Neural Voice Cloning Using Fusion Models with Multi-Head Attention Mechanism

by

Ojas Sharma &

Prakhar Pathak

Submitted to

Dr. ANURAG TIWARI

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

December 2023

Abstract

Voice cloning is a highly sought-after feature for personalized speech interfaces. Neural network-based speech synthesis has been shown to generate high-quality speech for a large number of speakers. In this project, we explore a neural voice cloning system that takes a few audio samples as input. Our approach includes speaker adaptation and speaker encoding. Speaker adaptation is based on fine-tuning a multi-speaker generative model with a few cloning samples. Speaker encoding is based on training a separate model to directly infer a new speaker embedding from cloning audios, which is used in a multi-speaker generative model. We aim to build a neural architecture that has a fusion model and a multi-head attention system to provide emphasis to different parts of speech. A great part of the project is inspired by and is a practical implementation of the base paper “Neural Fusion for Voice Cloning”.The research presents a novel approach to voice cloning using a neural fusion model integrated with a multi-head attention mechanism. This innovative architecture addresses the challenges of voice cloning with limited training data by leveraging the unit concatenation method and attention mechanisms. The incorporation of multi-head attention enhances the model's ability to capture complex dependencies and improve the synthesis of natural and coherent speech. The fusion of these techniques results in a significant improvement in speaker similarity and speech naturalness. Overall, the research demonstrates the potential of the proposed neural fusion model with multi-head attention for advancing voice cloning applications, particularly in scenarios with restricted training data.

Table of Contents

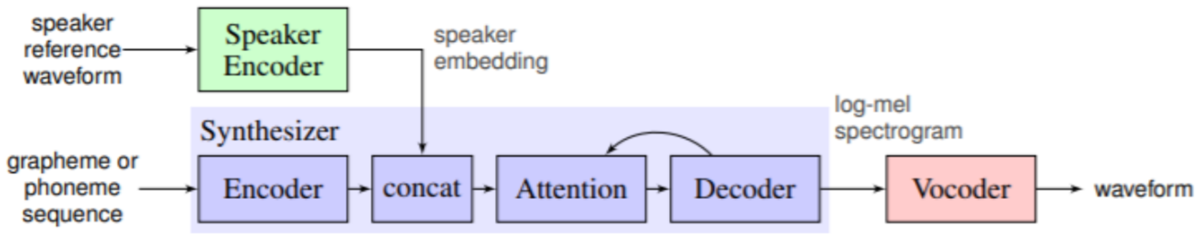
List of Figures

1. Introduction

1.1 Voice Cloning

Voice Cloning is a revolutionary technology that has captivated both the scientific community and the general public. Fundamentally, this breakthrough is about using large datasets of a person's recorded speech to train neural networks so that the system can recognise and mimic the individual's own vocal features. Generative adversarial networks, or GANs, are used to help these models improve their output by continuously pushing themselves to produce more realistic imitations while receiving feedback from a discriminator.

The end product is a synthesized voice that can accurately replicate the original speaker's intonation, cadence, and even subtle emotional changes. The need for responsible development and implementation of this technology is highlighted as we learn more about the physics of AI voice cloning and its possible applications.



1.2 Neural Fusion

The model of Neural fusion in a text to speech system is a model that uses unit concatenation method along with a parametric text-to-speech model.This is really helpful when there's not much training data available. The model has three main parts: one that understands the written words, one that turns them into sounds, and another that helps maintain a consistent speaker style. It uses special techniques to make the computer-generated speech sound more like a specific person, even if there's not a lot of information to learn from. The whole thing is built like a single, connected system and aims to make sure the spoken words sound really good and similar to the original speaker, even if there wasn't much information for the computer to learn from.

1.3 Multi Head Attention Model

Adding an attention mechanism to the neural fusion model can make it better at paying attention to specific parts of what's being said when it's generating speech. Here's how it works:

Imagine the model as having two main parts, one that understands the words and another that turns them into sounds. By adding attention, you're giving the sound part a way to zoom in on different parts of what's being said.To do this, the model figures out which parts of the words are important for the sounds it's making. It does this by calculating attention weights, which are like importance scores for each word. It can use different methods to decide how much attention to give to each word.Once it knows what to focus on, the model uses these attention weights to create a sort of summary called a context vector. This summary captures the important information from the words, and then the sound part uses it to generate the spoken words.During training, the model learns how to do this by looking at examples of written words and their corresponding spoken versions.By using attention, the neural fusion model becomes more flexible, adjusting its focus as it goes along, and this helps it sound better by capturing the tone and style of the speaker in the computer-generated speech.

1.4 Unit Concatenation

Text-to-speech systems employ the unit concatenation approach as one of its techniques. It involves putting together smaller speech components, such diphones or phonemes, to form words or phrases. The required speech output is formed by concatenating these smaller pieces together.

Phoneme-level unit embeddings taken from real audio data are selected and concatenated using the unit concatenation method within the neural fusion model. To solve the concatenation problem, these unit embeddings are then combined in a sequential manner with sampled embeddings. The process includes techniques to choose the best units for concatenation and to narrow the distance between adjacent embeddings.

All things considered, the unit concatenation approach is essential to the synthesis of speech since it allows the model to efficiently integrate smaller speech units to provide coherent and natural-sounding speech output.

1.5 Objectives and Scope of Project

Objectives:

- To develop a neural fusion model integrated with a multi-head attention mechanism for voice cloning applications.
- Enhance speaker similarity and speech naturalness in synthesized speech by leveraging the unit concatenation method and attention mechanisms.
- To address the challenges of voice cloning with limited training data by improving the model's ability to capture complex dependencies as well as prosodic features.
- Evaluate the performance of the neural fusion model with multi-head attention through various evaluation parameters.

Scope:

- Integration of the unit concatenation method and attention mechanisms to enable the model to capture and reproduce speaker-specific characteristics.
- Evaluation of the model's performance using diverse datasets and benchmarking against existing voice cloning techniques.
- Analysis of the synthesized speech quality, speaker similarity, and robustness in scenarios with limited training data.
- Exploration of potential applications in text-to-speech systems, voice conversion, and personalized speech generation.
- The project aims to advance the state-of-the-art in voice cloning by leveraging neural fusion with multi-head attention, with a focus on improving the fidelity and naturalness of synthesized speech while addressing the challenges associated with limited training data.

1.5 Overview of Methodologies Used

The methodologies used for the project "neural voice cloning using neural fusion with multi-head attention mechanism" encompass advanced techniques in deep learning, speech processing, and attention mechanisms. The methods overview consists of:

1. Neural Fusion Model Development:
    - Implementation of a neural fusion architecture tailored for voice cloning applications, integrating components such as text encoders, acoustic decoders, and phoneme-level reference encoders.
    - Use of the unit concatenation method to combine phoneme-level unit embeddings extracted from real audio data, enabling the model to

capture speaker-specific characteristics.

2. Multi-Head Attention Mechanism:
    - Integration of multi-head attention mechanisms to enable the model to capture complex dependencies and focus on relevant parts of the input sequence during speech synthesis.
    - Utilization of attention weights and context vectors to enhance the model's ability to reproduce prosodic features and improve speech naturalness.
3. Refinement Procedures:
    - Implementation of refinement procedures to mitigate concatenation problems and enhance the naturalness of synthesized speech.
    - Training the model to learn the attention weights and context vector generation based on the input-output pairs in the training data.
4. Evaluation and Analysis:
    - Comprehensive quantitative and qualitative assessments of the model's performance, including speaker similarity, speech naturalness, and robustness in scenarios with limited training data.
    - Benchmarking the synthesized speech quality against existing voice cloning techniques and analyzing the potential applications in text-to-speech systems, voice conversion, and personalized speech generation.

The methodologies employed in this project aim to leverage cutting-edge advancements in deep learning and attention mechanisms to address the challenges of voice cloning with limited training data, ultimately advancing the state-of-the-art in high-fidelity speech synthesis and personalized speech generation.

Literature Review

2.1 Unit Concatenation

The hybrid unit concatenation system is a widely used method for text-to-speech. For many years, there have been techniques for creating unit concatenation systems using parametric models. The HMM-based parametric text-to-speech model was the one that initially presented the hybrid system [1]. Concatenative systems and statistical systems have already been paired with HMM models in some publications [2], [3]. In [4], [5], the DNN-based hybrid system takes the role of the HMM-based hybrid system. These hybrid systems mostly adhere to [5]'s fundamental structure. We also use this strategy to parametric end-to-end systems. For the purpose of a unit selection text-to-speech system, the unit embeddings are taken out of Tacotron encoders [6]. Recently, the UnitNet was proposed for text-to-speech concatenation [7]. In addition to measuring the dependence between succeeding units, it extracts a hierarchical architecture for phonemesize units. The phoneme-level unit embeddings are taken from the audios and contexts as phoneme-level unit representations in these end-to-end based concatenated speech synthesis (CSS) models [7]. The unit sequence for concatenation is then chosen using the phoneme-level representations in the dynamic processing search. We use an end-to-end network in a similar manner to extract the unit embeddings. In contrast, the unit is included to improve the voice cloning's speaker resemblance. There has never been any research done on combining the parametric model's units during the synthesis phase. We fuse the unit embeddings with the inference embeddings after running the selection step at the prosody embedding level.

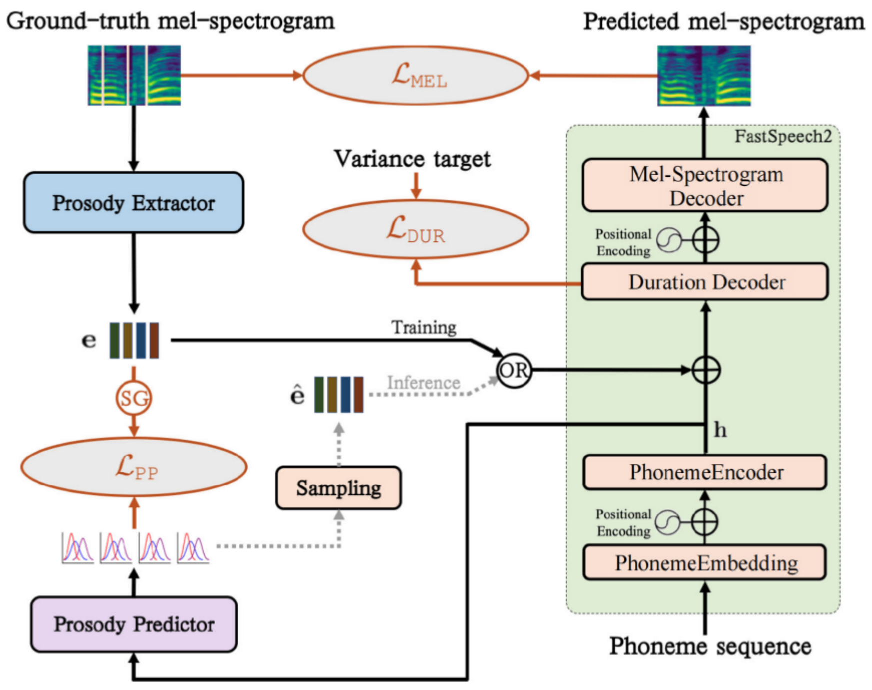


Fig. 2. The architecture of the GMM-based MDN phoneme conditioned FastSpeech2.

2.2 Phoneme-Level Conditioned FastSpeech2

One of the most advanced end-to-end text-to-speech models is FastSpeech2 [28]. To estimate the phoneme-level duration, the phoneme sequences are encoded into phoneme-level hidden embeddings. Next, in order to gradually predict frame-level prosody characteristics (pitch, energy) and spectrograms, the hidden embeddings are extended to frame-level embeddings using the phoneme duration. This design performs well when modeling spectrograms. Du et al suggested a GMM-based mixture density network to predict the characteristics of the GMM distribution, including means, variances, and mixture weights based on conditional FastSpeech2, since speakers have varied speaking styles for the same phoneme and text [20]. It is assumed that the prosody of phonemes is rich and

needs a certain level of complexity in order to be modeled. Based on this supposition, it is feasible that the unit may be found near a particular prosody of the mixture distribution in one or more probable phoneme prosodies.
The architecture of the mixture density network (MDN)-based GMM-based phoneme-level prosody modeling in conditioned FastSpeech2 [20] is seen in Fig. 2. The phoneme encoder, prosody extractor/predictor, and duration/mel-spectrogram decoder are the three components that make up the model. Fig. 3 displays the prosody extractor and prosody predictor in detail.

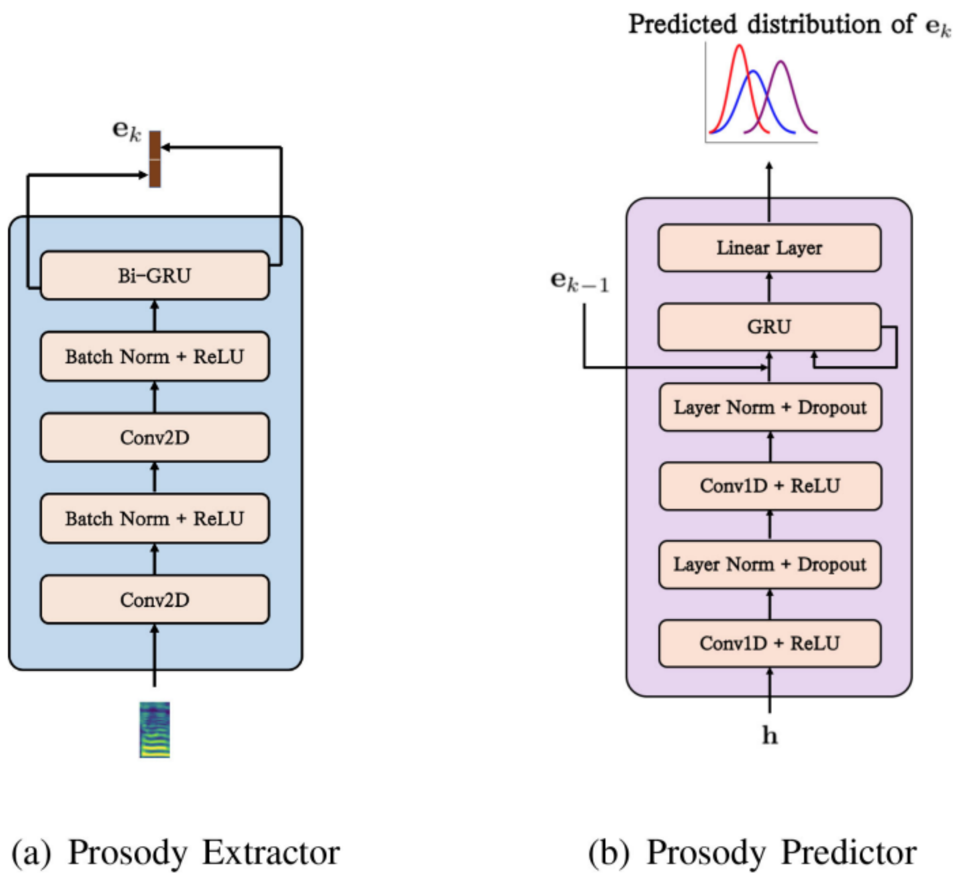


Fig. 3. Architecture of the prosody extractor and prosody predictor.

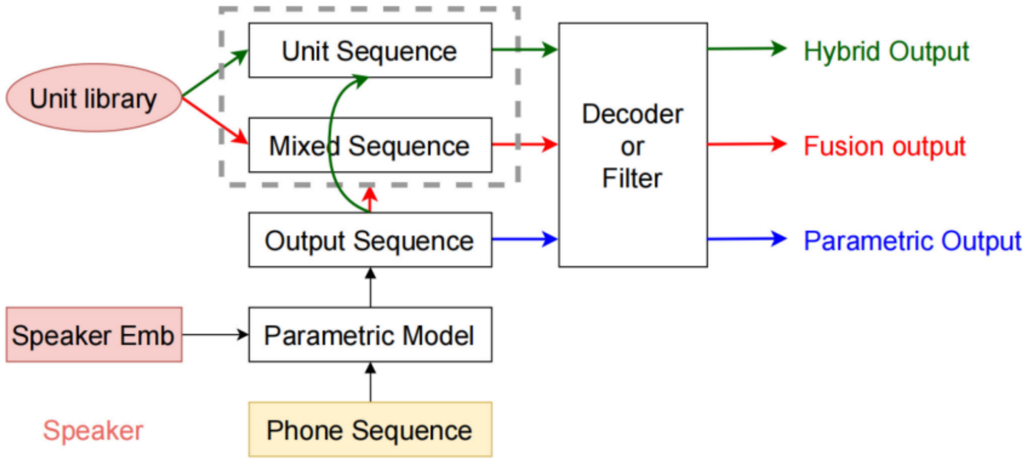


Fig. 4. The brief architecture of the fusion system compared to the hybrid concatenation system and the parametric system. The data flows are shown in green, red, and blue lines for different systems.

2.3 Neural Fusion for Voice Cloning [8]

In this paper, authors studied two approaches for neural voice cloning: speaker adaptation and speaker encoding. They demonstrate that both approaches can achieve good cloning quality even with only a few cloning audios. For naturalness, they showed that both speaker adaptation and speaker encoding can achieve an MOS for naturalness similar to the baseline multi-speaker generative model. Thus, the proposed techniques can potentially be improved with better multi-speaker models in the future (such as replacing GriffinLim with WaveNet vocoder as in Shen et al. (2017a)). For similarity, they demonstrated that both approaches benefit from a larger number of cloning audios. The performance gap between whole-model and embedding-only adaptation indicates that some discriminative speaker information still exists in the generative model besides speaker embeddings. The benefit of compact representation via embeddings is fast cloning and small footprint size per user. Especially for the applications with resource constraints, these practical considerations should clearly favor for the use of speaker encoding approach. Methods to fully embed the speaker information into the embeddings would be an important research direction to improve performance of voice cloning. They observed the drawbacks of training the multi-speaker generative model using a speech recognition dataset with low-quality audios and limited diversity in representation of universal set of speakers. Improvements in the quality of dataset will result in higher naturalness and similarity of generated samples. Also, increasing the amount and diversity of speakers should enable a more meaningful speaker embedding space, which can improve the similarity obtained by both approaches. We expect our techniques to benefit significantly from a large-scale and high-quality multi-speaker speech dataset.

3. Methodology

3.1 Datasets Used

LibriSpeech[9] dataset :- which contains audios for 2484 speakers sampled at 16 KHz, totalling 820 hours. LibriSpeech is a dataset for automatic speech recognition

VCTK dataset[10] :-VCTK consists of audios for 108 native speakers of English with various accents sampled at 48 KHz.In the VCTK dataset - 84 speakers are used for training of the multi-speaker generative model

3.2 Neural Fusion Model Development:

The project involves the development of a neural fusion architecture, integrating a unit concatenation method and a multi-head attention mechanism. This includes the implementation of text encoders, acoustic decoders, and phoneme-level reference encoders within the model.

3.3 Multi-Head Attention Mechanism:

The integration of multi-head attention mechanisms involves the use of advanced attention algorithms to capture complex dependencies and focus on relevant parts of the input sequence during speech synthesis. This includes the utilization of attention weights and context vectors to enhance the model's ability to reproduce prosodic features and improve speech naturalness.

3.4 Refinement Procedures:

Refinement procedures are implemented to mitigate concatenation problems and enhance the naturalness of synthesized speech. This involves training the model to learn attention weights and context vector generation based on the input-output pairs in the training data.

3.5 Algorithms and Tools:

We are using the deep learning framework PyTorch for implementation of model architecture and all the preprocessing steps required.

Advanced algorithms for unit concatenation, multi-head attention, and speech synthesis are employed to achieve the project objectives.

3.6 Data Collection and Preprocessing:

The project may involve the collection of audio data samples from various sources like kaggle.

Preprocessing techniques such as audio feature extraction, phoneme segmentation, and normalization may be applied to the collected data to prepare it for training the neural fusion model.

3.7 Evaluation and Analysis:

Comprehensive quantitative and qualitative assessments of the model's performance are conducted, including speaker similarity, speech naturalness, and robustness in scenarios with limited training data.

Benchmarking the synthesized speech quality against existing voice cloning techniques and analyzing potential applications in text-to-speech systems, voice conversion, and personalized speech generation.

The methodologies employed in this project aim to leverage cutting-edge advancements in deep learning, attention mechanisms, and speech processing to address the challenges of voice cloning with limited training data, ultimately advancing the state-of-the-art in high-fidelity speech synthesis and personalized speech generation.

IMPLEMENTATION

We now will discuss the description of how methodologies were implemented in this project.

4.1 DESCRIPTION

We examine a generative model capable of handling multiple speakers, denoted as f(ti,j , si; W, esi). This model receives input in the form of text (ti,j) and the identity of the speaker (si). The parameters subject to training are represented by W, and esi represents the trainable speaker embedding corresponding to si. The optimization process involves adjusting both W and esi by minimizing a loss function (L). This loss function penalizes discrepancies between the generated audio and the actual audio ground truth. For instance, it may involve a regression loss applied to spectrograms.

Speaker embeddings, demonstrated to effectively capture speaker distinctions in multi-speaker speech synthesis, serve as low-dimensional continuous representations of speaker characteristics. Despite being trained with a purely generative loss, these embeddings exhibit discriminative properties such as gender or accent in the embedding space .

Voice cloning aims to derive speaker characteristics for an unseen speaker sk (not within S) from a set of cloning audios Ask, generating a distinct audio based on a provided text for that speaker. The two key performance metrics for the generated audio include its naturalness and whether it exhibits the characteristics of the same speaker.

4.2 Proposed Algorithm

4.2.1 Speaker adaptation

The concept of speaker adaptation involves refining a pre-trained multi-speaker model for an unfamiliar speaker by utilizing a small set of audio samples and their corresponding texts through the application of gradient descent. This fine-tuning process can be implemented on either the speaker embedding or the entire model. For embedding-only adaptation, we have the following objective:

$$\min_{\mathbf{e}_{s_k}} \mathbb{E}_{(\mathbf{t}_{k,j},\mathbf{a}_{k,j})\sim\mathcal{T}_{s_k}} \left\{ L\left(f(\mathbf{t}_{k,j}, s_k; \widehat{W}, \mathbf{e}_{s_k}), \mathbf{a}_{k,j} \right) \right\}$$

where Tsk is a set of text-audio pairs for the target speaker sk. For whole model adaptation, we have the following objective:

$$\min_{W,\mathbf{e}_{s_k}} \mathbb{E}_{(\mathbf{t}_{k,j},\mathbf{a}_{k,j})\sim\mathcal{T}_{s_k}} \left\{ L\left(f(\mathbf{t}_{k,j}, s_k; W, \mathbf{e}_{s_k}), \mathbf{a}_{k,j} \right) \right\}$$

While utilizing the complete model offers increased flexibility for speaker adaptation, optimizing it becomes challenging, particularly when dealing with a limited number of cloning samples. In the optimization process, making a thoughtful decision about the number of iterations is crucial to prevent issues such as underfitting or overfitting.

4.2.2 Speaker Encoding

We suggest a speaker encoding approach that directly predicts the speaker embedding from audio samples of a previously unseen speaker. This model eliminates the need for any fine-tuning during voice cloning, allowing the same model to be applied to all unseen speakers.

In detail, the speaker encoding function, denoted as g(Ask; Θ), receives a set of cloning audio samples (Ask) and estimates the corresponding speaker embedding (esk). The function is parameterized by Θ. Ideally, the speaker encoder can be trained together with a multi-speaker generative model from the beginning, with a loss function designed to assess the quality of the generated audio.

$$\min_{W,\Theta} \mathbb{E}_{\substack{s_i\sim\mathcal{S},\\ (\mathbf{t}_{i,j},\mathbf{a}_{i,j})\sim\mathcal{T}_{s_i}}} \left\{L\left(f(\mathbf{t}_{i,j}, s_i; W, g(\mathcal{A}_{s_i};\Theta)), \mathbf{a}_{i,j}\right)\right\}$$

It's important to note that the speaker encoder undergoes training alongside the speakers for the multi-speaker generative model. Throughout the training phase, a set of cloning audio samples (Asi) is randomly chosen for each training speaker (si). In inference scenarios, the audio samples from the target speaker (sk), denoted as Ask, are used to calculate g(Ask; Θ).

However, we've encountered challenges when attempting to start training from scratch using the equation (Eq. 4). A significant issue is the difficulty in fitting an average voice to minimize the overall generative loss, often known as mode collapse in generative modeling literature. While introducing discriminative loss functions for intermediate embeddings or generated audios has been suggested to address mode collapse, in our case, these approaches only marginally enhance speaker distinctions.

In response, we propose a distinct training procedure for the speaker encoder. Speaker embeddings (besi) are derived from a trained multi-speaker generative model, represented as f(ti,j , si; W, esi). Subsequently, the speaker encoder model g(Ask; Θ) is trained to predict these embeddings from sampled cloning audios. Various objective functions can be considered for this regression problem.

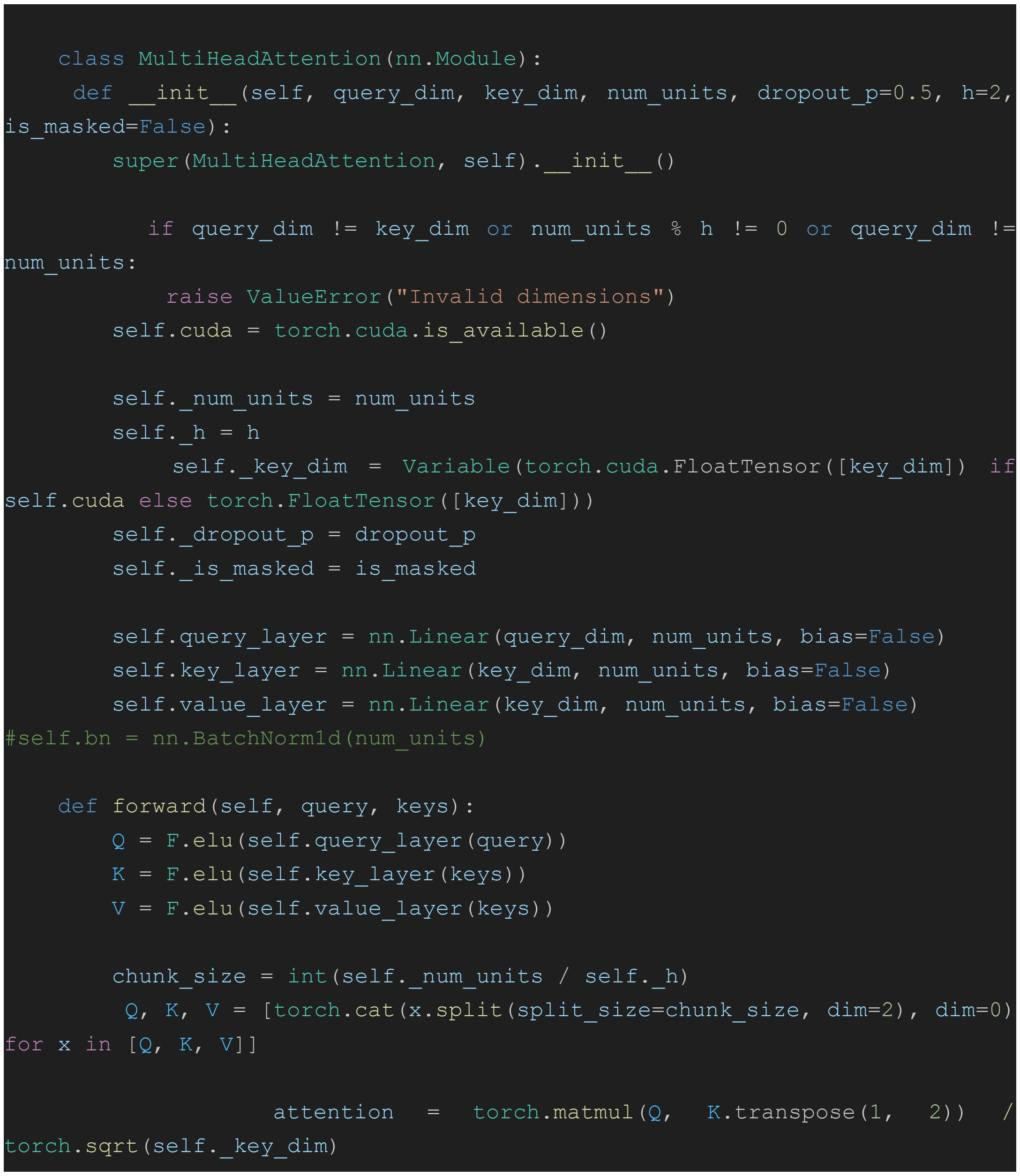
4.2.3 Discriminative models for evaluation

Assessing the performance of voice cloning often relies on human evaluations conducted through crowdsourcing platforms, but this approach can be both time-consuming and costly during the model development phase. In response, we suggest employing two evaluation methods that leverage discriminative models.

The speaker classifier plays a crucial role in identifying the speaker to whom an audio sample belongs. In the context of voice cloning evaluation, a speaker classifier is trained using the set of target speakers involved in the cloning process. Effective voice cloning is reflected in a high accuracy of speaker classification. Our speaker classifier incorporates comparable spectral and temporal processing layers along with an additional embedding layer preceding the softmax function.
Speaker verification involves verifying the claimed identity of a speaker by comparing a test audio with previously enrolled audios from the same speaker. This task is essentially a binary classification problem, determining whether the test audio and the enrolled audios belong to the same speaker .

We explore an end-to-end text-independent speaker verification model. This model can be trained on a multi-speaker dataset and subsequently tested to ascertain whether the cloned audio and the ground truth audio originate from the same speaker. Unlike the speaker classification approach, the speaker verification model doesn't necessitate training with audios from the target speaker for cloning. Consequently, it can be applied to unseen speakers with only a few samples. The equal error rate serves as a quantitative performance metric, indicating how closely the cloned audios align with the ground truth audios.

4.3 Code Snippets



```
    class MultiHeadAttention(nn.Module):
     def __init__(self, query_dim, key_dim, num_units, dropout_p=0.5, h=2,
is_masked=False):
        super(MultiHeadAttention, self).__init__()

          if query_dim != key_dim or num_units % h != 0 or query_dim !=
num_units:
            raise ValueError("Invalid dimensions")
        self.cuda = torch.cuda.is_available()

        self._num_units = num_units
        self._h = h
            self._key_dim = Variable(torch.cuda.FloatTensor([key_dim]) if
self.cuda else torch.FloatTensor([key_dim]))
        self._dropout_p = dropout_p
        self._is_masked = is_masked

        self.query_layer = nn.Linear(query_dim, num_units, bias=False)
        self.key_layer = nn.Linear(key_dim, num_units, bias=False)
        self.value_layer = nn.Linear(key_dim, num_units, bias=False)
#self.bn = nn.BatchNorm1d(num_units)

    def forward(self, query, keys):
        Q = F.elu(self.query_layer(query))
        K = F.elu(self.key_layer(keys))
        V = F.elu(self.value_layer(keys))

        chunk_size = int(self._num_units / self._h)
         Q, K, V = [torch.cat(x.split(split_size=chunk_size, dim=2), dim=0)
for x in [Q, K, V]]

                    attention = torch.matmul(Q, K.transpose(1, 2)) /
torch.sqrt(self._key_dim)
```

```
        if self._is_masked:
                                                            diag_mat     =
attention[0].sign().abs().tril().unsqueeze(0).expand(attention.size())
               mask = Variable(torch.ones(diag_mat.size()) * (-2**32 + 1),
requires_grad=False)
            if self.cuda: mask = mask.cuda()
                        attention  =  (attention  *  diag_mat)  +  (mask  *
(diag_mat-1).abs())

                    attention   =   F.dropout(F.softmax(attention,   dim=-1),
self._dropout_p)
        attention = torch.matmul(attention, V)
        restore_chunk_size = int(attention.size(0) / self._h)
                                                         attention        =
torch.cat(attention.split(split_size=restore_chunk_size,  dim=0),  dim=2)  +
query
               attention   =  F.normalize(attention.transpose(1,  2),  dim  =
1).transpose(1, 2)
        attention.contiguous()
        #attention = self.bn(attention).transpose(1, 2)

        return attention




def train(model_dv3,model_encoder,
              data_loader_dv3,
              optimizer_dv3,
              init_lr_dv3=0.002,
              checkpoint_dir_dv3=None,
              clip_thresh = 1.0,
              data_loader_encoder=None,
              optimizer_encoder=None,
              scheduler_encoder=None,
              checkpoint_interval=None,
              nepochs=None):
    # this training function is to train the combined model

    grad = {}
    def save_grad(name):
        def hook(grad):
            grads[name] = grad
        return hook
    # to remember the embeddings of the speakers

model_dv3.embed_speakers.weight.register_hook(save_grad('embeddi
ngs'))

    if use_cuda:
        model_dv3 = model_dv3.cuda()
        model_encoder = model_encoder.cuda()
    linear_dim = model_dv3.linear_dim
    r = hparams.outputs_per_step
    downsample_step = hparams.downsample_step
    current_lr = init_lr_dv3

    binary_criterion_dv3 = nn.BCELoss()

    global global_step, global_epoch
    while global_epoch < nepochs:
        running_loss = 0.0
          for step, (x, input_lengths, mel, y, positions, done,
target_lengths,
                   speaker_ids) \
                in tqdm(enumerate(data_loader_dv3)):


            model_dv3.zero_grad()
            encoder.zero_grad()

            #Declaring Requirements
            model_dv3.train()
            ismultispeaker = speaker_ids is not None
            # Learning rate schedule
            if hparams.lr_schedule is not None:
                        lr_schedule_f  =  getattr(dv3.lrschedule,
hparams.lr_schedule)
                current_lr = lr_schedule_f(
                                         init_lr,  global_step,
**hparams.lr_schedule_kwargs)
                for param_group in optimizer.param_groups:
                    param_group['lr'] = current_lr
            optimizer_dv3.zero_grad()

            # Used for Position encoding
            text_positions, frame_positions = positions
            # Downsample mel spectrogram
            if downsample_step > 1:
                mel = mel[:, 0::downsample_step, :].contiguous()

            # Lengths
            input_lengths = input_lengths.long().numpy()
            decoder_lengths = target_lengths.long().numpy() // r
// downsample_step

                                              voice_encoder    =
mel.view(mel.shape[0],1,mel.shape[1],mel.shape[2])
            # Feed data
            x, mel, y = map(Variable, [x, mel, y])
                voice_encoder, text_positions, frame_positions,
done, target_lengths, speaker_ids = map(lambda v: Variable(v) if
v  is  not  None  else  None,  [voice_encoder,  text_positions,
frame_positions, done, target_lengths, speaker_ids])
            if use_cuda:
                    x, text_positions, frame_positions, y, mel,
voice_encoder, done, target_lengths, speaker_ids = map(lambda v:
v.cuda()  if  v  is  not  None  else  None,  [x,  text_positions,
frame_positions,  y,  mel,  voice_encoder,  done,  target_lengths,
speaker_ids])

            # Create mask if we use masked loss
            if hparams.masked_loss_weight > 0:
                # decoder output domain mask
                decoder_target_mask = sequence_mask(
                    target_lengths / (r * downsample_step),
                    max_len=mel.size(1)).unsqueeze(-1)
                if downsample_step > 1:
                    # spectrogram-domain mask
                    target_mask = sequence_mask(
                                                 target_lengths,
max_len=y.size(1)).unsqueeze(-1)
                else:
                    target_mask = decoder_target_mask
                # shift mask
                decoder_target_mask = decoder_target_mask[:, r:,
:]

                target_mask = target_mask[:, r:, :]
            else:
                decoder_target_mask, target_mask = None, None
                            if global_step > 0 and global_step %
```

```
hparams.eval_interval == 0:
                          eval_model(global_step,  writer,  model,
checkpoint_dir, ismultispeaker)

            # Update
            loss_dv3.backward()
            encoder_out.backward(grads['embeddings'])

            optimizer_dv3.step()
            optimizer_encoder.step()

            # if clip_thresh> 0:
            #     grad_norm = torch.nn.utils.clip_grad_norm(
                  #             model.get_trainable_parameters(),
clip_thresh)
            global_step += 1
            running_loss += loss.data[0]

        averaged_loss = running_loss / (len(data_loader))

                       print("Loss:   {}".format(running_loss   /
(len(data_loader))))

        global_epoch += 1


    # dv3 loss function
    # backward on that
    mel_outputs.backward()
    # dv3_model.embed_speakers.weight.data = (encoder_out).data
```

4.4 Architecture Diagrams

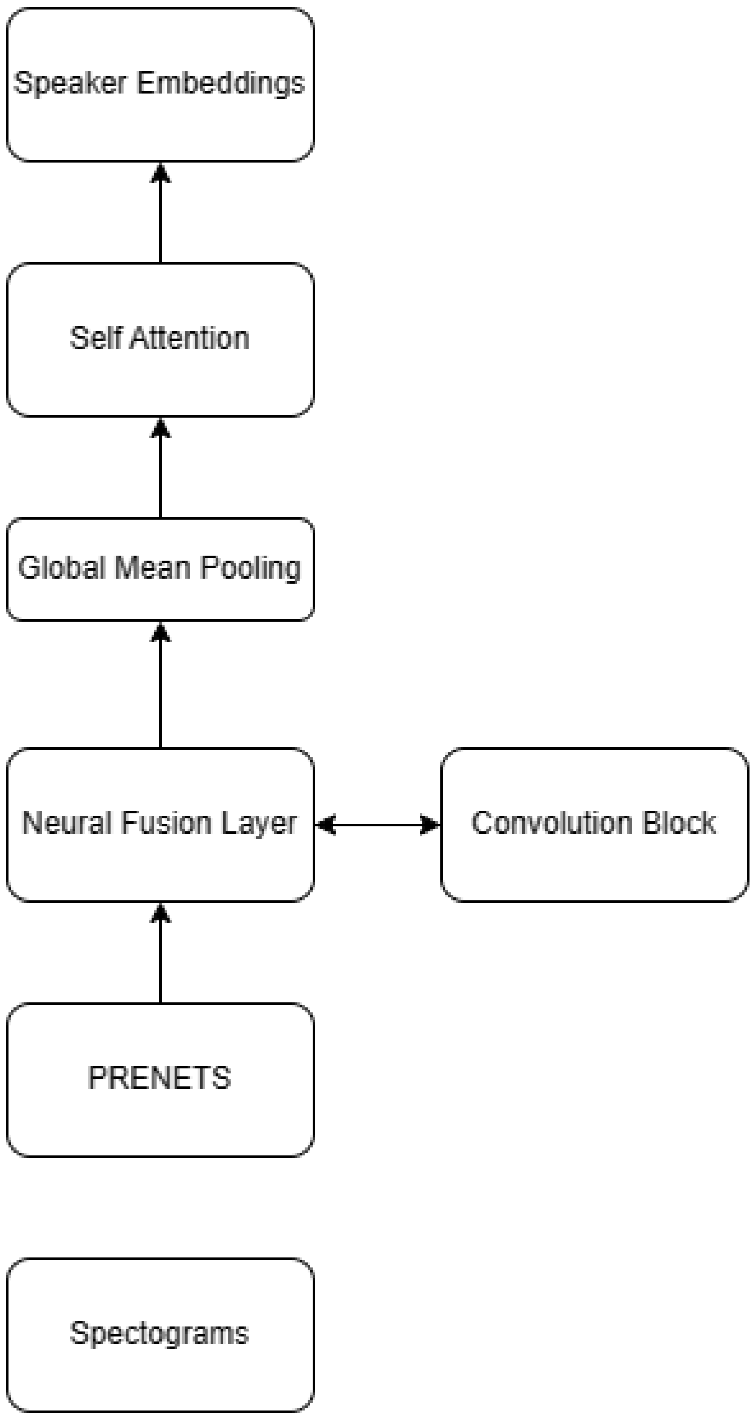


Fig 5

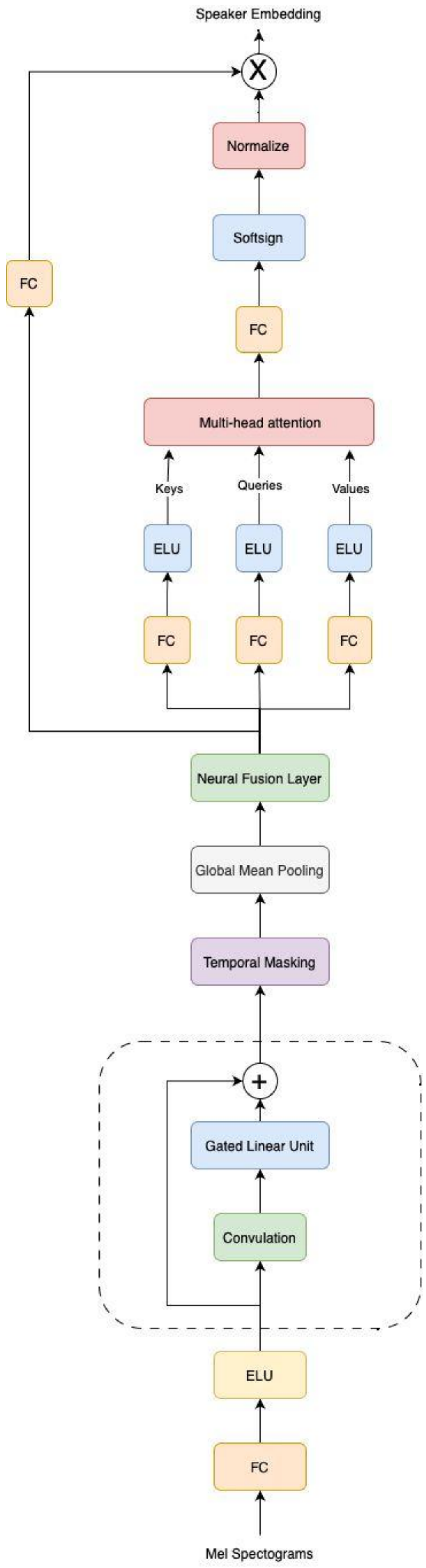

Speaker Embedding
X
Normalize
Softsign
FC
FC
Multi-head attention
Keys
Queries
Values
ELU
ELU
ELU
FC
FC
FC
Neural Fusion Layer
Global Mean Pooling
Temporal Masking
+
Gated Linear Unit
Convulation
ELU
FC
Mel Spectograms

Fig 6

Results

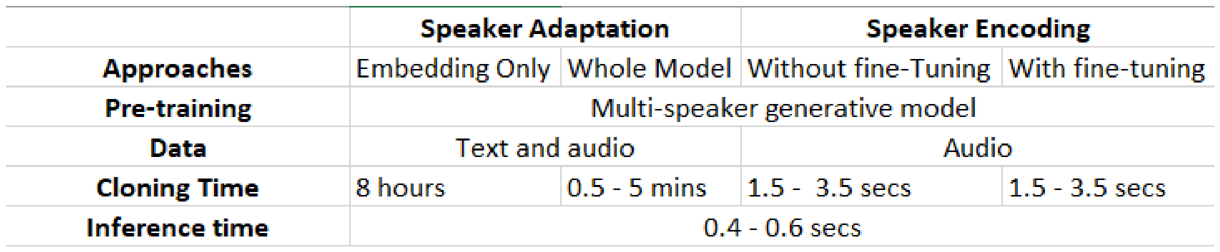
During our investigation into Neural Voice Cloning using neural fusion and attention models, significant progress in voice synthesis technology is shown. The incorporation of attention processes improves the model's overall performance by sharpening its focus. In particular, neural fusion works well in producing speech that sounds realistic and cohesive, even in the face of a shortage of training data thanks to its use of the unit concatenation approach.

The methodologies of speaker encoding and speaker adaptation provide encouraging results, showing that high-quality cloning may be accomplished even with a small number of samples. Among these, speaker encoding stands out for its capacity to estimate speaker embeddings directly, without the need for fine-tuning, demonstrating flexibility to unseen speakers.

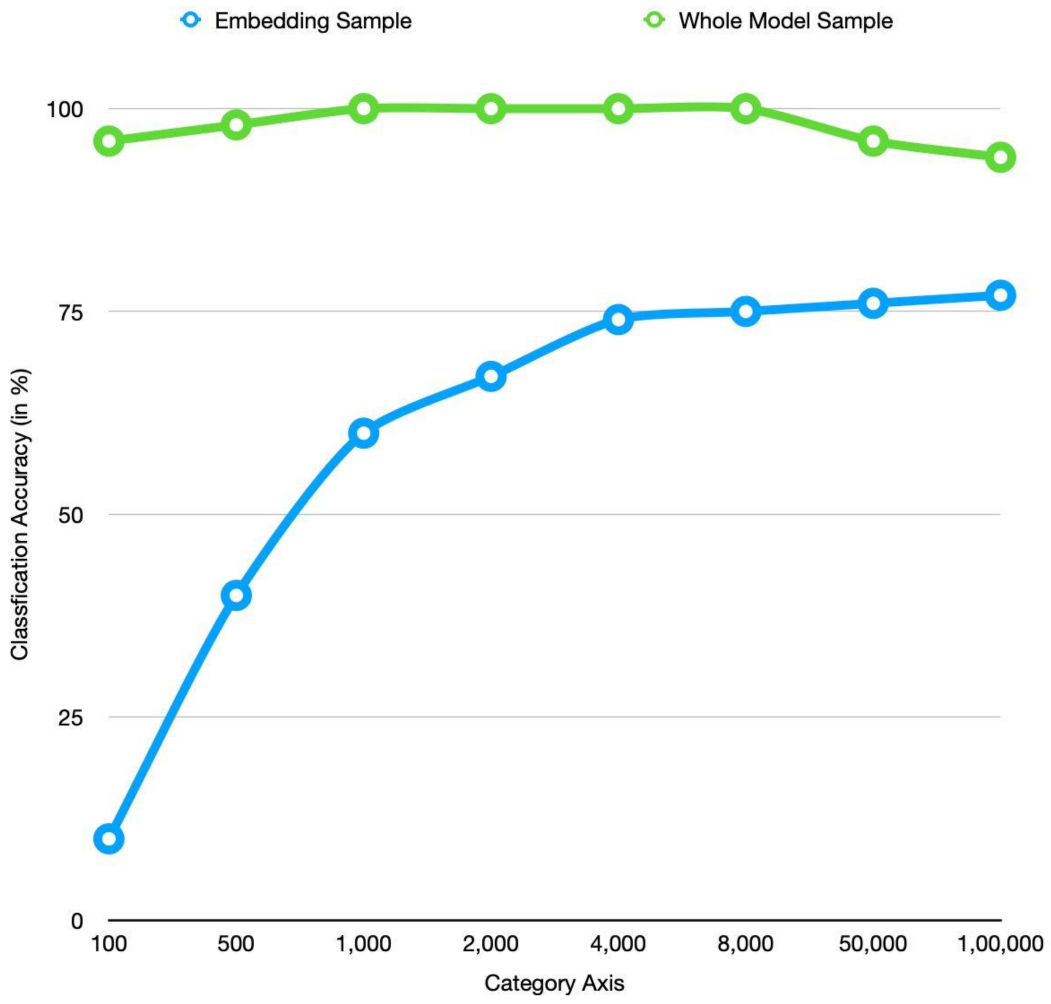
The effectiveness of the suggested techniques is validated by quantitative assessments using discriminative models, such as a speaker classifier and verification model. Trained on target speakers, the speaker classifier evaluates naturalness well, and the speaker verification model demonstrates its ability to confirm the purported identity of a speaker.

In summary, our work represents a major advancement in the areas of naturalness, similarity, and flexibility for voice synthesis, and it also emphasizes the potential of Neural Voice Cloning with neural fusion and attention models. These results open up new possibilities for the area and promise improved voice cloning technology applications and capabilities.

Neural Voice Cloning



	Speaker Adaptation		Speaker Encoding	
Approaches	Embedding Only	Whole Model	Without fine-Tuning	With fine-tuning
Pre-training	Multi-speaker generative model			
Data	Text and audio		Audio	
Cloning Time	8 hours	0.5 - 5 mins	1.5 - 3.5 secs	1.5 - 3.5 secs
Inference time	0.4 - 0.6 secs			



Conclusion

In our exploration of voice cloning with neural fusion, we've investigated two key approaches: speaker adaptation and speaker encoding. Both methods exhibit potential for high-quality cloning, even with a limited number of audios. In terms of naturalness, comparable performance was observed across speaker adaptation, speaker encoding.

Concerning speaker similarity, the study highlights the importance of a larger set of cloning audios. Notably, differences between whole-model and embedding-only adaptation emphasize the persistence of discriminative speaker information. The efficiency of compact representation through embeddings, enabling swift cloning and a small footprint, positions the speaker encoding approach favorably, especially in resource-constrained applications.

Acknowledging dataset limitations, we anticipate significant improvements with enhanced quality and increased diversity.

Looking ahead, promising avenues for voice cloning include meta-learning approaches and flexible methods for inferring model weights, showing potential for minimizing speaker information loss in diverse datasets.

In summary, our study provides insights into voice cloning through neural fusion, considering naturalness, similarity, and practical implementation. As the field progresses, we expect continued refinement, emphasizing the importance of dataset quality, diversity, and innovative learning approaches.

REFERENCES

[1] A. W. Black et al., “CMU blizzard 2007: A hybrid acoustic unit selection system from statistically predicted parameters,” in Proc. Blizzard Challenge Workshop, 2007, Paper 005.

[2] S. Tiomkin, D. Malah, S. Shechtman, and Z. Kons, “A hybrid text-tospeech system that combines concatenative and statistical synthesis units,” IEEE Trans. Audio, Speech, Lang. Process., vol. 19, no. 5, pp. 1278–1288, Jul. 2011.

[3] M. P. Aylett and J. Yamagishi, “Combining statistical parameteric speech synthesis and unit-selection for automatic voice cloning,” in Proc. LangTech, 2008.

[4] T. Merritt, R. A. Clark, Z. Wu, J. Yamagishi, and S. King, “Deep neural network-guided unit selection synthesis,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2016, pp. 5145–5149.

[5] T. Capes et al., “Siri on-device deep learning-guided unit selection textto-speech system,” in Proc. INTERSPEECH, 2017, pp. 4011–4015

[6] X. Zhou, Z.-H. Ling, and L.-R. Dai, “Extracting unit embeddings using sequence-to-sequence acoustic models for unit selection speech synthesis,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2020, pp. 7659–7663.

[7] X. Zhou, Z. Ling, and L.-R. Dai, “UnitNet: A sequence-to-sequence acoustic model for concatenative speech synthesis,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 29, pp. 2643–2655, 2021.

[8]Chen, B., Du, C. and Yu, K., 2022. Neural fusion for voice cloning. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 30, pp.1993-2001.

[9]Panayotov, V., Chen, G., Povey, D. and Khudanpur, S., 2015, April. Librispeech: an

asr corpus based on public domain audio books. In 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 5206-5210). IEEE.

[10]Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit, 2017