ML Training BenchMark

Mahesh Pandey

Framework	Models

Text to Speech **PyTorch** **[FastPitch]**

I Follow this Link :-
https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/FastPitch

ML Commons Github Link
https://github.com/mlcommons/training/tree/master/language_model/tensorflow/bert

https://mlcommons.org/benchmarks/training/

First to Understand the Article & All of things

FastPitch 1.1 for PyTorch

Setup

The following section lists the requirements that you need to meet in order to start training the FastPitch model.

Requirements

This repository contains Dockerfile that extends the PyTorch NGC container and encapsulates some dependencies. Aside from these dependencies, ensure you have the following components:

- NVIDIA Docker
- PyTorch 22.08-py3 NGC container or newer
- supported GPUs:
  - NVIDIA Volta architecture
  - NVIDIA Turing architecture
  - NVIDIA Ampere architecture

Step 1:- You Have to Pull Requirements Docker & Container From NVIDIA NCG Container

```
root@mpcl-master: ~
root@mpcl-master:~# docker images
REPOSITORY                 TAG             IMAGE ID       CREATED        SIZE
nvcr.io/nvidia/tensorflow  24.07-tf2-py3   4574f4bf6f57   6 weeks ago    15.1GB
nvcr.io/nvidia/pytorch     22.08-py3       b3d16c039217   2 years ago    14.6GB
nvcr.io/nvidia/tensorflow  20.06-tf2-py3   4ebde669c238   4 years ago    9.45GB
root@mpcl-master:~#
```

Then i am run this Commands to run the Container
Docker run –gpus all -it -v /root/ML/:/workspace/ML -p 1002:8889 <images ID >

```
root@mpcl-master:~# d^Cker run --gpus all -it -v /root/mahesh/Examples:/workspace/mahesh -p 1000:8080  4574f4bf6f57
root@mpcl-master:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS                  PORTS                                                       NAMES
e5acd662193f   b3d16c039217   "/opt/nvidia/nvidia_…"   About an hour ago    Up About an hour        6006/tcp, 8888/tcp, 0.0.0.0:1002->8889/tcp, :::1002->8889/tcp   mahes
h
fb545d1ab8b5   4574f4bf6f57   "/opt/nvidia/nvidia_…"   3 weeks ago          Exited (0) 9 days ago                                                               modes
t_gates
root@mpcl-master:~#
```

Then i got a Container id , i went inside Container to Clone a git repository inside a Container .

Step 1.1:-

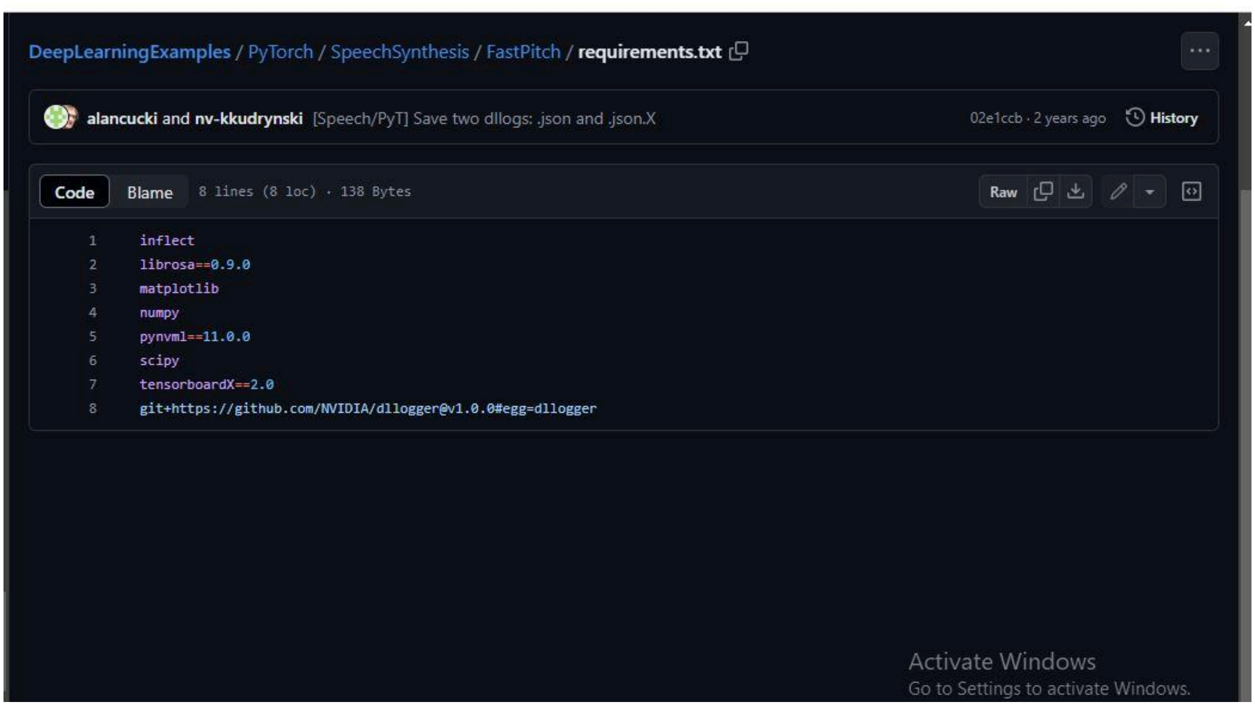
Clone the repository.
git clone https://github.com/NVIDIA/DeepLearningExamples.git
cd DeepLearningExamples/PyTorch/SpeechSynthesis/FastPitch

Step 1.2:-

Build and run the FastPitch PyTorch NGC container.
By default, the container will use all available GPUs.

```
bash scripts/docker/build.sh
bash scripts/docker/interactive.sh
```

Step 2:- you have to Run the Requrirements.txt in the inside Container
Pip install -r requirements.txt

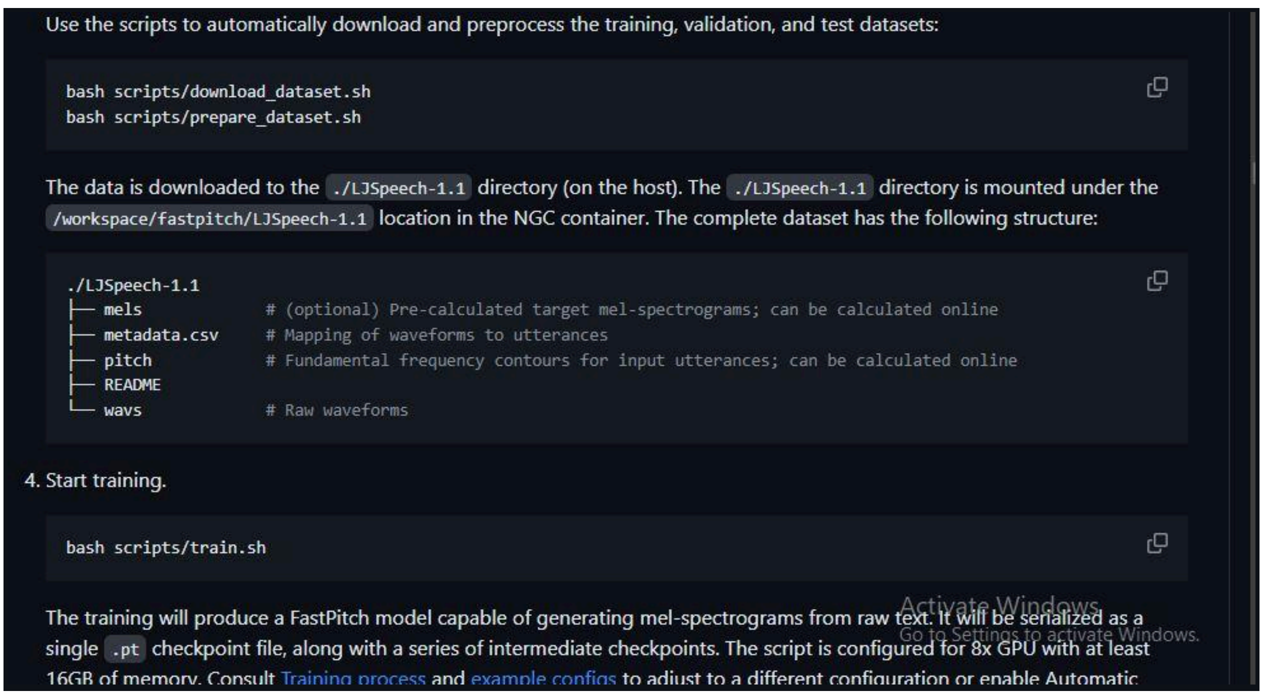


Step 3 :-

Download and preprocess the dataset.
Use the scripts to automatically download and preprocess the training, validation, and test datasets:

bash scripts/download_dataset.sh
bash scripts/prepare_dataset.sh

The data is downloaded to the `./LJSpeech-1.1` directory (on the host). The `./LJSpeech-1.1` directory is mounted under the `/workspace/fastpitch/LJSpeech-1.1` location in the NGC container. The complete dataset has the following structure:


Use the scripts to automatically download and preprocess the training, validation, and test datasets:

```
bash scripts/download_dataset.sh
bash scripts/prepare_dataset.sh
```

The data is downloaded to the `./LJSpeech-1.1` directory (on the host). The `./LJSpeech-1.1` directory is mounted under the `/workspace/fastpitch/LJSpeech-1.1` location in the NGC container. The complete dataset has the following structure:

```
./LJSpeech-1.1
├── mels             # (optional) Pre-calculated target mel-spectrograms; can be calculated online
├── metadata.csv     # Mapping of waveforms to utterances
├── pitch            # Fundamental frequency contours for input utterances; can be calculated online
├── README
└── wavs             # Raw waveforms
```

4. Start training.

```
bash scripts/train.sh
```

The training will produce a FastPitch model capable of generating mel-spectrograms from raw text. It will be serialized as a single `.pt` checkpoint file, along with a series of intermediate checkpoints. The script is configured for 8x GPU with at least 16GB of memory. Consult Training process and example configs to adjust to a different configuration or enable Automatic

```
Successfully built dllogger
Installing collected packages: typing-extensions, typeguard, more-itertools, tensorboardX, pynvml, librosa, inflect, dllogger
  Attempting uninstall: typing-extensions
    Found existing installation: typing-extensions 4.3.0
    Uninstalling typing-extensions-4.3.0:
      Successfully uninstalled typing-extensions-4.3.0
  Attempting uninstall: pynvml
    Found existing installation: pynvml 11.4.1
    Uninstalling pynvml-11.4.1:
      Successfully uninstalled pynvml-11.4.1
  Attempting uninstall: librosa
    Found existing installation: librosa 0.9.2
    Uninstalling librosa-0.9.2:
      Successfully uninstalled librosa-0.9.2
Successfully installed dllogger-1.0.0 inflect-7.3.1 librosa-0.9.0 more-itertools-10.4.0 pynvml-11.0.0 tensorboardX-2.0 typeguard-4.3.0 typing-extensions-4.12.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual e
nvironment instead: https://pip.pypa.io/warnings/venv
root@e5acd662193f:/workspace/ML/DeepLearningExamples/PyTorch/SpeechSynthesis/FastPitch# cd LJSpeech-1.1/
root@e5acd662193f:/workspace/ML/DeepLearningExamples/PyTorch/SpeechSynthesis/FastPitch/LJSpeech-1.1# ls
README  metadata.csv  wavs
root@e5acd662193f:/workspace/ML/DeepLearningExamples/PyTorch/SpeechSynthesis/FastPitch/LJSpeech-1.1# cd ..
root@e5acd662193f:/workspace/ML/DeepLearningExamples/PyTorch/SpeechSynthesis/FastPitch# bash scripts/prepare_dataset.sh
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_levels instead.
DLL 2024-08-26 10:38:05.457548 - PARAMETER dataset_path : LJSpeech-1.1
DLL 2024-08-26 10:38:05.457643 - PARAMETER wav_text_filelists : ['filelists/ljs_audio_text.txt']
DLL 2024-08-26 10:38:05.457690 - PARAMETER extract_mels : True
DLL 2024-08-26 10:38:05.457745 - PARAMETER extract_pitch : True
DLL 2024-08-26 10:38:05.457782 - PARAMETER save_alignment_priors : False
DLL 2024-08-26 10:38:05.457834 - PARAMETER log_file : preproc_log.json
DLL 2024-08-26 10:38:05.457888 - PARAMETER n_speakers : 1
DLL 2024-08-26 10:38:05.457941 - PARAMETER max_wav_value : 32768.0
DLL 2024-08-26 10:38:05.458025 - PARAMETER sampling_rate : 22050
DLL 2024-08-26 10:38:05.458097 - PARAMETER filter_length : 1024
DLL 2024-08-26 10:38:05.458161 - PARAMETER hop_length : 256
DLL 2024-08-26 10:38:05.458217 - PARAMETER win_length : 1024
DLL 2024-08-26 10:38:05.458259 - PARAMETER mel_fmin : 0.0
DLL 2024-08-26 10:38:05.458299 - PARAMETER mel_fmax : 8000.0
DLL 2024-08-26 10:38:05.458340 - PARAMETER n_mel_channels : 80
DLL 2024-08-26 10:38:05.458380 - PARAMETER f0_method : pyin
DLL 2024-08-26 10:38:05.458429 - PARAMETER batch_size : 1
DLL 2024-08-26 10:38:05.458484 - PARAMETER n_workers : 16
DLL 2024-08-26 10:38:05.458554 - PARAMETER symbol_set : english_basic
Processing filelists/ljs_audio_text.txt...
 13%|                                  | 1662/13100 [31:20<3:18:22,  1.04s/it]
```


Step 4 :- Start training below the Commands

Start training.

bash scripts/train.sh

In the root directory `./` of this repository, the `./train.py` script is used for training, while inference can be executed with the `./inference.py` script. The script `./models.py` is used to construct a model of the requested type and properties.

The repository is structured similarly to the NVIDIA Tacotron2 Deep Learning example so that they could be combined in more advanced use cases.

Parameters

In this section, we list the most important hyperparameters and command-line arguments, together with their default values that are used to train FastPitch.

- `--epochs` - number of epochs (default: 1000)
- `--learning-rate` - learning rate (default: 0.1)
- `--batch-size` - batch size for a single forward-backward step (default: 16)
- `--grad-accumulation` - number of steps over which gradients are accumulated (default: 2)
- `--amp` - use mixed precision training (default: disabled)
- `--load-pitch-from-disk` - pre-calculated fundamental frequency values, estimated before training, are loaded from the disk during training (default: enabled)
- `--energy-conditioning` - enables additional conditioning on energy (default: enabled)
- `--p-arpabet` - probability of choosing phonemic over graphemic representation for every word, if available (default: 1.0)

Step 6:- IF you Want to Inference

Start inference/predictions.
To synthesize audio, you will need a WaveGlow model, which generates waveforms based on mel-spectrograms generated with FastPitch. By now, a pre-trained model should have been downloaded by the `scripts/download_dataset.sh` script. Alternatively, to train WaveGlow from scratch, follow the instructions in NVIDIA/DeepLearningExamples/Tacotron2 and replace the checkpoint in the `./pretrained_models/waveglow` directory.
You can perform inference using the respective `.pt` checkpoints that are passed as `--fastpitch` and `--waveglow` arguments:

python inference.py \

```
    --cuda \
    --fastpitch output/<FastPitch checkpoint> \
    --energy-conditioning \
    --waveglow pretrained_models/waveglow/<WaveGlow checkpoint> \
    --wn-channels 256 \
    -i phrases/devset10.tsv \
    -o output/wavs_devset10
```

Performance

Benchmarking

The following section shows how to run benchmarks measuring the model performance in training and inference mode.

Training performance benchmark

To benchmark the training performance on a specific batch size, run:

- NVIDIA DGX A100 (8x A100 80GB)

```
AMP=true NUM_GPUS=1 BS=32 GRAD_ACCUMULATION=8 EPOCHS=10 bash scripts/train.sh
AMP=true NUM_GPUS=8 BS=32 GRAD_ACCUMULATION=1 EPOCHS=10 bash scripts/train.sh
AMP=false NUM_GPUS=1 BS=32 GRAD_ACCUMULATION=8 EPOCHS=10 bash scripts/train.sh
AMP=false NUM_GPUS=8 BS=32 GRAD_ACCUMULATION=1 EPOCHS=10 bash scripts/train.sh
```

- NVIDIA DGX-1 (8x V100 16GB)

```
AMP=true NUM_GPUS=1 BS=16 GRAD_ACCUMULATION=16 EPOCHS=10 bash scripts/train.sh
AMP=true NUM_GPUS=8 BS=16 GRAD_ACCUMULATION=2 EPOCHS=10 bash scripts/train.sh
AMP=false NUM_GPUS=1 BS=16 GRAD_ACCUMULATION=16 EPOCHS=10 bash scripts/train.sh
```

Results

The following sections provide details on how we achieved our performance and accuracy in training and inference.

Training accuracy results

Training accuracy: NVIDIA DGX A100 (8x A100 80GB)

Our results were obtained by running the `./platform/DGXA100_FastPitch_{AMP,TF32}_8GPU.sh` training script in the PyTorch 21.05-py3 NGC container on NVIDIA DGX A100 (8x A100 80GB) GPUs.

Loss (Model/Epoch)	50	250	500	750	1000	1250	1500
FastPitch AMP	3.35	2.89	2.79	2.71	2.68	2.64	2.61
FastPitch TF32	3.37	2.88	2.78	2.71	2.68	2.63	2.61

Training accuracy: NVIDIA DGX-1 (8x V100 16GB)

Our results were obtained by running the `./platform/DGX1_FastPitch_{AMP,FP32}_8GPU.sh` training script in the PyTorch 21.05-py3 NGC container on NVIDIA DGX-1 with 8x V100 16GB GPUs.

All of the results were produced using the `train.py` script as described in the Training process section of this document.

Loss (Model/Epoch)	50	250	500	750	1000	1250	1500

DeepLearningExamples / PyTorch / SpeechSynthesis / FastPitch / ↑ Top

Training accuracy: NVIDIA DGX-1 (8x V100 16GB)

Our results were obtained by running the `./platform/DGX1_FastPitch_{AMP,FP32}_8GPU.sh` training script in the PyTorch 21.05-py3 NGC container on NVIDIA DGX-1 with 8x V100 16GB GPUs.

All of the results were produced using the `train.py` script as described in the Training process section of this document.

Loss (Model/Epoch)	50	250	500	750	1000	1250	1500
FastPitch AMP	3.38	2.88	2.79	2.71	2.68	2.64	2.61
FastPitch FP32	3.38	2.89	2.80	2.71	2.68	2.65	2.62

Training performance results

Training performance: NVIDIA DGX A100 (8x A100 80GB)

Our results were obtained by running the `./platform/DGXA100_FastPitch_{AMP,TF32}_8GPU.sh` training script in the PyTorch 22.08-py3 NGC container on NVIDIA DGX A100 (8x A100 80GB) GPUs. Performance numbers, in output mel-scale spectrogram frames per second, were averaged over an entire training epoch.

Batch size / GPU	GPUs	Grad accumulation	Throughput - TF32	Throughput - mixed precision	Throughput speedup (TF32 to mixed precision)	Strong scaling - TF32	Strong scaling - mixed precision
128	1	2	141,028	148,149	1.05	1.00	1.00
64	4	1	525,879	614,857	1.17	3.73	4.15
32	8	1	914,350	1,022,722	1.12	6.48	6.90

DeepLearningExamples / PyTorch / SpeechSynthesis / FastPitch / ↑ Top

Training performance: NVIDIA DGX-1 (8x V100 16GB)

Our results were obtained by running the `./platform/DGX1_FastPitch_{AMP,FP32}_8GPU.sh` training script in the PyTorch 22.08-py3 NGC container on NVIDIA DGX-1 with 8x V100 16GB GPUs. Performance numbers, in output mel-scale spectrogram frames per second, were averaged over an entire training epoch.

Batch size / GPU	GPUs	Grad accumulation	Throughput - FP32	Throughput - mixed precision	Throughput speedup (FP32 to mixed precision)	Strong scaling - FP32	Strong scaling - mixed precision
16	1	16	31,863	83,761	2.63	1.00	1.00
16	4	4	117,971	269,143	2.28	3.70	3.21
16	8	2	225,826	435,799	1.93	7.09	5.20

Inference performance results

The following tables show inference statistics for the FastPitch and WaveGlow text-to-speech system, gathered from 100 inference runs. Latency is measured from the start of FastPitch inference to the end of WaveGlow inference. Throughput is measured as the number of generated audio samples per second at 22KHz. RTF is the real-time factor that denotes the number of seconds of speech generated in a second of wall-clock time per input utterance. The used WaveGlow model is a 256-channel model.

Note that performance numbers are related to the length of input. The numbers reported below were taken with a moderate length of 128 characters. Longer utterances yield higher RTF, as the generator is fully parallel.

Inference performance: NVIDIA DGX A100 (1x A100 80GB)

Our results were obtained by running the `./scripts/inference_benchmark.sh` inferencing benchmarking script in the PyTorch 22.08-py3 NGC container on NVIDIA DGX A100 (1x A100 80GB) GPU.

FastPitch (TorchScript, denoising)

Batch size	Precision	Avg latency (s)	Latency tolerance interval 90% (s)	Latency tolerance interval 95% (s)	Latency tolerance interval 99% (s)	Throughput (frames/sec)	Speed-up with mixed precision	Avg RTF
1	FP16	0.005	0.006	0.006	0.006	120,333	0.97	1397.07
4	FP16	0.006	0.006	0.006	0.006	424,053	1.12	1230.81

DeepLearningExamples / PyTorch / SpeechSynthesis / FastPitch / ↑ Top

Inference performance: NVIDIA DGX-1 (1x V100 16GB)

Our results were obtained by running the `./scripts/inference_benchmark.sh` script in the PyTorch 22.08-py3 NGC container. The input utterance has 128 characters, synthesized audio has 8.05 s.

FastPitch (TorchScript, denoising)

Batch size	Precision	Avg latency (s)	Latency tolerance interval 90% (s)	Latency tolerance interval 95% (s)	Latency tolerance interval 99% (s)	Throughput (frames/sec)	Speed-up with mixed precision	Avg RTF
1	FP16	0.007	0.008	0.008	0.008	88,908	1.10	1032.23
4	FP16	0.010	0.010	0.010	0.010	272,564	1.73	791.12
8	FP16	0.013	0.013	0.013	0.013	415,263	2.35	602.65
1	FP32	0.008	0.008	0.008	0.009	80,558	-	935.28
4	FP32	0.017	0.017	0.017	0.017	157,114	-	456.02
8	FP32	0.030	0.030	0.030	0.030	176,754	-	256.51

FastPitch + HiFi-GAN (TorchScript, denoising)

			Latency	Latency	Latency			