FastPitch Implementation

Roei Ben Zion **Tamir Sadovsky**

319090775 315316612

May 10, 2024

Abstract

In this project we'll present FastPitch, a fully-parallel text to speech (TTS) model based on FastSpeech. The model learns to predict the fundamental frequency (F_0) and condition on it for spectrogram generation, and by doing so the output speech can match the input utterance more appropriately. Also, this conditioning enables us to modify these predicted contours directly, thus the model can produce speech with enhanced expressivity, better semantic alignment, and increased listener engagement. Conditioning on F_0 improves the voice generation compared to FastSpeech while remaining a fully-parallel Transformer-based TTS model. While we have used a different Vocoder for waveform synthesis, and a different optimizer and environment for training, we show that experimental results of our trained model performance are similar to the ones presented in the paper. Subsequently, we suggest an idea for a possible improvement of the original paper, and discuss broader impact and risk of these TTS models and FastPitch in particular.

1 Introduction

Text to speech (TTS) is the task of generating a human-like voice that pronounces input text, in terms of intonation, pronunciation, content, and intelligibility level. It is a "many-to-one" task, since the same input text can be translated to many forms of pronouncing it in the same level of quality by the above criteria. The most general TTS framework usually consists of two stages - text analysis and waveform generation. In parametric methods, the two stages become transforming the input text to a spectrogram representation using an acoustic model, and transforming the resulting spectrogram to a waveform using a Vocoder model. Many modern acoustic models are autoregressive, meaning that in inference the spectrogram generation happens with conditioning on the previous predicted frames, making the inference time to increase linearly with the number of predicted frames. The FastSpeech (1) architecture solves the slow inference time by training a transformer encoder that can be decoded in parallel, making the generation orders of magnitude faster then autoregressive models while preserving quality of the generated sound. FastPitch improves the quality of synthesized speech over plain feed-forward Transformer architectures (including FastSpeech) by conditioning on F_0 estimation for every input character. Conditioning on F_0 not only improves the quality of the synthesized speech, but also improves convergence and enables voluntary modulation of the output speech (as we show later, it can be done via the pitch predictor directly). It is worth noting that the pitch is a perceptual construct, and it is been approximated by F_0 which is a mathematical construct.

2 Model Description

The models architecture is shown in figure 1. Let $X = (x_1, ..., x_n)$ the input lexical units and $Y = (y_1, ..., y_t)$ be the target mel-scaled spectrogram, where each y_i is a frame of the spectrogram ($i \in \{1...t\}$). The first FFT (Fead Forward

Transformer) generates a hidden representation h. We then take h to the duration predictor and the pitch predictor. The duration predictor predicts the number of mel-spectrogram frames that each lexical unit corresponds to, denote it's output as $\hat{d} = DurationPredictor(h)$, while the pitch predictor predicts the average pitch of every character, denote it by $\hat{p} = PitchPredictor(h)$.
Then, the pitch is projected to match the dimensionality of h and added to it to get:

$g = h + PitchEmbedding(p)$ when the pitch embedding is a 1-D convolution layer.
Then $g = [g_1, ..., g_n]$ is upsampled using the durations to get:

$$g' = [g_1, ..., g_1, ..., g_n, ..., g_n]$$

When every g_i appears d_i times in g'. The resulting g is then being passed through the decoder FFT, to get the final output $\hat{y}$. In training we use the ground truth pitch and duration p, d, while in inference we use $\hat{p}, \hat{d}$.

The model's loss function is:

$$\mathcal{L} = ||\hat{y} - y||^2 + \alpha||\hat{p} - p||^2 + \gamma|\hat{d} - d||^2$$

When α, γ are scaling factors.

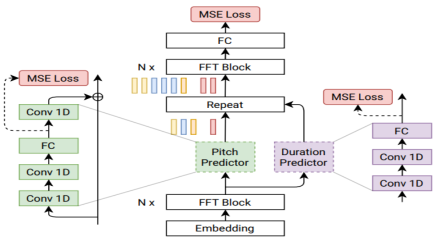


Figure 1: FastPitch architecture

2.1 Duration GT Extraction

Duration of the input characters are estimated using a pretrained Tacotron2 model, who serves as a teacher model in this case in a similar manner to FastSpeech. Let $A \in \mathcal{R}^{nXt}$ be the single attention matrix of the Tacotron2 model, the duration of the i^{th} input character is the number of output spectrogram frames that attend to it:

$$d_i = \sum_{c=1}^{t} [argmax_r A_{r,c} = i]$$

In other words, for every input symbol i, we set the duration to be the number of time steps where the attention weight is maximized in i.

2.2 Pitch GT Extraction

We note that the official implementation changed the way the GT F_0 values are estimated from the original paper (can be found in the official GitHub of FastPitch, under Release notes, Changelog, August 2021), and we refer to the method used in our trained model rather then the paper. The ground truth pitch values are computed using the probabilistic YIN (PYIN) (11) algorithm for F_0 estimation, we'll describe it shortly here. For a signal $x(t)$ the fundamental frequency (2) is defined as $F_0 = \frac{1}{T_0}$ where:

$$T_0 = min_T[x(t) = x(t + T)] \qquad (1)$$

. Hence, our goal is to find T_0.

For a certain window, we can form the Magnitude squared Difference Function (MDF) and find the values τ which minimizes it, i.e.:

$$d(\tau) = \sum_{j=1}^{W} (x(j) - x(j + \tau))^2 \qquad (2)$$

Of course, there is a large set of such values, all multiples of T_0, so we take the minimum τ value. To do this for a certain frame, YIN selects some threshold which below it the local minima points are detected, which are called candidates, and the estimation is the first τ that is a candidate. The probabilistic version does not choose constant thresholds, but defines a probability distribution over possible thresholds. Thus, the output of the P-YIN algorithm are the possible pitch values and the corresponding probabilities. To find the dynamics of the pitch evolution, PYIN uses a HMM with the probabilities obtained above as the emission probabilities. Thus, the task boils down to finding the most likely pitch sequence given the emissions (AKA decoding). As seen in class, this is done by the Viterbi algorithm for the Maximum Likelihood Sequence Estimation.

3 Experiments

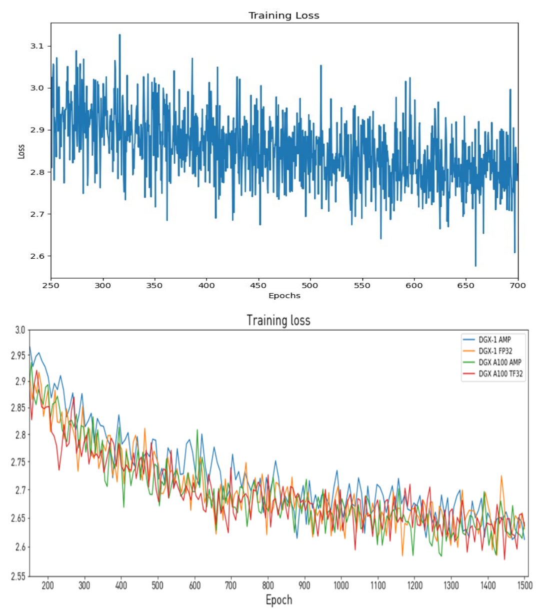


Figure 2: Comparison of Training Losses: Our Training Loss (up) vs. Authors' Training Loss (down)

3.1 Training

The model is trained on the LJSpeech 1.1 dataset (5) which contains approximately 24 hours of single-speaker speech recorded at 22 050 Hz. We found that Setting the NGC docker containers on the university cluster produced many techinical difficulties, so to match our running environment more easily we do not use Apex optimizers as in the original source code (specifically Lamb optimizer), but use the Pytorch implementation of the fused Adam optimizer with learning rate 0.1, weight decay $1e^{-6}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e^{-9}$. Learning rate is increased during 1000 warmup steps, and then decayed according to the Transformer schedule (8). We use batch size of 16, and scaling factors of $\alpha = \gamma = 0.1$ (the pitch and duration predictors loss scaling factors). We train the model on the university Slurm cluster for 680 epochs, taking a total of around 3 days. We do not change the model architecture at all, as the one from the official implementation worked best. The training targets were obtained by converting the audio waveforms from the LJSpeech-1.1 dataset into mel-spectrograms using the short-time Fourier transform (9). The training loss represents the average loss over an entire training epoch. We observed a similarity in the loss trends, as shown in Figure 2. We do not train a Vocoder model ourselves, but use a pretrained HIFI-GAN Vocoder model from Torchaudio (10) for inference. While the authors do not note the settings of their WaveGlow (4) Vocoder, we make sure our Vocoders settings matches the default settings presented in the FastPitch GitHub.

3.2 Evaluation

We evaluate our model against a baseline of taking the ground truth waveforms, computing their mel-spectrogram and recosntracting using the same Vocoder model we used for our trained model. We have evaluated our trained FastPitch model using Mean Opinion Scores (MOS), asking 3 people on 50 examples the model and the baseline, while not reviling to them which output they hear (weather its our trained model or the baseline) and asked them to prioritize grading based on naturalness of the audio, placing less emphasis on disturbances. The 50 phrases we used are taken from the LJSpeech-1.1 test set, the exact phrases are in the project GitHub repository.

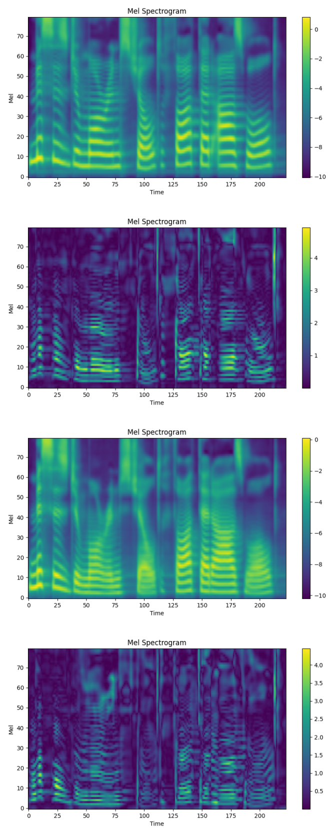
Model	**MOS**
FastPitch (Synthesized Mel-Spectrogram)	3.804 ± 0.905
Ground Truth Mel-Spectrogram	3.878 ± 0.999

Table 1: Mean Opinion Scores confidence intervals comparing synthesized mel-spectrograms to ground truth mel-spectrograms.

3.3 Pitch Conditioning and Inference Performance

A predicted pitch contour can be modified during inference to control certain perceived qualities of the generated speech. It can be used to increase or decrease F_0, raise expressiveness and variance of pitch. The audio samples accompanying this paper demonstrate the effects of increasing, decreasing or inverting the frequency around the mean value for a single utterance. Similarly to the authors, we conducted experiments on modifying the predicted pitch contour during inference to control certain perceived qualities of the generated speech.Figure 3 shows an example of shifting the frequency by 50 Hz. Compared to simple shifting in the frequency domain, FastPitch preserves the perceived identity of the speaker, and models the action of vocal chords that happens during voluntary modulation of voice.

Figure 3: Shifting F_0 with FastPitch by adding a constant to the predicted pitch $\hat{p}$ during inference. Top pair: +50 Hz shift and its difference from the original. Bottom pair: -50 Hz shift and its difference.



4 Broader Impact and Risks

FastPitch is a powerful TTS model and it represents a significant advancement in TTS technology. Two key enhancements highlighted in the paper are parallel inference and pitch conditioning, each bringing distinct benefits and implications. Firstly, the implementation of parallel inference greatly enhances the efficiency of real-time TTS systems. This improvement is critical for applications requiring rapid response times, such as virtual assistants, navigation systems, or interactive educational tools. Real-time responsiveness is essential for creating seamless and engaging user experiences.
Secondly, the incorporation of pitch conditioning enables FastPitch to produce a more diverse range of outputs, enriching the expressiveness and naturalness of synthesized speech. By conditioning the model on pitch, it becomes possible to capture subtle nuances in intonation and emotional inflection, facilitating more engaging and human-like interactions. However, this enhancement introduces risks related to bias amplification or unintended reinforcement of certain speech patterns. For example, training on data with a dominant amount of high pitch voices examples might cause the model to predict higher pitches for all input texts, regardless of the intended pitch of the speaker in the text.
Other ethical concern involves the misuse of synthesized voices for deceptive purposes, such as creating fake audio content that could mislead or deceive individuals. Additionally, there are implications related to privacy and consent, particularly if synthesized voices are generated without the knowledge or permission of individuals whose voices are used for training datasets. Ensuring transparency and ethical use of such technology will be essential in mitigating these risks and promoting responsible deployment of voice synthesis models like FastPitch.

5 Method Improvement Proposal

In search of ways to improve the model, we note that FastSpeech 2 (3), which is a more recognized architecture that is also based on FastSpeech, uses a different way of extracting the GT durations. Instead of using a pretrained Tacotron 2 as a teacher model for extracting the GT durations, as done is FastPitch and FastSpeech, FastSpeech 2 uses the Montreal forced alignment (MFA) (7). The authors of FastPitch show robustness for to the quality of alignments only within distinct Tacotron 2 models, by testing multiple Tacotron 2 models, but not for other duration estimation methods. However, in FastSpeech 2 they show that estimating the durations with MFA achieves superior results over a teacher model estimation in their model (under subsection 3.2.2, "More Accurate Duration for Model Training"). Thus, changing the estimation to MFA might be beneficial for output quality. Of course, FastSpeech 2 is a different model than FastPitch, so the improvement shown in FastSpeech 2 might not show the same results here. The authors of FastPitch have shown that there is no improvement in quality using the MFA while using phonemes (instead of characters) as inputs, but never show what happens in simply changing the duration estimation to MFA, in the same way done in FastSpeech 2.

6 Conclusions

In conclusion, our project has provided a comprehensive overview and successful implementation of FastPitch. Through our efforts, we have not only gained valuable insights into its functionalities but also demonstrated its practical application. Moving forward, the knowledge and experience gained from this project will undoubtedly contribute to our ongoing endeavors in the exciting field of audio processing using deep learning.

References

[1] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in NeurIPS 2019, 2019.

[2] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," arXiv:1609.03499, 2016.

[3] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," arXiv:2006.0455, 2020.

[4] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in ICASSP, 2019, pp. 3617–3621.

[5] K. Ito, "The LJ Speech Dataset," `https://keithito.com/LJ-Speech-Dataset`, 2017.

[6] Z. Zeng, J. Wang, N. Cheng, T. Xia, and J. Xiao, "AlignTTS: Efficient Feed-Forward Text-to-Speech System without Explicit Alignment," arXiv:2003.01950, 2020.

[7] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi," in Proc. Interspeech 2017, 2017, pp. 498–502.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, pp. 5998–6008. 2017.

[9] F.J. Owens, M.S. Murphy, A short-time Fourier transform, Signal Processing, Volume 14, Issue 1, 1988, Pages 3-10, ISSN 0165-1684, 1988.

[10] HIFIGAN-VOCODER-BUNDLE documentation: https://pytorch.org/audio/main/generated/torchaudio.prototype.pipelines.HiFiGANVocoderBundle.html

[11] Mauch, M., Dixon, S. (2014). *PYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions*. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 659-663). IEEE.