Conditional VAE to convert text to Emotional speech

Shivam

1. Introduction :

The advent of deep learning revolutionized the field of TTS with models like Tacotron2[1] that combined Tacotron's spectrogram prediction with a WaveNet[3] vocoder, achieving near-human naturalness. Parallel to TTS, **voice conversion** research transforming one speaker or style into another advanced via Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs)[4]. The **Conditional VAE** framework extended VAEs by introducing label-based conditioning, facilitating attribute-guided generation in images and audio. [5]

This project develops a pipeline for converting text into emotionally expressive speech, with a focus on "Happy" and "Angry" emotions. To achieve this, we use 2 models: First model used here is google's Text to speech model.[6] It takes an input text which is converted into neutral speech audio using Google's gTTS API. Next, we perform emotion conversion via audio-domain style transfer in the audio domain. In this stage, given the audio file of neutral speech, we learn to apply an "emotional style" (happiness or anger). To do this, we implement a Conditional Variational Autoencoder (cVAE) in PyTorch that operates directly on mel-spectrogram representations. By conditioning on discrete emotion labels, our cVAE tries to learn to disentangle content and style in its latent space and to reconstruct neutral spectrograms into their emotionally styled counterparts.

2. Datasets

1.1 Datasets used

Since we use pretrained model for TTS, we are not training it on any dataset. To train the 2nd model(cVAE), we use the Audio emotions[7] dataset. The dataset contains audio files of variable length collected and combined from multiple sources for 7 different emotions. We filter out the audio files for "Happy" and "Angry" emotions. Our cVAE requires 3 inputs to train and learn the emotions: Neutral audio spectrogram, Emotional audio spectrogram and Emotion label(Happy or sad). From the audio emotions dataset we already have Emotional audio spectrogram and Emotion label(Happy or sad). This provided us with 2167 records of each label. (Total 4334 records). But we still need the neutral audio spectrogram corresponding to each emotional audio spectrogram.

2.2 Generation of neutral dataset

We generate the neutral dataset by converting the Emotional audio to text and then converting the text to audio using gtts. To convert audio to text we use OpenAI whisper model. We use the whisper-large for the highest accuracy while generating the text. Whisper can sometimes output non-English tokens or artifacts. We scan the transcriptions with langdetect[8] to ensure they're English removing 35 files that were not in english. This guarantees our neutral TTS stage only handles English utterances and prevents garbage-to-speech cases. Next, we use gtts to convert all of these text files to audio. To optimize gtts API usage, we compute an MD5 hash of each text file; identical transcripts reuse existing MP3s via copy previously created files to new ones rather than repeating the HTTP call. Finally, since all the audio by gtts were generated in the female speaker mode, we increase the pitch of the audio files using pydub library from python. This makes the audio generated more gender neutral. To keep the input of the model consistent all the audio files greater than 5 seconds were removed. This led to the removal of 61 files. This left us with a total of 4238 file pairs(Fig 2.1 shows spectrograms of 1 such pair) for training. Trimming/padding is applied to the waveforms to 5 seconds uniformly. Each neutral MP3 is then converted to a mel-spectrogram with the same STFT parameters as the emotional audio datasets(SAMPLE_RATE = 22050 ,N_FFT = 2048, HOP_LENGTH = 512), giving us the triplet *(neutral spectrogram, emotional spectrogram, emotion label)* that we feed into the cVAE for training. Each spectrogram individually to zero mean and unit variance. The shape of the spectrograms was 80,216 corresponding to (N_MELS, time_frames).

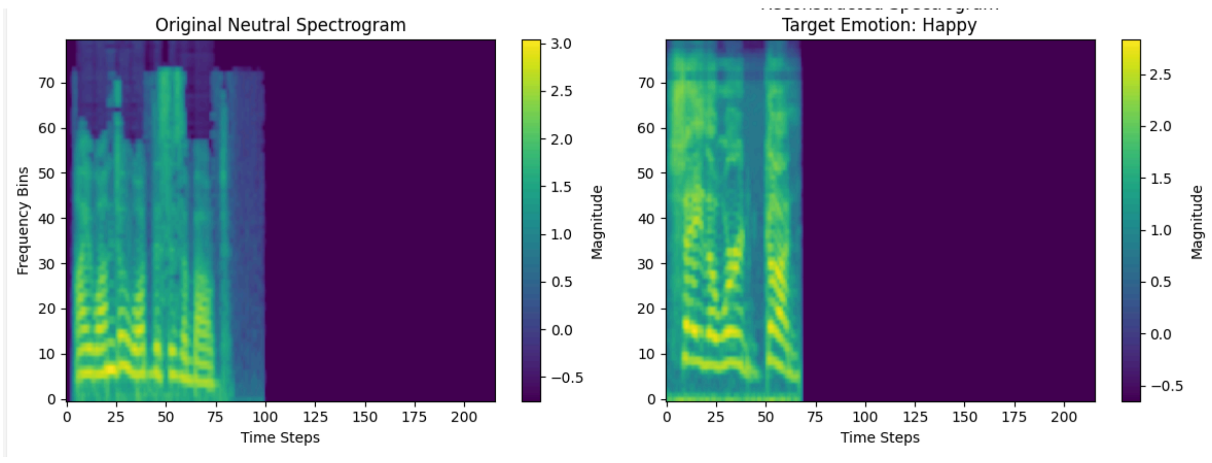


Fig 2.1 : Neutral and happy spectrograms for the same speech.

3. Method

To achieve the task of text to conditioning based emotion we use 2 models: Google's TTS to convert text to speech and a cVAE. In this section, we will see how these models were trained to convert text to emotion infused speech.

3.1 Google TTS model

Our gTTS wrapper leverages Google's proprietary TTS API, which internally runs a two-stage neural synthesis: (1) a Tacotron-style sequence-to-sequence model that converts text into mel-spectrograms, and (2) a WaveNet-based vocoder that transforms spectrograms into high-fidelity waveforms. Together, these components enable human-like prosody, timbre, and intelligibility without requiring on-device training.

3.2 cVAE

The training loop for our Conditional Variational Autoencoder (cVAE) begins by sampling minibatch triplets of batch size=16 (x_neutral,x_emotion,c) where are x_neutral,x_emotion are mel-spectrogram tensors of shape [B,1,H,W] (B=16, H=80,W=216), and c is the one-hot emotion label. Each forward pass first computes the encoder's approximate posterior parameters $\mu, \log\sigma^2$. The encoder sequentially applies strided convolutions followed by adaptive instance normalization (AdaIN) layers[9] that inject style information at every feature map. The reparameterization trick then draws a latent sample ensuring proper flow of gradients and creates a latent vector(z)

Decoding concatenates z with c into a unified code z_c, projects it via a linear layer into a feature map, and applies transposed convolutions plus AdaIN to reconstruct x_i.

Training minimizes the sum of a reconstruction loss and a KL (Kullback Leibler) divergence. The reconstruction loss is the mean absolute error between x_i and x_out (Comparing the model's output to the target emotion spectrogram). This encourages the cVAE to match spectral details of genuine emotional speech. penalizes deviations of the approximate posterior from the prior distribution. By minimizing this KL divergence, the encoder is discouraged from collapsing the outputs into irregular shapes, instead learning a smooth latent manifold where samples of z remain broadly distributed around zero with unit variance.

The total loss is back-propagated through all AdaIN, convolutional, and linear layers, and parameters are updated via Adam optimization. As training proceeds, the encoder learns to map neutral spectrograms into a latent manifold where samples cluster by emotion label, while the decoder learns to impose the corresponding style onto neutral content. Ideally, at convergence, feeding a neutral spectrogram with a novel label such as swapping "Happy" to "Angry" should yield a spectrogram whose prosody and spectral envelope convincingly mimic the target emotion, thereby demonstrating effective audio domain style transfer. Unfortunately, this was not the case. We will continue this discussion in the results section.

3.3 Spectrogram to audio

After receiving the transformed spectrogram from the model, it needs to be converted to audio waveform. The spectrograms are first inverse normalized to target decibel boundaries(Chosen to be -150 to 50 after checking the actual range of the input spectrograms before normalization). Converting back to decibels is necessary before magnitude inversion: the mel-spectrogram values must represent realistic log-amplitudes for proper Griffin–Lim convergence. With similar parameters of STFT magnitude which were used during spectrogram creation before training, the code employs the Griffin–Lim algorithm to iteratively recover phase information. Starting from random phase initialization, each iteration computes an inverse STFT to the time domain, re-STFTs the signal to recompute magnitude, and replaces its magnitude with the target while retaining the new phase estimate. This was done for 60 iterations.

4. Experiments and evaluation

4.1 gtts evaluation

As discussed in section 2.2, gtts was used to create the neutral audio dataset from text files(Let's call these text_pre). To evaluate the accuracy of gtts, 200 files from the audio datasets(100 from each emotion) were converted back to text(text_post) using openAI whisper model in the same way as discussed in section 2.2. Text_post and text_pre are compared for the following :

i) Exact Match Accuracy is the percentage of utterances where the cleaned prediction exactly equals the reference. ii) Word Error Rate (WER) computes the fraction of word‑level edits (substitutions + insertions + deletions) over total words. iii) Character Error Rate (CER) is the same at the character level. iv) BLEU measures n-gram overlap between hypothesis and reference (0–1 scale), with smoothing to handle short texts.

Table 4.1 results of evaluation on 200 files:

Condition	Total Files	Exact Match Accuracy	Average WER	Average CER	Average BLEU
Happy (no ?/!)	100	88.00%	4.17%	2.99%	0.86
Angry (no ?/!)	100	90.00%	3.08%	2.20%	0.89
Happy (with ?/!)	100	71.00%	8.77%	3.75%	0.79
Angry (with ?/!)	100	68.00%	8.72%	3.19%	0.8

Table 4.1 : Accuracy metrics for gtts+whisper combined

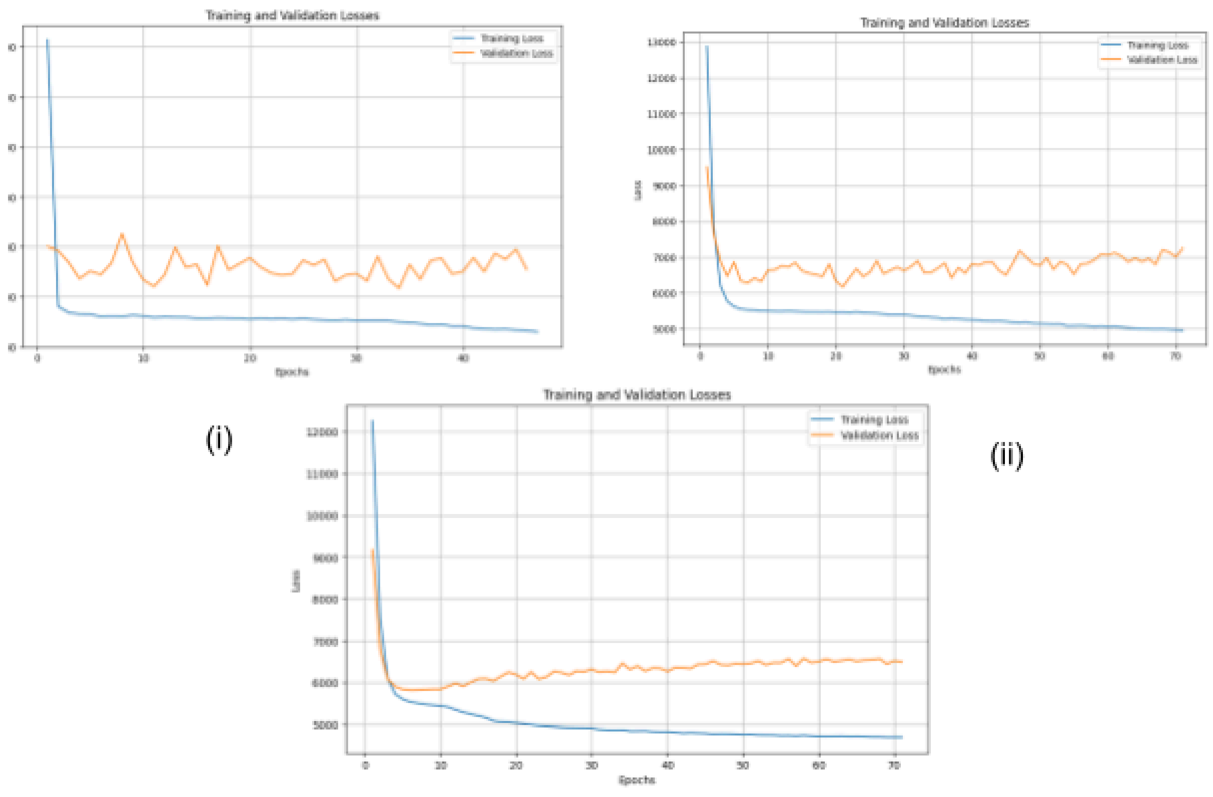
It can be seen that without punctuation(? and !), the system performs approximately 20% better in exact matches. The reason for this is that, being that when whisper is good at catching exclaimation and identifying questions when transcribing the audio dataset audio files. However, gtts does not have this ability when converting text to audio and creates a flat, punctuation agnostic audio instead. This is why when the gtts outputs are converted back to texts the punctuations(? And !) are missing. This highlights an important disability of the gtts model lacking expressiveness. This is another example application of audio style transfer, similar to the one we are working on.

It is important to note, that the results reported in table 4.1 are the combined results for the gtts+whisper and not just gtts alone. However, they are a good estimate of gtts performance.

4.2 cvae

The cvae was run with multiple architectures with different depths. However, none of these architectures returned satisfactory results. The models were highly underfit after training. The emotion transformed audio files had no audible speech in any of the cases.

Figure 4.1 shows the loss curves of 3 main architecture variation.

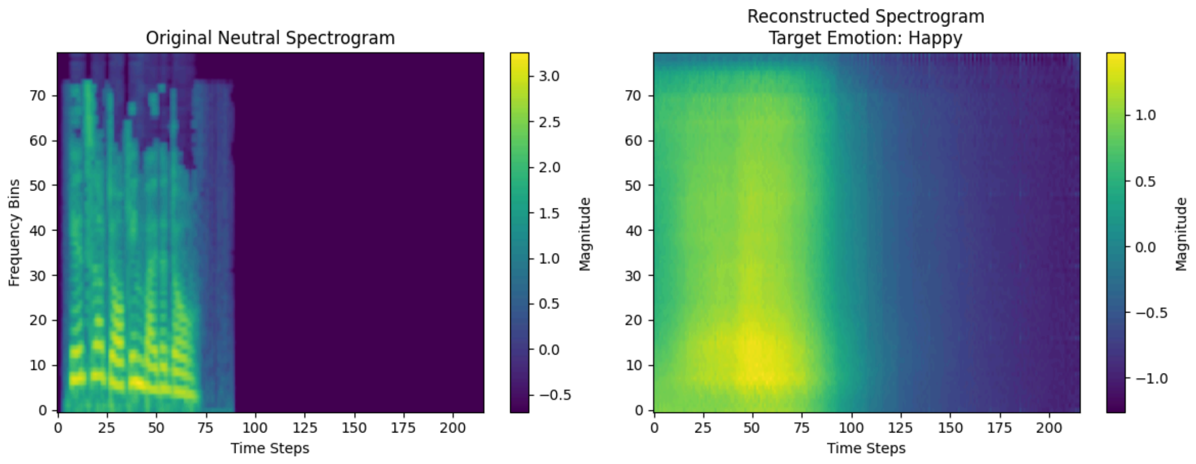


(iii)

Fig 4.1 : i) Loss curve without AdaIN in either encoder or decoder. ii) Loss curve with AdaIN in encoder only. iii) Loss curve with AdaIN in encoder and decoder both.(fully conditioned)

AdaIN is a normalization technique that aligns the mean and variance of content features with style features. It is a good practice to use AdaIN in the decoder. This helps the emotion to influence every layer of the decoder and allows for better feature modulation based on emotional characteristics. Using AdaIN in the encoder has it's pros and cons. Pro : The network now extracts different features based on the emotion, potentially capturing emotion-specific characteristics better. Cons : The latent space may now encode a mix of content and emotion rather than pure content. This could make the disentanglement between content and emotion less clear.

The results show that a fully conditioned network, with AdaIN in both encoder and the decoder has slightly better loss curve than the other two as seen in figure 4.1. However, all 3 model types are underfit. Fig 4.2 shows a transformed output from the fully conditioned model. It can be seen the outputs have no nuances similar to that of speech.



To evaluate the model output, speech emotion recognition model[10] can be run on the output of the model.

5. Discussion

The attempt to create a text-to-emotional speech pipeline using a two-stage approach (gTTS followed by emotion style transfer via cVAE) encountered significant challenges, primarily in the emotion conversion stage. Despite implementing various architectural modifications including different depths and conditional mechanisms through AdaIN layers, the cVAE models consistently underfitted and failed to produce intelligible emotional speech. Several factors likely contributed to these limitations.

Data and Alignment: This is the major issue that leads to the cVAE performing poorly. Each neutral (synthesized via gTTS) and emotional clip pair has no guaranteed lexical or temporal alignment. This is visible in the comparision of spectrograms(Fig 2.1). Non-parallel training tends to confuse the model, because it must learn to map arbitrary neutral content to emotional counterparts without explicit frame-level supervision.

Limited and Noisy Data : Our filtered subset (4,238 utterances across two emotions) is modest compared to benchmarks like ESD. However, processing such large datasets also demands storage and computational efficiency.

Spectrogram Inversion Artifacts: Using Griffin–Lim to invert mel-spectrograms introduces phase reconstruction errors and muffled artifacts, masking any subtle emotional characteristics of speech learned by the model. Neural vocoders (WaveNet, WaveGlow) are typically required to hear nuanced style changes.

Future work : **Two-stage training** can be used to First train an autoencoder to reconstruct speech accurately, then fine-tune with emotion conditioning might preserve speech intelligibility better. Forced-aligner can used to align phoneme timestamps in emotional recordings to the neutral transcripts, then apply time‑scale modification to neutral audio to match durations. Adding noise, pitch shifts, or time-stretch augmentations at training time to improve model generalization to real-world audio variability.

References

[1] https://pytorch.org/hub/nvidia_deeplearningexamples_tacotron2/
[2] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, “Tacotron: Towards end-to-end speech synthesis,”
[3] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “WaveNet: A generative model for raw audio,”
[4] Tero Karras, Samuli Laine, Timo Aila , “A Style-Based Generator Architecture for Generative Adversarial Networks”
[5] MEI LI, JIAJUN ZHANG, and XIANG LU , “CHENGQING ZONG” , “Dual-View Conditional Variational Auto-Encoder for Emotional Dialogue Generation”
[6] https://cloud.google.com/text-to-speech?hl=en
[7] https://huggingface.co/docs/transformers//model_doc/whisper
[8] https://pypi.org/project/langdetect/
[9] Ziye Zhang, Li Sun, Zhilin Zheng, Qingli Li “Disentangling the Spatial Structure and Style in Conditional VAE”
[10]
https://huggingface.co/ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition