming any trailing silences; for each chunk, we then used Whisper to get a first tentative transcript, which we then cleaned and curated manually, before putting the whole dataset into a format that could be passed to our model for fine-tuning.

- Mean utterance length 4.25s (min 0.28s, max 20.57s)

	Training	Validation
Tot number of utterances	2178	542
Mean number of phonemes including silences (per utterance)	51.16	54.79
Min number of phonemes	4	4
Max number of phonemes	230	210

ADASPEECH CONDITIONAL FINE-TUNING

- We performed a naïve fine-tuning, only adapting the weights of a single bias vector added to the linear layer which encodes the utterance-level speaker projection (256 parameters).
- Following the paradigm outlined in AdaSpeech (https://arxiv.org/pdf/2103.00993), we modify layer normalisations by making them conditional on the speaker embedding. When fine-tuning, we only adapt the model parameters directly related to the conditional layer normalisations. This allows us to fine-tune as few parameters as possible, while still hoping to ensure good adaptation quality.
- In the specific, we are only fine-tuning 74496 parameters out of the 31560867 total parameters in the model, which is 0.23% of the total. What is truly remarkable is that, by conditionally fine-tuning these few selected layers normalisations, the loss functions still decrease considerably on validation, relative to the base model with our modifications.

FINE-TUNING RESULTS

- The results (on the validation set) after fine-tuning for roughly 53k steps are:

Loss functions	f0 (fund. frequency)	l1_coarse (l1)	pdur (phoneme duration)	ssim_coarse (ssim)	uv (unvoiced)	wdur (word duration)	total_loss (total loss)
Baseline (no fine-tuning)	0.2072	0.2393	0.1128	0.2545	1.1229	0.1465	2.0832
Naïve fine-tuning	0.1781	0.1993	0.0868	0.2187	0.5680	0.0889	1.3397
AdaSpeech fine-tuning	0.1482	0.1487	0.0460	0.1670	0.3653	0.0415	0.9167

- We note that, while the fine-tuned voice resembles quite closely the ground truth voice, the phonemes enunciation becomes worse (it sounds like some phonemes get swapped with some others).

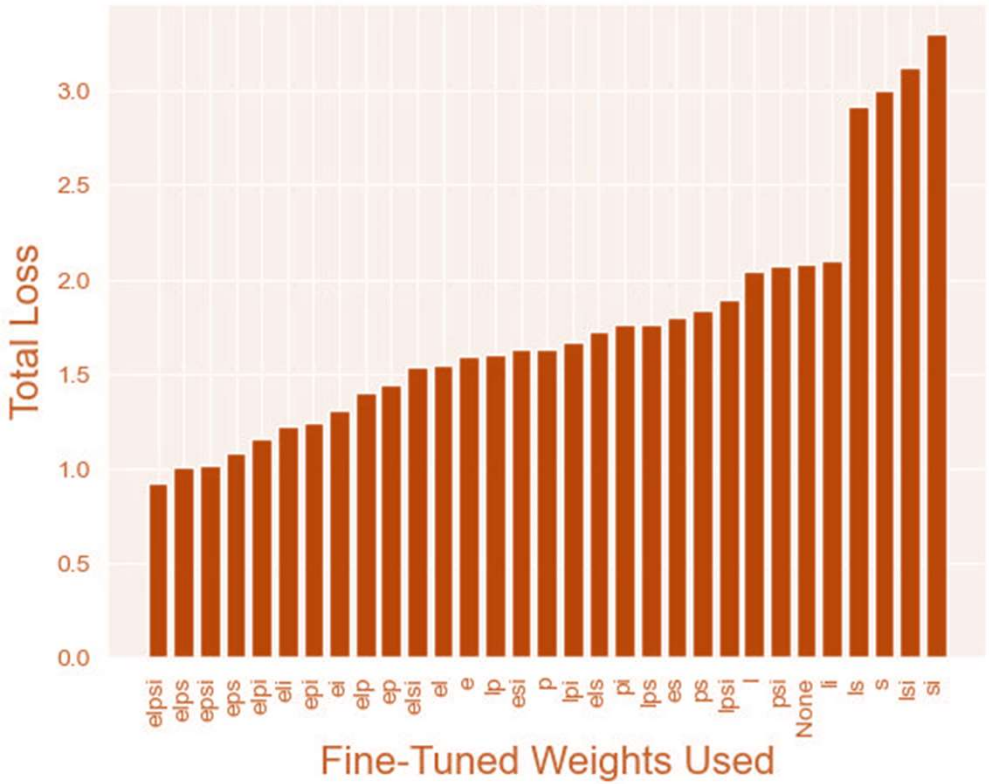
GROUND TRUTH

RECONSTRUCTED (ADASPEECH)

- We speculate that this might be due to the difference in phoneme pronunciation between American and British speakers. In the base model, phoneme processing components are trained on mainly American speakers. With our fine-tuning, we have changed the weights of the encoder, but not those of the decoder.

ABLATION STUDY ON FINE-TUNING

- After fine-tuning AdaSpeech-style, we performed an ablation study by reverting back to their non-fine-tuned values different combinations of groups of weights. The following picture shows the total loss on the validation set for the different combinations.



LEGEND
e: speaker embed layer normalisations
l: duration predictor layer normalisations
p: pitch predictor layer normalisations
s: utterance-level speaker embed linear layer
i: speaker id weight

NOTE: If a letter appears, it means that we used the fine-tuned values; if it doesn't appear, it means that we reverted back to the non-fine-tuned values.

LIVE SPEECH EDITING

- Exploiting the near-live inference time of our speech editing model (roughly 0.5s for a 3s audio clip), we built a live speech editing app, which simulates a live streaming scenario, where the audio (and video) input is processed chunk by chunk.
- The user can specify some key words to be edited and, with a small delay, the app processes the input, checks if any of the key words has been found, and, if so, using our model, it changes the words to the desired targets, and outputs the (potentially edited) audio, together with the transcript of the current audio segment, its spectrogram, and video frames (if video is used). The input can come from local .wav and .mp4 files, from a YouTube link, or from the user's microphone and camera.
- We used the Tkinter module to create the app's GUI, and threading to coordinate and run in parallel the various background processes needed for the app to work.

FUTURE WORK

- Properly clean the code (there are lots of vestigial remains from the FluentSpeech code that we don't actually use) and better integrate the various models (to eliminate redundancies).
- Conditionally fine-tune the phonemes decoder (instead of just the encoder) as well, to improve the quality of the generated edited speech when tailoring our model to an individual speaker, in order to solve some phoneme enunciation problems encountered.
- Include a loss which attempts to improve the smoothness of the mel spectrogram boundary (like in FluentEditor).
- Use WhisperX as ground truth phoneme alignment during training. This would implicitly also allow for different languages if we also use it as our phoneme tokenizer.
- Consistency models to speed up spectrogram denoiser.
- AdaSpeech-style conditional layer norms for speaker ids.
- Improve the stability and expand the range of functionalities of the live-streaming speech editing app.