ming024 / **FastSpeech2**

An implementation of Microsoft's "FastSpeech 2: Fast and High-Quality End-to-End Text to Speech"

MIT License

545 stars 165 forks

Star Notifications

Code Issues 33 Pull requests 3 Actions Projects Wiki Security Insights

master Go to file

ming024 ... on 8 Jul

View code

README.md

FastSpeech 2 - PyTorch Implementation

This is a PyTorch implementation of Microsoft's text-to-speech system **FastSpeech 2: Fast and High-Quality End-to-End Text to Speech**. This project is based on xcmyz's implementation of FastSpeech. Feel free to use/modify the code.

There are several versions of FastSpeech 2. This implementation is more similar to version 1, which uses F0 values as the pitch features. On the other hand, pitch spectrograms extracted by continuous wavelet transform are used as the pitch features in the later versions.

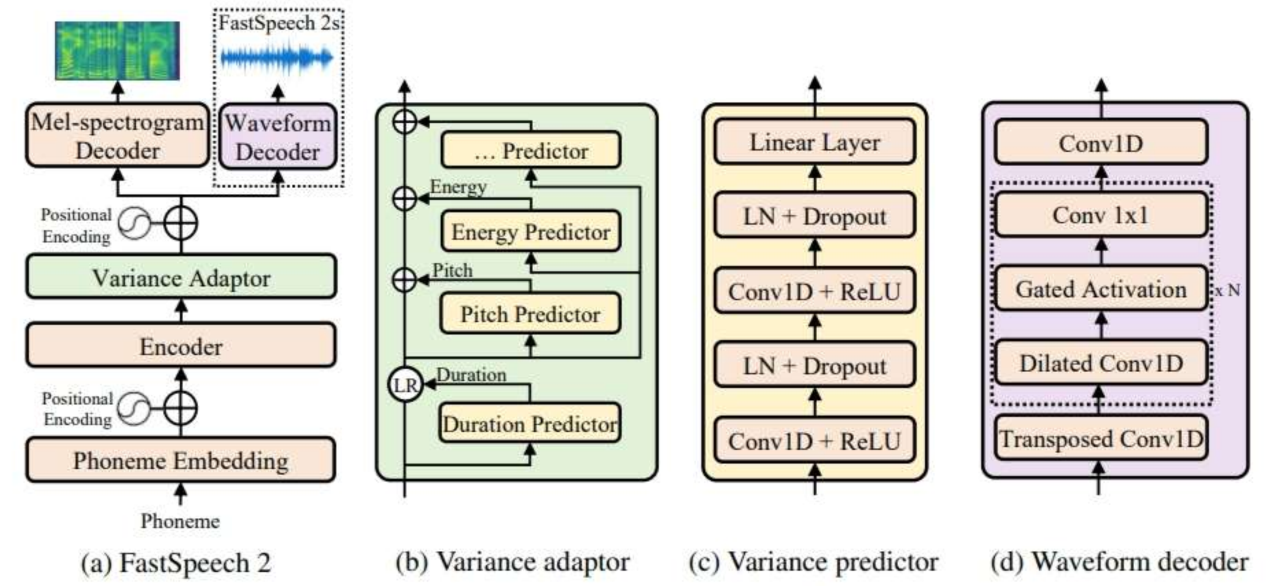


Figure 1: The overall architecture for FastSpeech 2 and 2s. LR in subfigure (b) denotes the length regulator operation proposed in FastSpeech. LN in subfigure (c) denotes layer normalization. Variance predictor represents duration/pitch/energy predictor.

Updates

- 2021/7/8: Release the checkpoint and audio samples of a multi-speaker English TTS model trained on LibriTTS
- 2021/2/26: Support English and Mandarin TTS
- 2021/2/26: Support multi-speaker TTS (AISHELL-3 and LibriTTS)
- 2021/2/26: Support MelGAN and HiFi-GAN vocoder

Audio Samples

Audio samples generated by this implementation can be found here.

Quickstart

Dependencies

You can install the Python dependencies with

```
pip3 install -r requirements.txt
```

Inference

You have to download the pretrained models and put them in `output/ckpt/LJSpeech/`, `output/ckpt/AISHELL3`, or `output/ckpt/LibriTTS/`.

For English single-speaker TTS, run

```
python3 synthesize.py --text "YOUR_DESIRED_TEXT" --restore_step 900000 --mode
single -p config/LJSpeech/preprocess.yaml -m config/LJSpeech/model.yaml -t
config/LJSpeech/train.yaml
```

For Mandarin multi-speaker TTS, try

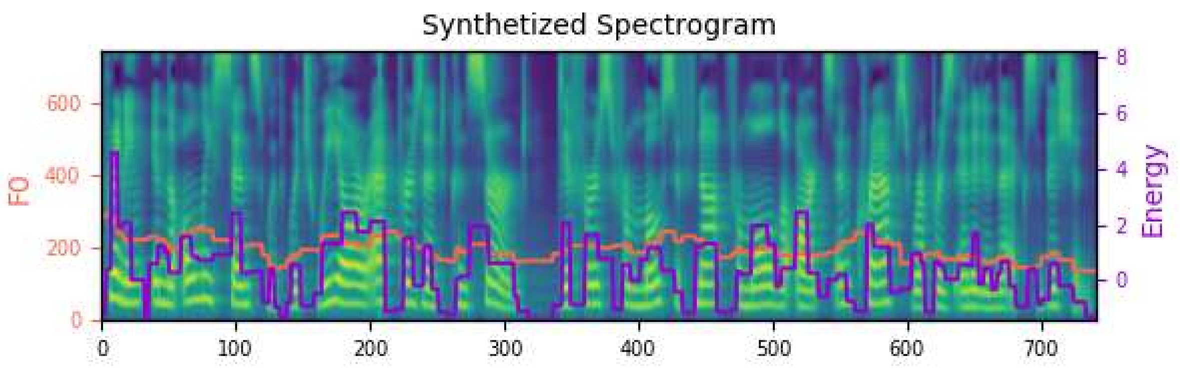
```
python3 synthesize.py --text "大家好" --speaker_id SPEAKER_ID --restore_step 600000
--mode single -p config/AISHELL3/preprocess.yaml -m config/AISHELL3/model.yaml -t
config/AISHELL3/train.yaml
```

For English multi-speaker TTS, run

```
python3 synthesize.py --text "YOUR_DESIRED_TEXT"  --speaker_id SPEAKER_ID --
restore_step 800000 --mode single -p config/LibriTTS/preprocess.yaml -m
config/LibriTTS/model.yaml -t config/LibriTTS/train.yaml
```

The generated utterances will be put in `output/result/`.

Here is an example of synthesized mel-spectrogram of the sentence "Printing, in the only sense with which we are at present concerned, differs from most if not from all the arts and crafts represented in the Exhibition", with the English single-speaker TTS model.



Batch Inference

Batch inference is also supported, try

```
python3 synthesize.py --source preprocessed_data/LJSpeech/val.txt --restore_step
900000 --mode batch -p config/LJSpeech/preprocess.yaml -m
config/LJSpeech/model.yaml -t config/LJSpeech/train.yaml
```

to synthesize all utterances in `preprocessed_data/LJSpeech/val.txt`

Controllability

The pitch/volume/speaking rate of the synthesized utterances can be controlled by specifying the desired pitch/energy/duration ratios. For example, one can increase the speaking rate by 20 % and decrease the volume by 20 % by

```
python3 synthesize.py --text "YOUR_DESIRED_TEXT" --restore_step 900000 --mode
single -p config/LJSpeech/preprocess.yaml -m config/LJSpeech/model.yaml -t
config/LJSpeech/train.yaml --duration_control 0.8 --energy_control 0.8
```

Training

Datasets

The supported datasets are

- LJSpeech: a single-speaker English dataset consists of 13100 short audio clips of a female speaker reading passages from 7 non-fiction books, approximately 24 hours in total.
- AISHELL-3: a Mandarin TTS dataset with 218 male and female speakers, roughly 85 hours in total.
- LibriTTS: a multi-speaker English dataset containing 585 hours of speech by 2456 speakers.

We take LJSpeech as an example hereafter.

Preprocessing

First, run

```
python3 prepare_align.py config/LJSpeech/preprocess.yaml
```

for some preparations.

As described in the paper, Montreal Forced Aligner (MFA) is used to obtain the alignments between the utterances and the phoneme sequences. Alignments of the supported datasets are provided here. You have to unzip the files in `preprocessed_data/LJSpeech/TextGrid/` .

After that, run the preprocessing script by

```
python3 preprocess.py config/LJSpeech/preprocess.yaml
```

Alternately, you can align the corpus by yourself. Download the official MFA package and run

```
./montreal-forced-aligner/bin/mfa_align raw_data/LJSpeech/ lexicon/librispeech-lexicon.txt english preprocessed_data/LJSpeech
```

or

```
./montreal-forced-aligner/bin/mfa_train_and_align raw_data/LJSpeech/ lexicon/librispeech-lexicon.txt preprocessed_data/LJSpeech
```

to align the corpus and then run the preprocessing script.

```
python3 preprocess.py config/LJSpeech/preprocess.yaml
```

Training

Train your model with

```
python3 train.py -p config/LJSpeech/preprocess.yaml -m config/LJSpeech/model.yaml -t config/LJSpeech/train.yaml
```

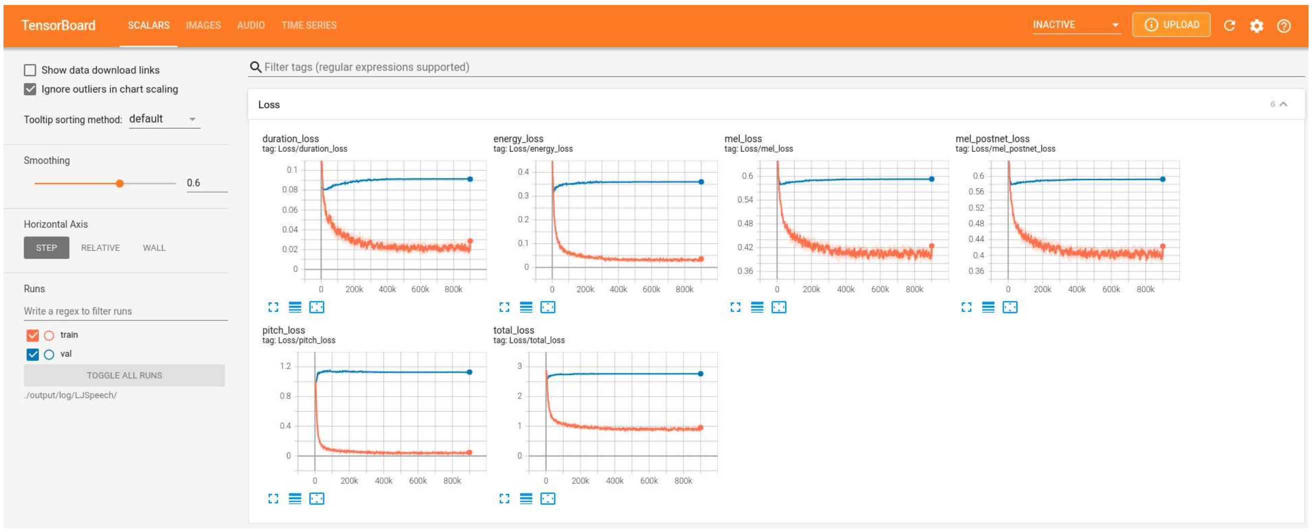
The model takes less than 10k steps (less than 1 hour on my GTX1080Ti GPU) of training to generate audio samples with acceptable quality, which is much more efficient than the autoregressive models such as Tacotron2.

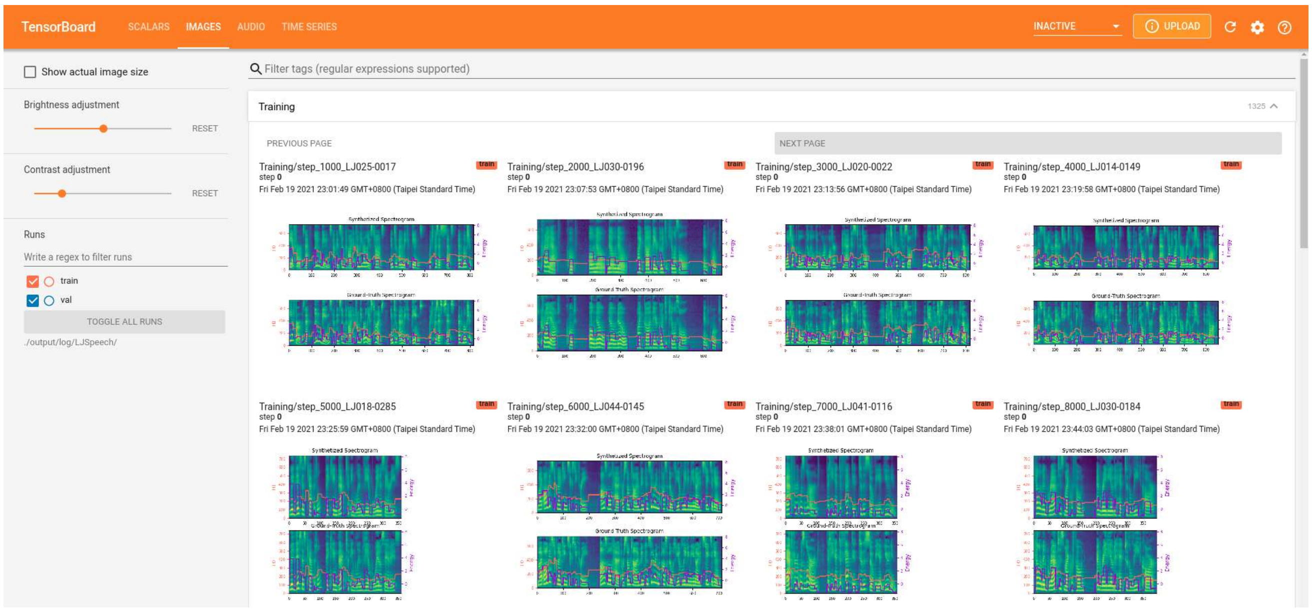
TensorBoard

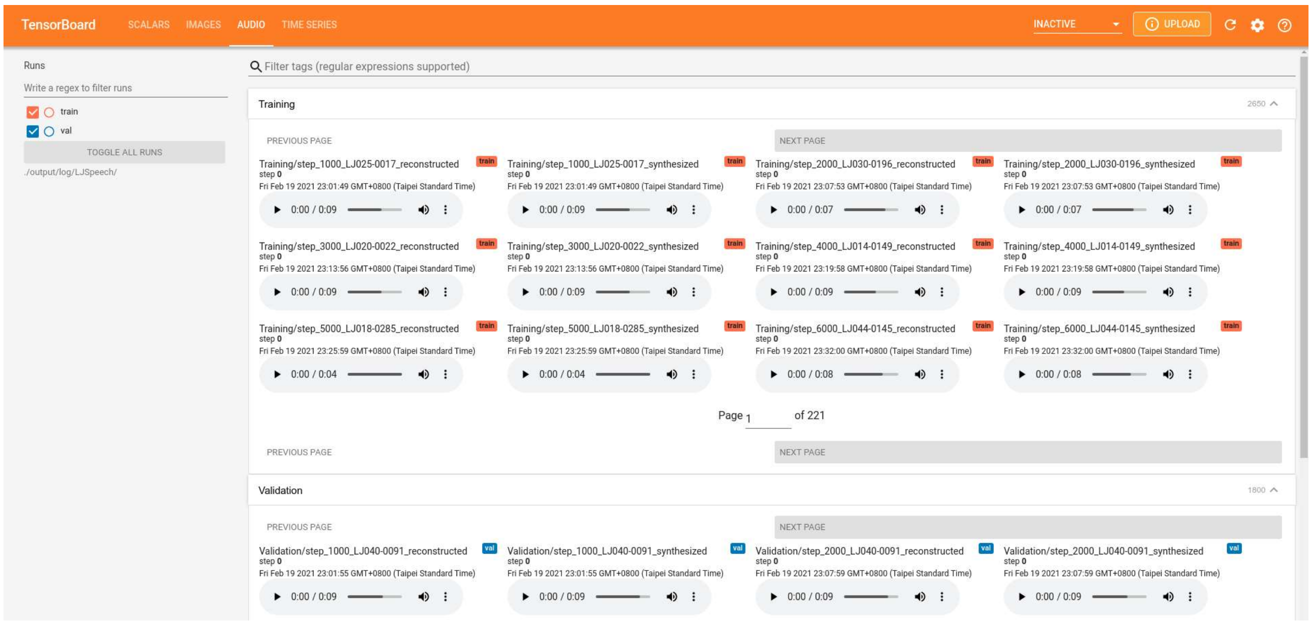
Use

```
tensorboard --logdir output/log/LJSpeech
```

to serve TensorBoard on your localhost. The loss curves, synthesized mel-spectrograms, and audios are shown.




TensorBoard
SCALARS
IMAGES
AUDIO
TIME SERIES
INACTIVE
UPLOAD
Show actual image size
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter runs
train
val
TOGGLE ALL RUNS
./output/log/LJSpeech/
Filter tags (regular expressions supported)
Training
1325
PREVIOUS PAGE
NEXT PAGE
Training/step_1000_LJ025-0017
Training/step_2000_LJ030-0196
Training/step_3000_LJ020-0022
Training/step_4000_LJ014-0149
Training/step_5000_LJ018-0285
Training/step_6000_LJ044-0145
Training/step_7000_LJ041-0116
Training/step_8000_LJ030-0184
Synthetized Spectrogram
Ground-Truth Spectrogram


TensorBoard
SCALARS
IMAGES
AUDIO
TIME SERIES
INACTIVE
UPLOAD
Runs
Write a regex to filter runs
train
val
TOGGLE ALL RUNS
./output/log/LJSpeech/
Filter tags (regular expressions supported)
Training
2650
PREVIOUS PAGE
NEXT PAGE
Training/step_1000_LJ025-0017_reconstructed
Training/step_1000_LJ025-0017_synthesized
Training/step_2000_LJ030-0196_reconstructed
Training/step_2000_LJ030-0196_synthesized
Training/step_3000_LJ020-0022_reconstructed
Training/step_3000_LJ020-0022_synthesized
Training/step_4000_LJ014-0149_reconstructed
Training/step_4000_LJ014-0149_synthesized
Training/step_5000_LJ018-0285_reconstructed
Training/step_5000_LJ018-0285_synthesized
Training/step_6000_LJ044-0145_reconstructed
Training/step_6000_LJ044-0145_synthesized
Page 1 of 221
Validation
1800
Validation/step_1000_LJ040-0091_reconstructed
Validation/step_1000_LJ040-0091_synthesized
Validation/step_2000_LJ040-0091_reconstructed
Validation/step_2000_LJ040-0091_synthesized

Implementation Issues

- Following xcmyz's implementation, I use an additional Tacotron-2-styled Post-Net after the decoder, which is not used in the original FastSpeech 2.

- Gradient clipping is used in the training.
- In my experience, using phoneme-level pitch and energy prediction instead of frame-level prediction results in much better prosody, and normalizing the pitch and energy features also helps. Please refer to `config/README.md` for more details.

Please inform me if you find any mistakes in this repo, or any useful tips to train the FastSpeech 2 model.

References

- FastSpeech 2: Fast and High-Quality End-to-End Text to Speech, Y. Ren, *et al*.
- xcmyz's FastSpeech implementation
- TensorSpeech's FastSpeech 2 implementation
- rishikksh20's FastSpeech 2 implementation

Citation

```
@INPROCEEDINGS{chien2021investigating,
  author={Chien, Chung-Ming and Lin, Jheng-Hao and Huang, Chien-yu and Hsu, Po-
chun and Lee, Hung-yi},
  booktitle={ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech
and Signal Processing (ICASSP)},
  title={Investigating on Incorporating Pretrained and Learnable Speaker
Representations for Multi-Speaker Multi-Style Text-to-Speech},
  year={2021},
  volume={},
  number={},
  pages={8588-8592},
  doi={10.1109/ICASSP39728.2021.9413880}}
```

Releases

No releases published

Packages

No packages published

Contributors 2

ming024 Chung-Ming Chien

cyhuang-tw Chien-yu Huang

Languages

● **Python** 81.3% ● **HTML** 18.7%