NVIDIA / **NeMo** Public

NeMo: a toolkit for conversational AI

nvidia.github.io/nemo/

Apache-2.0 license

6.5k stars **1.5k** forks

☆ Star Watch

Code Issues 40 Pull requests 52 Discussions Actions Projects Security 1 In

main ▾ ...

Slyne ... ✓ 17 hours ago

View code

repo status Active | docs passing | CodeQL passing | License Apache 2.0 | pypi package 1.17.0 | python 3.8 | 3.9
downloads 483k | code style black

NVIDIA NeMo

Introduction

NVIDIA NeMo is a conversational AI toolkit built for researchers working on automatic speech recognition (ASR), text-to-speech synthesis (TTS), large language models (LLMs), and natural language processing (NLP). The primary objective of NeMo is to help researchers from industry and academia to reuse prior work (code and pretrained models) and make it easier to create new conversational AI models.

All NeMo models are trained with Lightning and training is automatically scalable to 1000s of GPUs. Additionally, NeMo Megatron LLM models can be trained up to 1 trillion parameters using tensor and pipeline model parallelism. NeMo models can be optimized for inference and deployed for production use-cases with NVIDIA Riva.

Getting started with NeMo is simple. State of the Art pretrained NeMo models are freely available on HuggingFace Hub and NVIDIA NGC. These models can be used to transcribe audio, synthesize speech, or translate text in just a few lines of code.

We have extensive tutorials that can all be run on Google Colab.

For advanced users that want to train NeMo models from scratch or finetune existing NeMo models we have a full suite of example scripts that support multi-GPU/multi-node training

README.rst

and also has an Autoconfigurator which can be used to find the optimal model parallel configuration for training on a specific cluster.

Also see our introductory video for a high level overview of NeMo.

Key Features

- *Speech processing*
  - HuggingFace Space for Audio Transcription (File, Microphone and YouTube)
  - *Automatic Speech Recognition (ASR)*
    - ***Supported ASR models: https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/models.html***
      - Jasper, QuartzNet, CitriNet, ContextNet
      - Conformer-CTC, Conformer-Transducer, FastConformer-CTC, FastConformer-Transducer
      - Squeezeformer-CTC and Squeezeformer-Transducer
      - LSTM-Transducer (RNNT) and LSTM-CTC
    - ***Supports the following decoders/losses:***
      - CTC
      - Transducer/RNNT
      - Hybrid Transducer/CTC
      - NeMo Original Multi-blank Transducers
    - Streaming/Buffered ASR (CTC/Transducer) - Chunked Inference Examples
    - Cache-aware Streaming Conformer - https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/models.html#cache-aware-streaming-conformer
    - Beam Search decoding
    - Language Modelling for ASR: N-gram LM in fusion with Beam Search decoding, Neural Rescoring with Transformer
    - Support of long audios for Conformer with memory efficient local attention
  - Speech Classification, Speech Command Recognition and Language Identification: MatchboxNet (Command Recognition), AmberNet (LangID)
  - *Voice activity Detection (VAD): MarbleNet*
    - ASR with VAD Inference - Example
  - Speaker Recognition: TitaNet, ECAPA_TDNN, SpeakerNet
  - *Speaker Diarization*
    - Clustering Diarizer: TitaNet, ECAPA_TDNN, SpeakerNet
    - Neural Diarizer: MSDD (Multi-scale Diarization Decoder)
  - Speech Intent Detection and Slot Filling: Conformer-Transformer
  - Pretrained models on different languages.: English, Spanish, German, Russian, Chinese, French, Italian, Polish, ...
  - NGC collection of pre-trained speech processing models.
- *Natural Language Processing*
  - NeMo Megatron pre-training of Large Language Models
  - Neural Machine Translation (NMT)
  - Punctuation and Capitalization
  - Token classification (named entity recognition)
  - Text classification

  - Joint Intent and Slot Classification
  - Question answering
  - GLUE benchmark
  - Information retrieval
  - Entity Linking
  - Dialogue State Tracking
  - Prompt Learning
  - NGC collection of pre-trained NLP models.
  - Synthetic Tabular Data Generation

- ***Speech synthesis (TTS)***
  - Spectrogram generation: Tacotron2, GlowTTS, TalkNet, FastPitch, FastSpeech2, Mixer-TTS, Mixer-TTS-X
  - Vocoders: WaveGlow, SqueezeWave, UniGlow, MelGAN, HiFiGAN, UnivNet
  - End-to-end speech generation: FastPitch_HifiGan_E2E, FastSpeech2_HifiGan_E2E, VITS
  - NGC collection of pre-trained TTS models.

- ***Tools***
  - Text Processing (text normalization and inverse text normalization)
  - CTC-Segmentation tool
  - Speech Data Explorer: a dash-based tool for interactive exploration of ASR/TTS datasets

Built for speed, NeMo can utilize NVIDIA's Tensor Cores and scale out training to multiple GPUs and multiple nodes.

Requirements

1. Python 3.8 or above
2. Pytorch 1.10.0 or above
3. NVIDIA GPU for training

Documentation

Version	Status	Description
Latest	docs passing	Documentation of the latest (i.e. main) branch.
Stable	docs passing	Documentation of the stable (i.e. most recent release) branch.

Tutorials

A great way to start with NeMo is by checking one of our tutorials.

Getting help with NeMo

FAQ can be found on NeMo's Discussions board. You are welcome to ask questions or start discussions there.

Installation

Conda

We recommend installing NeMo in a fresh Conda environment.

```
conda create --name nemo python==3.8.10
conda activate nemo
```

Install PyTorch using their configurator.

```
conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
```

The command used to install PyTorch may depend on your system. Please use the configurator linked above to find the right command for your system.

Pip

Use this installation mode if you want the latest released version.

```
apt-get update && apt-get install -y libsndfile1 ffmpeg
pip install Cython
pip install nemo_toolkit['all']
```

Depending on the shell used, you may need to use `"nemo_toolkit[all]"` instead in the above command.

Pip from source

Use this installation mode if you want the version from a particular GitHub branch (e.g main).

```
apt-get update && apt-get install -y libsndfile1 ffmpeg
pip install Cython
python -m pip install git+https://github.com/NVIDIA/NeMo.git@{BRANCH}#egg=nemo_toolkit[all]
```

From source

Use this installation mode if you are contributing to NeMo.

```
apt-get update && apt-get install -y libsndfile1 ffmpeg
git clone https://github.com/NVIDIA/NeMo
cd NeMo
./reinstall.sh
```

If you only want the toolkit without additional conda-based dependencies, you may replace `reinstall.sh` with `pip install -e .` when your PWD is the root of the NeMo repository.

RNNT

Note that RNNT requires numba to be installed from conda.

```
conda remove numba
pip uninstall numba
conda install -c conda-forge numba
```

NeMo Megatron

NeMo Megatron training requires NVIDIA Apex and Megatron-core to be installed. Install them manually if not using the NVIDIA PyTorch container.

To install Apex, run

```
git clone https://github.com/NVIDIA/apex.git
cd apex
git checkout 57057e2fcf1c084c0fcc818f55c0ff6ea1b24ae2
pip install -v --disable-pip-version-check --no-cache-dir --global-option="--cpp_ext" --global-option="--
```

To install Megatron-core, run

```
git clone https://github.com/NVIDIA/Megatron-LM.git
cd Megatron-LM
git checkout 3db2063b1ff992a971ba18f7101eecc9c4e90f03
pip install -e .
```

It is highly recommended to use the NVIDIA PyTorch or NeMo container if having issues installing Apex or any other dependencies.

While installing Apex, it may raise an error if the CUDA version on your system does not match the CUDA version torch was compiled with. This raise can be avoided by commenting it here: https://github.com/NVIDIA/apex/blob/master/setup.py#L32

cuda-nvprof is needed to install Apex. The version should match the CUDA version that you are using:

```
conda install -c nvidia cuda-nvprof=11.8
```

packaging is also needed:

```
pip install -y packaging
```

Transformer Engine

NeMo Megatron GPT has been integrated with NVIDIA Transformer Engine Transformer Engine enables FP8 training on NVIDIA Hopper GPUs. Install it manually if not using the NVIDIA PyTorch container.

```
pip install --upgrade git+https://github.com/NVIDIA/TransformerEngine.git@stable
```

It is highly recommended to use the NVIDIA PyTorch or NeMo container if having issues installing Transformer Engine or any other dependencies.

Transformer Engine requires PyTorch to be built with CUDA 11.8.

NeMo Text Processing

NeMo Text Processing, specifically (Inverse) Text Normalization, is now a separate repository https://github.com/NVIDIA/NeMo-text-processing.

Docker containers:

We release NeMo containers alongside NeMo releases. For example, NeMo `r1.16.0` comes with container `nemo:23.01`, you may find more details about released containers in releases page.

To use built container, please run

```
docker pull nvcr.io/nvidia/nemo:23.01
```

To build a nemo container with Dockerfile from a branch, please run

```
DOCKER_BUILDKIT=1 docker build -f Dockerfile -t nemo:latest .
```

If you chose to work with main branch, we recommend using NVIDIA's PyTorch container version 23.03-py3 and then installing from GitHub.

```
docker run --gpus all -it --rm -v <nemo_github_folder>:/NeMo --shm-size=8g \
-p 8888:8888 -p 6006:6006 --ulimit memlock=-1 --ulimit \
stack=67108864 --device=/dev/snd nvcr.io/nvidia/pytorch:23.03-py3
```

Examples

Many examples can be found under the "Examples" folder.

Contributing

We welcome community contributions! Please refer to the CONTRIBUTING.md CONTRIBUTING.md for the process.

Publications

We provide an ever growing list of publications that utilize the NeMo framework. Please refer to PUBLICATIONS.md. We welcome the addition of your own articles to this list !

License

NeMo is under Apache 2.0 license.

Releases 39

NVIDIA Neural Modules 1.17.0 Latest
3 weeks ago

+ 38 releases

Packages

No packages published

Contributors 224

+ 213 contributors

Environments 1

github-pages Active

Languages

- ● **Python** 75.6%
- ● **Jupyter Notebook** 23.9%
- ● **Shell** 0.2%
- ● **C++** 0.2%
- ● **HTML** 0.1%
- ● **Dockerfile** 0.0%