

SPEECH TO SPEECH SYNTHESIS FOR VOICE IMPERSONATION

Bjorn Johnson, and Jared Levy

University of California San Diego, La Jolla, CA 92093-0238

1. ABSTRACT

Numerous models have shown great success in the fields of speech recognition as well as speech synthesis, but models for speech to speech processing have not been heavily explored. We propose STSSN, a model based on current state of the art systems that fuses the two disciplines in order to perform effective speech to speech style transfer for the purpose of voice impersonation. We show that our proposed model is quite powerful, and succeeds in generating realistic audio samples despite a number of drawbacks in its capacity. We benchmark our proposed model by comparing it with a generative adversarial model which accomplishes a similar task, and show that ours produces more convincing results.

Index Terms—speech recognition, speech synthesis, GAN, mean opinion score

2. INTRODUCTION

Generating realistic voice impersonations is a field which has not received much focus in research, but we believe it has a wide variety of potential benefits and applications. A solution to this problem could be used for identity protection in interactive online settings, overcoming language barriers and speech impediments to ease communication, or increasing diversity of character voices in video games and media without the need to hire a large number of voice actors. At the same time, however, the dangers of such a technology must be considered. Convincing telemarketer scams, falsified audio recordings, and other such malicious applications exist, but we believe that enough positive use cases exist to justify the creation of such a system.

The proposed model takes as input two audio clips, one of a source speaker and a second of a target speaker, and generates a new audio clip with the same language content as the first, but with the voice and mannerisms of the second.

We benchmarked our model by additionally implementing CycleGAN, a known style transfer generative adversarial network, applied to audio spectrograms. The input is an audio clip of a source speaker (which is preprocessed) whose phrase we want to capture as well as an audio clip of the target speaker whose style we want to capture, then, the output is an audio clip of the target speaker saying the source’s phrase. The statistic most commonly used to analyze results of this

problem is the mean opinion score — an average score composed from the ratings of real humans; the ratings reflect how realistic someone believes the produced speech is.

CycleGAN mostly follows the normal generative adversarial network technique in which we define a competition between a generator network and a discriminator network, however, for CycleGAN, there are two generators and discriminators. The competition relies on the generator being able to generate results (audio spectrograms of speech in our case) as similar to the training set as possible, while the discriminator learns to be able to discriminate between the generated data and the training data. In CycleGAN, one generator generates A to B, and another is B to A, while each generator also is associated with its own discriminator. We see however, that the GANs model is not very robust to noise and the results are ok but not optimal. We attempt to get around this issue by combining multiple large pretrained networks that are used for speech recognition with large pretrained networks that synthesize speech from text. Using these state of the art deep neural networks, we can notice better results.

3. RELATED WORK

The current approaches to voice impersonation are mostly parametric systems, which heuristically tune aspects of the audio such as frequency and speed, and concatenative systems, which slice up audio samples and stitch phonemes together to generate full speech. These systems are severely lacking in their capacity to modify audio files, and generate highly artificial samples.

In terms of machine learning approaches, there have been a number of different attempts to solve this problem, but none have shown generalizable and realistic results. As far as we can tell, a model that solves this exact problem has only been implemented by a number of startups which haven’t shared details about their models or training data. There are, however, a massive number of systems which perform incredibly similar tasks, such that we were able to make a few modifications to existing networks and concatenate them to create a functional speech-to-speech synthesis network. Both Google [1] and Baidu [2] have developed text-to-speech synthesis networks which use “speaker profile” encodings to generate audio which appears to have actually been spoken by their target speaker. Other approaches exist as well, a Carnegie

Mellon group [3] approached the problem using GANs (generative adversarial networks), training one network to detect fake voices and another to produce a voice from text, forcing the generative network to synthesize natural sounding speech. A Stanford group [4] tried yet another approach, implementing neural style transfer on spectrograms in a similar fashion to popular art generation techniques, and found good results. Some even tried more direct conversion techniques, [5] used a deep neural net to convert from one speaker’s “spectral envelope” to another’s, transforming instantaneous features such as pitch.

We take inspiration for STSSN from the Tacotron2 model [6], which shows that a recurrent neural network architecture with location sensitive attention can produce incredibly realistic speech samples given only a text sequence as an input. This model still has state of the art performance years after it’s creation, but can only generate speech samples in a single voice. We also take inspiration from Baidu’s DeepSpeech model [7], which shows great success in using a recurrent architecture to produce a text transcription given an audio file as input. Our fully formed model is the concatenation of both of these models, with the insertion of a style embedding for the target speaker to the decoding stage, expanding the power of our model to be able to generate speech in multiple different voices without needing to be retrained.

4. DATASET DETAILS

For the STSSN we use a subset of the LibriSpeech dataset [8], which is a corpus of 1000 hours of audiobook readings with transcriptions. The dataset features hundreds of different speakers, both male and female with a variety of accents, making the dataset perfect for research into style transfer. Each audio clip is relatively short, up to a few tens of seconds. We used this dataset to train our speech to text model, and utilized a pretrained implementation of Tacotron2 for the decoder. We trained our network on the train-clean-100 subset of the dataset, composed of approximately 100 hours of speech, with about 25 minutes of content per speaker. We validated our model using the test-clean subset, composed of roughly 5.4 hours of speech with 8 minutes per speaker. This data is preprocessed by conversion to a mel spectrogram, using a 128 dimensional mel filterbank and a sample rate of 16kHz. These spectrograms are the actual inputs and outputs of our model.

For the CycleGAN benchmark, we implemented the plethora of male and female audio clips provided by the voice conversion challenge in 2016 titled VCC 2016 Dataset [9]. This data, split up into 1620 training samples and 108 validation samples, is preprocessed by first breaking down each spectrogram into its f0 signal, which is the fundamental frequency in speech, its spectral envelope, which is how the amplitude of the spectrogram changes, and its aperiodicity. From there we develop encodings of the spectral envelopes

and use the f0 signal to analyze pitch statistics of the audio — using logarithm gaussian normalization. The encoded spectral envelopes are then passed in as inputs to the network.

This large of a dataset provides many possible predictive tasks beyond ours such as predicting gender of a speaker, speech recognition, and speech synthesis. Our decision to implement speech synthesis with voice style transfer stems from the lack of research already done. Speech recognition and speech synthesis especially has already had a lot of work done and large companies have already produced exciting results.

5. METHODS

As discussed above, we built STSSN off of a collection of pre-existing networks. We utilize a modified DeepSpeech architecture similar to that described in [7] to generate text transcriptions of our content audio files, trained using the CTC loss function. This loss mechanism operates on a network in which the output is a 2D matrix, as shown in Figure 1, with one axis representing timesteps in the sequence and the other representing alphabet labels. The entries correspond to the probability of a specific character being uttered at each timestep. This output is then scored by comparing the output to the true label, at every possible alignment in the sequence. This treats all outputs that differ only by alignment as equivalent, and has been shown to work exceedingly well to decode a high dimensional sequence to a lower dimensional label, in our case a text transcription.

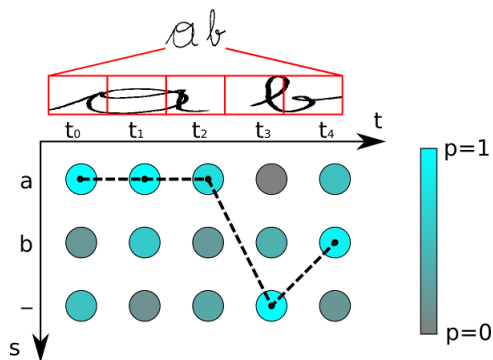


Fig. 1. Network output for use with CTC loss (dashed line represents best output label), taken from [10]

For the target speaker representation, we utilize a speaker encoding network, taken from [11], which was trained on a speaker discrimination task and shown to create high quality representations of speaker styles. This model uses a stack of LSTM layers followed by a projection to 256 dimensions, generating a fixed dimensional speaker encoding given an arbitrary length audio sequence input. It is optimized to maximize the cosine similarity of multiple embeddings of the same

speaker, while simultaneously maximizing distance between different speakers in the output space.

We feed this transcript through an implementation of the Tacotron2 network, shown graphially in Figure 2, concatenating the style encoding with the character embedding to synthesize a spectrogram representing a voice with a specific speaker style. This model operates by replacing every character in the sequence with a high dimensional learnable embedding, feeding this embedding to a stack of convolutional layers, and finally a bidirectional LSTM layer to generate an encoding for the input. At this stage, the style encoding is concatenated with the sequence encoding at every timestep, and fed into a location sensitive attention layer to generate a fixed length context vector for each decoder step. The network feeds each decoder output step through a small linear prenet, concatenates that output with the current attention context vector, feeds this into a unidirectional LSTM stack, and uses a linear projection to generate the next output step. This model is trained using text and spectrogram pairs, with a mean squared error loss function on the output spectrogram. Our motivation for this approach comes from the wealth of prior research summarized above, we believed it would be the most practical to implement, as we found open source github repositories for the encoding network as well as the Tacotron implementation, and would have a high potential for end to end success.

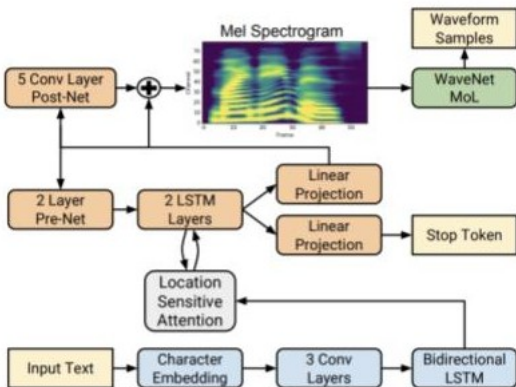


Fig. 2. Tacotron2 Architecture

Now let’s look at the main optimization objective for the CycleGAN.

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)} [\log(1 - D_Y(G(x)))]$$

The function above (from [12]) is the adversarial loss function which is part of the overall loss in which CycleGAN trains on. The G is the generator function/model that generates results as close to domain Y as possible. D is the discriminator function/model whose goal is to discriminate between data generated by G and actual examples, y. The generator model tries

to minimize this loss function while the discriminator tries to maximize it. We notice that if the discriminator is fooled by G, then $\log(1 - D(G(x)))$ will decrease and as a result the loss would decrease — which goes against the discriminators goal. From the other perspective, if the G fools D and the loss goes down, then G is successfully minimizing the loss and improving.

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)} [\|G(F(y)) - y\|_1]$$

The next part of the CycleGAN’s loss is the cycle consistency loss (also from [12]). In order to increase robustness, we want our algorithm to be able to generate an audio file and also be able to deconstruct it back into the inputs. This is why we have two generators, G and F. Generator G produces the style transferred result where F’s goal is to take in the style transferred result and produce a normal result. Thus, we notice how the formula uses the style transferred result ($G(x)$) as an input to F, and records its difference with the original non-style transferred data point. The second part of the formula is more familiar in context with our problem. We want to take a normal input and output a new style transferred result that is very similar to how the dataset is.

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F)$$

We can define our total loss as the sum of the adversarial losses of our two generators/discriminators with the cycle consistency loss. Lambda is a hyperparameter for the importance of cycle consistency.

6. EXPERIMENTS/RESULTS/DISCUSSION

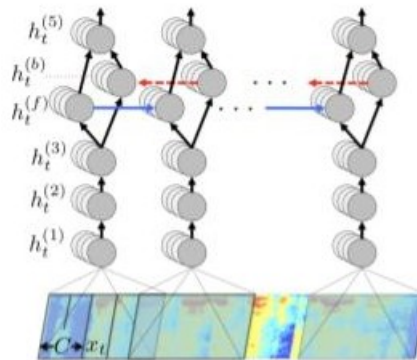


Fig. 3. DeepSpeech Architecture

We’ll describe the DeepSpeech model a bit more in depth, as this was the component most of our effort went into. Summarized graphically in Figure 3, this model consists of a single convolutional layer with 32 filters for basic feature extrac-

tion, followed by 3 residual blocks, each composed of 2 convolutional layers with 32 filters and layer normalization. This feeds into a fully connected layer which consumes the vertical dimension and all of the channels, and outputs 256 features per timestep in the sequence. The resulting sequence is run through 4 bidirectional GRU layers, each with 512 hidden units, one more fully connected layer with 512 hidden units and a GELU activation function, and finally a fully connected layer with 29 hidden units. We trained this model using the CTC Loss Function on a 29 character alphabet in conjunction with a greedy decoder. We used a batch size of 8, as this was the highest our GPU could handle without running out of memory, and trained to convergence after roughly 70 epochs of our dataset. We used a one-cycle learning rate scheduler, which linearly increased learning rate to a max of $5e-4$, and then linearly decayed. Training took roughly 48 hours on a single GPU, so it was difficult for us to experiment with many of our network’s parameters, but we did try tweaking a number of hyperparameters and comparing results. For example, research suggested using 5 GRU layers each with 1024 hidden units, but we found that cutting this down to 4 layers with only 512 units each still produced decent results. We cut down on the capacity of our model in a number of places to speed up training and reduce model size, as the fidelity of our text transcription needed not be perfect. Since we fed the transcription back into Tacotron2, minor misspellings and word errors often did not greatly impact the quality of the final result, likely because misspellings would typically have similar pronunciations to the correct word. More experimentation in this area would be valuable, and thus will be described more in the future work section.

We leave the Tacotron2 implementation unchanged, exactly to the specifications set forth in [6], using the pretrained implementation provided in [11]. This produces a mel spectrogram, which we process into an audio file using a vocoder contained in the same repository. We see some example results of our full network in Figure 4, and can visually compare the spectrograms to get an idea of how our network functions. Listening to the audio files is by far the best way to understand the success of our model, so please refer to the demo section of our presentation for that, but you can also qualitatively see results through the spectrograms of the three audio files. That the output spectrogram contains similar features to the content spectrogram (pay attention to the patterns, the horizontal stripes and vertical blanks are preserved), while having an overall appearance more similar to the style spectrogram (less haze, similar blotchiness).

In terms of hyperparameters chosen for the CycleGAN, we can see the architecture of the model in Figure 5. This architecture stems from the architecture seen in [12]. Since our problem is heavily recurrent, we use the gated linear unit (GLU) activation. This neuron activation performs well in recurrent tasks by learning when to pass information down the sequence and when not to. Furthermore, the GAN’s network

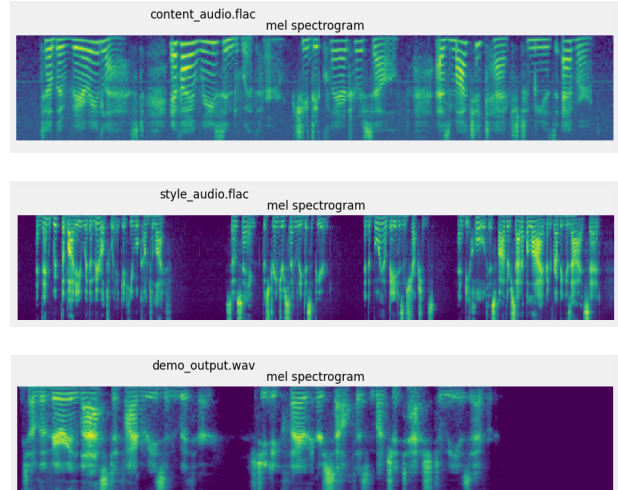


Fig. 4. Spectrograms of content (top), style (mid), and output (bottom) audio files

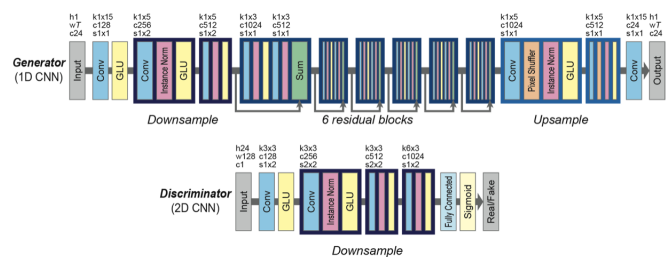


Fig. 5. CycleGAN Architecture

uses a minibatch size of 1 (1 works the best out of the multiple we tried). Although this is unusual in most deep networks, the small batch size allows our model to perform better, but takes longer in training. This hyperparameter was chosen because the dataset wasn’t very large.

Although the general adversarial loss function has the discriminator attempting to maximize, it is actually trivial, as long as the generator and discriminator have opposite goals (e.g. generator maximizes and discriminator minimizes or vice versa). We see this in our code heavily influenced from [13] in which the losses with respect to iteration is shown below.

We see that here the generators attempt to maximize loss while our discriminators attempt to minimize the CycleGAN loss explained in the Methods portion. Either way, we can see that the results are very promising.

However, after analysis of the results of both the CycleGANs and the STSSN, we can noticeably tell that although CycleGAN was good, it seemed to overfit, while the STSSN produced very interesting results and thus, we spent more timing tuning the hyperparameters for the STSSN.

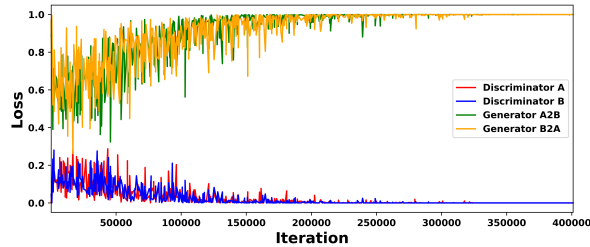


Fig. 6. GAN model losses

7. CONCLUSION/FUTURE WORK

We present a concatenation of a high quality speech to text recognition model with a state of the art text to speech synthesis model conditioned to generate speech in a variety of styles, and show that this conglomerate model has high potential in use as a speech impersonation network. We also implement a rivalling technique, CycleGAN, and compare outputs to see that the conglomerate model is capable of producing much more realistic speech.

Our results were significantly better than we had initially expected, and thus we are looking to the future to continue this work and develop an even better voice impression system. Our current approach is limited by the fact that the output of our encoding network must be decoded to raw text, then encoded again to be run through the synthesis network. We are in the process of rebuilding our model as a single, end to end model rather than a concatenation of separate models, by removing the decoder from the first network, and the encoder from the second, with the intention of generating higher quality embeddings for the spectrograms and avoiding the text bottleneck. We predict that this change will preserve significantly more information from the input audio, making it easier for the network to generate highly realistic samples. In addition, this change would reduce the size of the overall model, resulting in reduced training times and more potential for hyperparameter experimentation. This is, however, a massive undertaking and is beyond the scope of this course and research project due to time constraints. In addition to rewriting our code as a single model, we would need to establish a loss function that forces the output to resemble both the content of the first input, and the style of the second. This would likely be done through a triplet loss function, by comparing the distance of the output to the content input as well as the style input using mean squared error of the corresponding embeddings, and summing these losses together.

8. REFERENCES

[1] Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, and Yonghui Wu. Trans-

fer learning from speaker verification to multispeaker text-to-speech synthesis, 2018.

- [2] Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples, 2018.
- [3] Yang Gao, Rita Singh, and Bhiksha Raj. Voice impersonation using generative adversarial networks, 2018.
- [4] Prateek Verma and Julius O. Smith. Neural style transfer for audio spectrograms, 2018.
- [5] L. Chen, Z. Ling, L. Liu, and L. Dai. Voice conversion using deep neural networks with layer-wise generative training. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1859–1872, 2014.
- [6] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyriannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2017.
- [7] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.
- [8] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.
- [9] Ling-Hui; Saito Daisuke; Villavicencio Fernando; Wester Mirjam; Wu Zhizheng; Yamagishi Junichi Toda, Tomoki; Chen. The voice conversion challenge 2016, 2016 [dataset], 2016.
- [10] Harald Scheidl. An intuitive explanation of connectionist temporal classification, June 2018.
- [11] Coentin Jemine. Coentinj/real-time-voice-cloning, 2019.
- [12] Jun-Yan Zhu; Taesung Park; Phillip Isola; Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, November 2018.
- [13] Lei Mao. leimao/voice_converter_cyclegan, 2018.

Group 82: Speech to Speech Synthesis for Voice Impersonation

Individual Contributions

Bjorn Johnson worked on building and training the speech recognition system, and connecting this with the Tacotron2 model for end to end speech impersonation.

Jared Levy worked on implementing the CycleGAN for the same purpose, and benchmarked this with the concatenative approach.

Replies to critical reviews

Critical review from team 19

However, a part of the information of the voice may be lost since there are two huge gaps in the model. The first one is that while generating the text using the original voice, a part of the information of the original voice is lost like the little pose and the intonation of the speaker.

This is a great point, and is by far the biggest drawback in our current approach. We are looking to remedy this issue by building a single end to end model similar to the concatenation of both of our models, without the projection down to raw text. This would allow much more information to flow through the network, and hopefully preserve other features such as intonation.

What is the model proposed by Google to classify different speakers as mentioned in the project Google may use different datasets to train their model. Is the model used in this project pre-trained or not? If so, how is the performance?

How was this model proposed by Google merged into your model?

This model is a stack of 3 LSTM layers with 768 units each, and a linear projection down to 256 dimensions in between each. This was trained on a speaker discrimination task, with the output layer removed to just leave a 256 dimensional embedding of the speaker's unique characteristics. We did not merge this model into ours, we simply used it to calculate the embeddings for each of our speakers, and included these embeddings as part of our training data. It was pretrained and shown in their paper to produce embeddings suitable for this task

As mentioned in the video, the model's training process is done using PyTorch and Tensorflow.

How did you train your model across the frameworks?

How did you evaluate your model? What is the standard?

We spent a huge amount of time trying to train our model across the two different frameworks, but ultimately failed to do this. In the end we only trained the speech recognition model, and used a pretrained synthesizer. For evaluation, our speech recognition model used a combination of character error rates and word error rates. We did not have a comprehensive metric for the performance of the entire system, the standard for such things is “mean opinion score,” which is just a subjective rating of how realistic the results sound to human listeners.

Critical review from team 56

It would be interesting to go into more depth of the model, and how exactly it generates speech using a different voice from the original speech clip.

The speaker embedding is concatenated with the content encoding in the middle of the speech synthesis network, and this network was trained on text-audio pairs to generate speech in the voice of the embedded speaker.

Was there any pre-processing of the speech data prior to it being fed into the preliminary speech to text network, and then additional pre-processing of the text data fed into the text to speech network?

The speech data was preprocessed by transforming it to a mel spectrogram for the speech to text network. During training, it was also transformed by timestep and frequency masking to increase robustness of the model. The text was not preprocessed in any way before being fed into the second network, but the first part of the Tacotron2 architecture is replacement of the characters with a learned 512 dimensional embedding representation of each, followed by a convolutional layer and bidirectional lstm meant to encode the content of the sequence.

Due to the mean opinion score metric for network evaluation not being feasible for this project, as an alternate method to evaluate the accuracy of the model, could you characterize the average decibel level of the target speech voice, and then compute the decibel level of the synthesized speech and compare that to the average?

This is a possibility, but would likely fail for a number of reasons. Firstly, while average volume is a feature of speaker style, it is far from the only one, and not the most perceptible. Many other features are more significant, such as difference in frequencies and pace. Secondly, a deep neural

model could easily learn to fit the average volume of a speaker just by linearly increasing or decreasing all output to match, without learning any relevant style information.

Critical review from team 81

Why are Gated Linear Units better as an activation function for natural language processing?

How can their advantage be quantified compared to using different activation functions, such as those commonly used in deep learning like ReLU/tanh/sigmoid?

What does it mean “GLU activation focuses on the information flow of a single neuron?”

This activation function works by learning a “gate” at every neuron, meaning it decides which features get passed on and which get replaced with zeros. This can be thought of as a learnable, generalized version of a ReLU activation, and is incredibly valuable in natural language processing and recurrent neural networks by giving more freedom to the network in regards to how it will or will not pass on information.

Is there an embedding unique to each speaker? It is shown in Figure 1 on slide 10 that the input text is passed into character embeddings, but it is not stated whether the embedding is unique to a particular speaker.

Yes, the embedding is unique to the input “style” file, it is generated by passing this file through a neural network designed to create these embeddings.

As part of the motivation for the project, it is stated that “Overcoming language barriers or speech impediments” are one of the motivating factors. However, it is not further explained in the presentation whether this approach can be used to perform translation from one language to another or how a speech impediment (or its severity) might impact the performance of the approach.

By this motivation, we meant the addition or the removal of an accent, rather than actual language translation. The latter is an entirely different problem. As for speech impediments, we did not get the chance to explore the issue, but we hope that a successful model would not be greatly impacted by a mild impediment.

One of the critiques of the existing works is that “these systems require complex mathematical analysis.” It would be good if there were some mention of how the proposed approach using deep learning reduces the complexity. Is the complexity reduced at all, and if so, what does this reduction of complexity afford? For example, can this speech to speech synthesis be applied in a real-time setting or easily deployed on a mobile device?

Using deep learning greatly *increases* the complexity at training time, but once a model is fully trained complexity of cloning a voice is simple, you just provide the relevant files to the model as input and it outputs what you desire. This model could be applied in near real time, it needs to process the sequence in both directions so it cannot operate until the person is done speaking, but the forward pass is very quick, so the model could generate output in under a few seconds. With the right software backend this is theoretically possible on a mobile device.