MELLOTRON: MULTISPEAKER EXPRESSIVE VOICE SYNTHESIS BY CONDITIONING ON RHYTHM, PITCH AND GLOBAL STYLE TOKENS

Rafael Valle, Jason Li*, Ryan Prenger, Bryan Catanzaro*

NVIDIA Corporation

ABSTRACT

Mellotron is a multispeaker voice synthesis model based on Tacotron 2 GST that can make a voice emote and sing without emotive or singing training data. By explicitly conditioning on rhythm and continuous pitch contours from an audio signal or music score, Mellotron is able to generate speech in a variety of styles ranging from read speech to expressive speech, from slow drawls to rap and from monotonous voice to singing voice. Unlike other methods, we train Mellotron using only read speech data without alignments between text and audio. We evaluate our models using the LJSpeech and LibriTTS datasets. We provide F0 Frame Errors and synthesized samples that include style transfer from other speakers, singers and styles not seen during training, procedural manipulation of rhythm and pitch and choir synthesis.

Index Terms— Text-to-Speech Synthesis, Singing Voice Synthesis, Style Transfer, Deep learning

1. INTRODUCTION

Speech synthesis is typically formulated as the conversion of text to speech (TTS). This formulation, however, leaves out control for all the aspects of speech not contained in the text. Here we approach the problem of expressive speech synthesis which includes not just text, but other characteristics such as pitch, rhythm and emphasis. There are formulations to expressive speech synthesis that require animated and emotive voice data. This is an inconvenient drawback given the limited access to such data. In our approach, we can make a voice emote and sing without any such data.

Recent approaches that utilize deep learning for expressive speech synthesis combine text and a learned latent embedding for prosody or global style [1, 2]. While these approaches have shown promise, manipulating such latent variables only offers a coarse control over expressive characteristics of speech. Mellotron was motivated by the desire for fine grained control over these expressive characteristics. Notably, we show that it is easy to condition Mellotron on pitch and rhythm information automatically extracted from an audio signal or music score.

By accounting for melodic information such as pitch and rhythm, expressive speech synthesis with Mellotron can be easily extended to singing voice synthesis (SVS) [3, 4]. Unfortunately, recent attempts [4] require a singing voice dataset and heavily quantized pitch and rhythm data obtained from a digital representation of a music score, for example MIDI [5] or musicXML [6]. Mellotron does not require any singing voice in the dataset nor manually aligned pitch and text in order to synthesize singing voice.

Mellotron can make a voice emote and sing without emotion or singing data. Training Mellotron is very simple and only requires read speech and transcriptions. During inference, we can change the generated voice's speaking style, make it emote or sing by extracting pitch and rhythm characteristics from an audio file or a music score. As a bonus, with Mellotron we can explore latent characteristics from an audio corpus by sampling a dictionary of learned latent characteristics. In summary, Mellotron is a versatile voice synthesis[1] model that enables the combination of characteristics from different sources and generalizes to characteristics not seen in training data.

2. METHOD

Mellotron is a voice synthesis model that uses a combination of explicit and latent variables. Whereas well-established signal processing algorithms provide explicit variables that are valuable to expressive speech such as fundamental frequency contours and voicing decisions, deep learning strategies can be used to learn latent variables that express characteristics of an audio corpus that are unknown to the user and hard to formalize.

We factorize a single speaker mel-spectrogram M into explicit variables such as text, speaker identity, a fundamental frequency contour augmented with voiced/unvoiced decisions and two latent variables learned by the model during training. The first latent variable refers to a dictionary of vectors that can be queried with an audio input or sampled directly as described in [2]. The second latent variable is the learned attention map between the text and the mel-spectrogram as described in [7].

From now on we will refer to the augmented fundamental frequency contour as pitch contour and refer to the first

[1] Includes speech synthesis, singing voice synthesis, etc.

and second latent variables as global style tokens (GST) and rhythm respectively.

We are interested in factorizing $M = [T, S, P, R, Z]$, where T represents the text, S represents the speaker identity, P represents the pitch contour, R represents the rhythm and Z represents the global style tokens. Given this formulation, during training we maximize the following:

$$P(mel^{(i)}|T^{(i)}, S^{(i)}, P^{(i)}, R^{(i)}, Z_{mel^{(i)}}; \theta), \quad (1)$$

where the superscript i represents the i-th mel, $T^{(i)}$, $S^{(i)}$ and $P^{(i)}$ represent the text, speaker, and pitch contour associated with the i-th mel, R^i represents the learned alignments between the text and mel-spectrogram frames, $Z_{mel^{(i)}}$ represents the global style token conditioned on $mel^{(i)}$ as presented in [2], and θ represents the model parameters.

The explicit factors offers two advantages. First, by providing the model with text and speaker information, we prevent the problem of entanglement between text and speaker information. Second by providing the model with pitch contour and voicing information, we are able to directly control pitch and voicing decisions during inference.

Similarly the latent factors offers two advantages. First, by learning the alignment map between the text and mel-spectrogram during training, we do not need to extract phoneme alignments for training and can control the rhythm during inference by providing the model with an alignment map. Second, by providing the model with a dictionary of latent variables, we are able to learn latent factors that are harder to express or extract explicitly, thus leveraging the full power of latent variables.

Using this formulation we are able to transfer the text, rhythm and pitch contour from a source, e. g. audio signal or musical score, to a target speaker by replacing the variables in Equation 1 accordingly. For example, we first collect the text, pitch and rhythm (T_s, P_s, R_s), from the source, sample a GST Z_{query} from the GST dictionary learned by Mellotron, and chose a target speaker S_t.

$$P(mel_{out}|T_s, P_s, S_t, R_s, Z_{query}; \theta) \quad (2)$$

mel_{out} should now have the same text, pitch and rhythm as the source, latent characteristics obtained from the global style token and the voice of the target speaker. In our current formulation, the target speaker, S_t, would always be found in the training set, while the source text, pitch and rhythm (T_s, P_s, R_s) could be from outside the training set. This allows us to train a model that makes a voice emote and sing without using any singing voice in the training dataset, without any manual labelling of emotions nor pitch, and without any manual alignments between words and audio, nor between pitch and audio.

3. IMPLEMENTATION

In this section we are going to describe our model architecture and our training and inference setups. We plan to release our implementation and pre-trained models on github.

3.1. Architecture

Mellotron extends Tacotron 2 GST [2] with speaker embeddings and pitch countours. Unlike [8, 9], where site specific speaker embeddings are used, we use a single speaker embedding that is channel-wise concatenated with the encoder outputs over every token. The pitch contour goes through a single convolution layer followed by a ReLU non-linearity. We experiment with kernel sizes 1 and 3 and convolution dimensions 1 and 8. The pitch contour is channel-wise concatenated with the decoder inputs. We use phoneme representations whenever possible.

3.2. Training

Our implementation only requires text and audio pairs with a speaker id. Our pitch contours are automatically extracted using the Yin algorithm [10] with harmonicity thresholds between 0.1 and 0.25. Unlike [4], during training our model does not require manually aligned text, pitch and mel-spectrogram. We use the L2 loss between ground truth and predicted mels described in [2] without any modifications.

3.3. Inference

Following the description in Section 2, during inference we provide Melloron with text, rhythm and pitch information that is obtained either from an audio signal or from a musical score, a global style token and a speaker id.

3.3.1. Audio Signal

Obtaining text, rhythm and pitch information consists of three steps. First, we extract text information from an audio file by either using an automatic speech recognition model [11, 12] or by manually transcribing the text. The text information is pre-processed with our text cleaners and then converted from graphemes to phonemes.

Second, we extract rhythm information by using a forced-alignment tool [13, 14] or by using Mellotron as a forced-aligner. Alignment maps can be obtained with Mellotron by performing a teacher-forced forward pass using the data from the source signal. Whenever necessary, we fine tune the alignment maps by hand or by training Mellotron on the source signal for a few iterations with small learning rate.

The pitch data is obtained by using Yin [10] or Melodia [15]. In our quantitative experiments we use Yin to replicate the setup described in [1]. In our qualitative experiments we

use Melodia instead as we find it to be more precise than Yin, specially with regards to false voiced decisions.

3.3.2. Music Score

We operate on music scores in XML format containing event tuples with pitch, note duration and syllables for each part in the score. We directly convert pitch to frequency and use the FFT hop size to convert event durations from seconds to frames. We remind the reader that although we refer to pitch, our model's representation of pitch is continuous.

We concatenate the syllables into words and convert graphemes to phonemes. For single phone events, the duration of each phone is equal to the duration of the event. For multi-phone events, the duration of each phone is dependent on its type: we use heuristics to assign durations between 20 and 100ms to consonants and assign the remainder of the event's duration to vowels. For example, consider a one second long single note event on the word *Bass* with phoneme representation [B, AE, S]. We set B to 20 ms, S to 100 ms and the remaining duration to AE, and hence have full control over the duration of each phone.

4. EXPERIMENTS

We train our models using the LJSpeech (LJS) dataset [16], the Sally dataset, a proprietary single speaker dataset with 20 hours, and a subset of LibriTTS [17]. All datasets used in our experiments are from read speech.

We provide results that include style transfer[2] from source speakers seen and unseen in the dataset, from singers, procedural manipulation of rhythm and choir synthesis from music scores. Visit our website[3] to listen to Mellotron samples.

4.1. Training Setup

For all the experiments, we trained on LJS, Sally and the *train-clean-100* subset of LibriTTS with over 100 speakers and 25 minutes on average per speaker. Speakers with less than 5 minutes of data and files that are larger than 10 seconds were filtered out. We do not perform any data augmentation, hence any extension to a speaker's characteristics such as vocal range and speech rate is made possible with Mellotron.

We use a sampling rate of 22050 Hz and mel-spectrograms with 80 bins using librosa mel filter defaults. We apply the STFT with a FFT size of 1024, hop size of 256, and window size of 1024 samples.

We use the ADAM [18] optimizer with default parameters, start with a 1e-3 learning rate and anneal the learning rate as the loss starts to plateau. We decrease training time by using a single NVIDIA DGX-1 with 8 GPUs.

[2]Transferring text, rhythm and pitch contour to a target speaker.

[3]`https://nv-adlr.github.io/Mellotron`

For decoding the mel-spectrograms produced by Mellotron, we use a single WaveGlow [19] model trained on the Sally dataset. Our results suggest that Waveglow can be used as an universal decoder.

In our setup, we find it easier to first learn attention alignments on speakers with large amounts of data and then fine tune to speakers with less data. Thus, we first train Mellotron on LJS and Sally and finetune it with a new speaker embedding on LibriTTS, starting with a learning rate of 5e-4 and annealing the learning rate as the loss starts to plateau.

4.2. Quantitative Results

In this section we provide quantitative results that compare Gross Pitch Error (GPE) [20], Voicing Decision Error (VDE) [20] and F0 Frame Error (FFE) [21] between Mellotron and E2E-Prosody [1]. Following [1], all pitch and voicing metrics are computed using the Yin algorithm [10]. Due to the rhythm conditioning, our reference and predicted audio have the same length and does not require padding.

The results in Table 1 below show that by conditioning on pitch we can drastically reduce the error between the source and the synthesized voice. For singing voice, low pitch error is extremely important otherwise the melody might lose its identity. For prosody transfer, a lower FFE provides evidence that the style will be more precisely transferred to the target.

Model	Voice	GPE	VDE	FFE
E2E-Prosody	Single	-	-	28.1%
E2E-Prosody	Multi	-	-	27.5%
Mellotron	LJS-Sally	0.08%	9.19%	9.28%
Mellotron	LibriTTS	0.08%	8.69%	8.77%

Table 1: GPE, VDE, FFE for Mellotron and E2E-Prosody. The reference is always the same speaker.

4.3. Style transfer from Audio Signal

Mellotron is able to emote and match the style of an input audio by replicating its rhythm or both its rhythm and pitch. Overall, we note that our experiments using audio data are directly impacted by the quality of the rhythm and pitch contours provided to the model. Whereas Melodia provides rather precise pitch contours, we find that the rhythm data obtained from forced-alignments had to be constantly fine-tuned. In all audio experiments we obtain the rhythm by fine-tuning alignment maps obtained by using Mellotron as a forced-aligner. Occasionally we find that some of the pitch contours seem to be outside of a speaker's vocal range. When this happens, Mellotron defaults to a constant highest or lowest pitch value. We circumvent this by scaling the pitch contour by a constant to matches the speaker's vocal range.

4.3.1. *Rhythm Transfer*

In this experiment we transfer the rhythm and its associated text from a source audio signal to a target speaker. Our formulation provides procedural control over the duration of every phoneme, hence allowing for simple manipulations such as changing the speech rate or complex effects like speeding up or slowing down. In rhythm transfer, we provide Mellotron with an array of zeros as the pitch contour.

We show examples where we transfer the rhythm from an excerpt by Nicki Minaj to Sally. We showcase the procedural capabilities of Mellotron by processing the source rhythm with a function that produces an *accelerando* starting at half the speed and accelerating to twice the speed. For comparison, we also provide samples conditioned on the pitch contour from Nicki's track. Figure 1 shows the alignment maps.

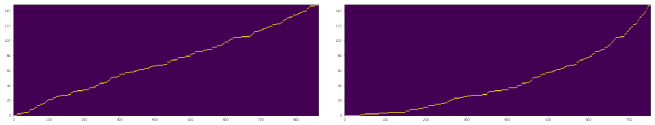

Fig. 1: Left: source alignment. Right: processed alignment.

4.3.2. *Rhythm and Pitch Transfer*

By conditioning on both rhythm and pitch, we can express characteristics of the source speaker's style. An interesting application is the creation of a hybrid with the style from a source speaker but the voice from another speaker. We show an example where we transfer the characteristics of a solemn speech to Sally. We see that Mellotron contains the same pauses and speech rate as the source which adds to the solemnity of the speech. For comparison, we provide the same phrases synthesised with the original Tacotron 2 which fails to convey the same solemnity.

4.4. Singing Voice Synthesis

Mellotron is able to generalize to rhythm and pitch from styles and speakers not in the training set. We are able to synthesize singing voice from a wide range of input speakers across a range of music styles such as rap, pop, Hindustani and western European classical music.

4.4.1. *Singing Voice from Audio Signal*

Figure 2 shows an example where we use the *Sweet Dreams* sample from the E2E-Prosody paper [1] and transfer its text, rhythm and scaled pitch to Sally. Figure 2 shows that Mellotron's pitch contour is closer to the source than E2E-Prosody is.

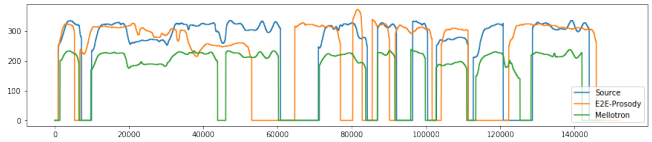

Fig. 2: Source, Mellotron and E2E-Prosody pitch contours.

4.4.2. *Style transfer from Music Score*

Unlike the experiments on audio, the rhythm and pitch contours provided to the model by a music score are correct by design. We provide a 4-part example with 20 voices per part on an excerpt of Handel's *Hallelujah*, a 8-part example with 1 voice per part on Ligeti's *Lux Aeterna* and a single voice example synthesizing the opening flute intro from Debussy's *Prlude l'aprs-midi d'un faune*. Except from cases where the pitch is beyond the speaker's vocal range, such as in Handel's sample, Mellotron has very precise pitch and rhythm.

5. CONCLUSION

In this paper we described Mellotron, a multispeaker voice synthesis model that allows for direct control of style by conditioning on rhythm and pitch obtained from an audio signal or a music score.

Our numerical results show that Mellotron is superior to other models with respect to F0 Frame Error. Our qualitative results show that Mellotron is able to generate speech in a variety of styles ranging from read speech to expressive speech, from slow drawls to rap, and from monotonous voice to singing voice although none of these styles are present in the training data.

Recent singing voice synthesis papers [4] state that "even in the case of a real recording sample recorded by listening to the original midi accompaniment, it is not easy to adjust the timing and pitch of the correct note" indicating that it is difficult for professional human singers and synthesized voice to match a source audio or source music score perfectly. Our results show that one of the advantages of Mellotron is that the rhythm and pitch contour of a synthesized sample is extremely similar to the source audio file or music score, under the assumption that the pitch is within a speaker's vocal range. When outside a speaker's vocal range, Mellotron defaults to either the lowest tone or highest tone.

For future work, we plan to study the effect of rhythm and pitch contours on the audio quality by comparing samples conditioned on pitch and rhythm data obtained from audio signals versus music scores. With respect to pitch, we are also interested in understanding the effect of multi-speaker training on a speaker's vocal range and extending a speaker's vocal range as much as possible. Last, we would like to train Mellotron on a animated and emotive storytelling style dataset to investigate the contribution of such dataset to Mellotron.

6. REFERENCES

[1] RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron J Weiss, Rob Clark, and Rif A Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron," *arXiv preprint arXiv:1803.09047*, 2018.

[2] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Fei Ren, Ye Jia, and Rif A Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," *arXiv preprint arXiv:1803.09017*, 2018.

[3] Masanari Nishimura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Singing voice synthesis based on deep neural networks," in *Interspeech 2016*, 2016, pp. 2478–2482.

[4] Juheon Lee, Hyeong-Seok Choi, Chang-Bin Jeon, Junghyun Koo, and Kyogu Lee, "Adversarially trained end-to-end korean singing voice synthesis system," *arXiv preprint arXiv:1908.01919*, 2019.

[5] Robert A Moog, "Midi: musical instrument digital interface," *Journal of the Audio Engineering Society*, vol. 34, no. 5, pp. 394–404, 1986.

[6] Michael Good, "Musicxml for notation and analysis," *The virtual score: representation, retrieval, restoration*, vol. 12, pp. 113–124, 2001.

[7] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al., "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[8] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in neural information processing systems*, 2017, pp. 2962–2970.

[9] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," *arXiv preprint arXiv:1710.07654*, 2017.

[10] Alain De Cheveigné and Hideki Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[11] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *arXiv preprint arXiv:1904.03288*, 2019.

[12] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[13] R. M. Ochshorn and M. Hawkins, "Gentle forced aligner," .

[14] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi.," in *Interspeech*, 2017, pp. 498–502.

[15] Justin Salamon and Emilia Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, 2012.

[16] Keith Ito, "The lj speech dataset," `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[17] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[18] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] Ryan Prenger, Rafael Valle, and Bryan Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.

[20] Tomohiro Nakatani, Shigeaki Amano, Toshio Irino, Kentaro Ishizuka, and Tadahisa Kondo, "A method for fundamental frequency estimation and voicing decision: Application to infant utterances recorded in real acoustical environments," *Speech Communication*, vol. 50, no. 3, pp. 203–214, Mar. 2008.

[21] Wei Chu and Abeer Alwan, "Reducing f0 frame error of f0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 3969–3972.