pytorch / pytorch

Type / to search

Code | Issues 5k+ | Pull requests 838 | Actions | Projects 30 | Wiki | Security 1 | Insights

pytorch Public

main |),181 Branches | 4,282 Tags | Go to file t | Go | + | e | Add fil | Code

ezyang and pytorchmergebot R. 25f7219 · 5 minutes ago 68,767 Commits

.ci	[executorch hash update] u...	13 hours ago
.circleci	[EZ][BE] Move `build_andro`...	last week
.ctags.d	Add a .ctags.d/ toplevel dire...	5 years ago
.devcontainer	[Dev Container]Add readme...	4 months ago
.github	Fix mergeability check for g...	15 hours ago
.vscode	[3/3] Update `.pyi` Python st...	last year
android	Fix syntax highlighting in an...	last week
aten	fix lint for cudnn codes (#11...	19 minutes ago
benchmarks	Move skip sets into a new fil...	2 days ago
binaries	Remaining replacement of c...	4 months ago
c10	Get Device instance with co...	2 days ago
caffe2	Add missing cuda libraries f...	yesterday
cmake	[ROCm] backward compatib...	9 hours ago
docs	[Export] Remove ScriptObje...	17 hours ago
functorch	Fix missing words in READM...	3 weeks ago
ios	[BE]: Enable F821 and fix bu...	last month
modules	[Cmake] Check that gcc-9....	2 months ago
mypy_plugins	Enable UFMT on a bunch of ...	6 months ago
scripts	Update update_failures.py g...	yesterday
test	Realize inputs to DynamicSc...	5 minutes ago
third_party	Update PocketFFT submod...	16 hours ago
tools	[Exception] [6/N] Remove u...	2 days ago
torch	Realize inputs to DynamicSc...	5 minutes ago
torchgen	Fix return type hint for list ty...	18 hours ago

About

Tensors and Dynamic neural networks in Python with strong GPU acceleration

pytorch.org

#python #machine-learning #deep-learning #neural-network #gpu #numpy #autograd #tensor

Readme
View license
Code of conduct
Security policy
Cite this repository
Activity
Custom properties
74.8k stars
1.7k watching
20.5k forks
Report repository

Releases 48

PyTorch 2.1.2 Release, bug fi... Latest
on Dec 14, 2023

+ 47 releases

Packages

No packages published

Used by 370k

+ 370,416

Contributors 3,117

+ 3,103 contributors

.bazelignore	put third_party/ittapi/ in .ba...	8 months ago
.bazelrc	[bazel] add python targets (...	8 months ago
.bazelversion	update Bazel to the latest re...	10 months ago
.buckconfig.oss	[BE] Add regression test for ...	10 months ago
.clang-format	[BE][MPS] Add MPS to clan...	10 months ago
.clang-tidy	[BE]: Enable readability-red...	last month
.cmakelintrc	Fix/relax CMake linter rules (...	4 years ago
.coveragerc	Use JIT Plug-in for coverag...	3 years ago
.dockerignore	Add .dockerignore. (#3333)	7 years ago
.flake8	Enhance torch.vmap suppor...	4 days ago
.gdbinit	gdb special command to pri...	3 years ago
.git-blame-ignore-revs	Add 116583 to `.git-blame-`...	3 weeks ago
.gitattributes	third_party: Fix build_bundl...	2 years ago
.gitignore	[no ci] Add `.watchman` to .git...	2 months ago
.gitmodules	Revert "[Reland2] Update N...	last month
.isort.cfg	Remove pyproject.toml (#61...	3 years ago
.lintrunner.toml	fix lint for cudnn codes (#11...	19 minutes ago
.lldbinit	Add helpful pretty pretting s...	10 months ago
BUCK.oss	[4] move pt_operator_librar...	2 years ago
BUILD.bazel	[BE] [cuDNN] Always build ...	3 weeks ago
CITATION.cff	use cff standard for citation ...	2 years ago
CMakeLists.txt	[2/4] Intel GPU Runtime Ups...	last week
CODEOWNERS	[no ci] Add pytorch-dev-infr...	3 weeks ago
CODE_OF_CONDUCT...	Create CODE_OF_CONDUC...	4 years ago
CONTRIBUTING.md	[DevX] Add tool and doc on ...	last month
Dockerfile	Dockerfile; Add cuda bin to ...	2 weeks ago
GLOSSARY.md	Add remaining ToCs to ToC l...	3 years ago
LICENSE	[Model Averaging] Support ...	2 years ago
MANIFEST.in	fix citation file in MANIFEST ...	2 years ago
Makefile	make triton uses the wheel ...	10 months ago
NOTICE	Add uint8 support for interp...	last year
README.md	[CMake] Explicitly error out ...	18 hours ago

Deployments 500+

- upload-stats
- conda-aws-upload 2 hours ago
- pytorchbot-env

+ more deployments

Languages

- Python 49.4%
- C++ 40.9%
- Cuda 3.8%
- C 2.1%
- Objective-C++ 1.4%
- CMake 0.8%
- Other 1.6%

RELEASE.md	[release] Add Launch Execu...	3 weeks ago
SECURITY.md	Update SECURITY.MD (#93...	last year
WORKSPACE	[BE] [cuDNN] Always build ...	3 weeks ago
aten.bzl	add explicit vectorization fo...	10 months ago
buckbuild.bzl	[pt-vulkan] Enable Python c...	last month
build.bzl	[ROCm] Disabling Kernel As...	last month
build_variables.bzl	add _amp_foreach_non_fini...	last week
c2_defs.bzl	[caffe2] Add option for build...	last month
c2_test_defs.bzl	Add all bzl files per D36874...	2 years ago
defs.bzl	Remove unused build syste...	4 months ago
docker.Makefile	[releng] Docker release Ref...	last month
mypy-inductor.ini	[mypy] Enable follow_impor...	yesterday
mypy-strict.ini	[pytree] Extract reusable ge...	3 months ago
mypy.ini	Move test_utils.py back to ...	2 months ago
pt_ops.bzl	[xplat][buck2][typing] Fix ty...	4 months ago
pt_template_srcs.bzl	Use global variables to regis...	4 months ago
pyproject.toml	[BE]: Enable F821 and fix bu...	last month
pytest.ini	Reduce pytest prints (#1170...	3 days ago
requirements-flake8.txt	Update TorchFix to 0.2.0 (#1...	2 months ago
requirements.txt	Pin the version of expecttes...	last month
setup.py	export ATen/native/sparse/*....	19 hours ago
ubsan.supp	Upgrade Pybind submodule...	7 months ago
ufunc_defs.bzl	move build_variables.bzl an...	2 years ago
version.txt	[releng] version 2.2 -> 2.3 (...	last month

PyTorch

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

You can reuse your favorite Python packages such as NumPy, SciPy, and Cython to extend PyTorch when needed.

Our trunk health (Continuous Integration signals) can be found at hud.pytorch.org.

More About PyTorch

Learn the basics of PyTorch

At a granular level, PyTorch is a library that consists of the following components:

Component	Description
torch	A Tensor library like NumPy, with strong GPU support
torch.autograd	A tape-based automatic differentiation library that supports all differentiable Tensor operations in torch
torch.jit	A compilation stack (TorchScript) to create serializable and optimizable models from PyTorch code
torch.nn	A neural networks library deeply integrated with autograd designed for maximum flexibility

torch.multiprocessing	Python multiprocessing, but with magical memory sharing of torch Tensors across processes. Useful for data loading and Hogwild training
torch.utils	DataLoader and other utility functions for convenience

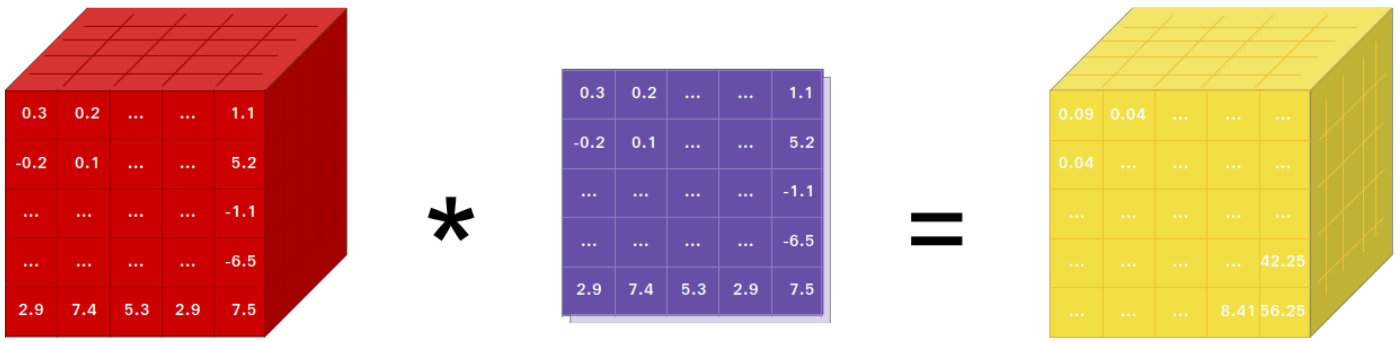
Usually, PyTorch is used either as:

- A replacement for NumPy to use the power of GPUs.
- A deep learning research platform that provides maximum flexibility and speed.

Elaborating Further:

A GPU-Ready Tensor Library

If you use NumPy, then you have used Tensors (a.k.a. ndarray).



PyTorch provides Tensors that can live either on the CPU or the GPU and accelerates the computation by a huge amount.

We provide a wide variety of tensor routines to accelerate and fit your scientific computation needs such as slicing, indexing, mathematical operations, linear algebra, reductions. And they are fast!

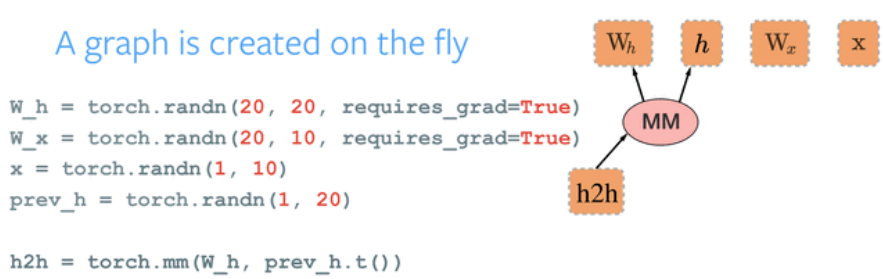
Dynamic Neural Networks: Tape-Based Autograd

PyTorch has a unique way of building neural networks: using and replaying a tape recorder.

Most frameworks such as TensorFlow, Theano, Caffe, and CNTK have a static view of the world. One has to build a neural network and reuse the same structure again and again. Changing the way the network behaves means that one has to start from scratch.

With PyTorch, we use a technique called reverse-mode auto-differentiation, which allows you to change the way your network behaves arbitrarily with zero lag or overhead. Our inspiration comes from several research papers on this topic, as well as current and past work such as torch-autograd, autograd, Chainer, etc.

While this technique is not unique to PyTorch, it's one of the fastest implementations of it to date. You get the best of speed and flexibility for your crazy research.



Python First

PyTorch is not a Python binding into a monolithic C++ framework. It is built to be deeply integrated into Python. You can use it naturally like you would use NumPy / SciPy / scikit-learn etc. You can write your new neural network layers in Python itself, using your favorite libraries and use packages such as Cython and Numba. Our goal is to not reinvent the wheel where appropriate.

Imperative Experiences

PyTorch is designed to be intuitive, linear in thought, and easy to use. When you execute a line of code, it gets executed. There isn't an asynchronous view of the world. When you drop into a debugger or receive error messages and stack traces, understanding them is straightforward. The stack trace points to exactly where your code was defined. We hope you never spend hours debugging your code because of bad stack traces or asynchronous and opaque execution engines.

Fast and Lean

PyTorch has minimal framework overhead. We integrate acceleration libraries such as Intel MKL and NVIDIA (cuDNN, NCCL) to maximize speed. At the core, its CPU and GPU Tensor and neural network backends are mature and have been tested for years.

Hence, PyTorch is quite fast — whether you run small or large neural networks.

The memory usage in PyTorch is extremely efficient compared to Torch or some of the alternatives. We've written custom memory allocators for the GPU to make sure that your deep learning models are maximally memory efficient. This enables you to train bigger deep learning models than before.

Extensions Without Pain

Writing new neural network modules, or interfacing with PyTorch's Tensor API was designed to be straightforward and with minimal abstractions.

You can write new neural network layers in Python using the torch API or your favorite NumPy-based libraries such as SciPy.

If you want to write your layers in C/C++, we provide a convenient extension API that is efficient and with minimal boilerplate. No wrapper code needs to be written. You can see a tutorial here and an example here.

Installation

Binaries

Commands to install binaries via Conda or pip wheels are on our website: https://pytorch.org/get-started/locally/

NVIDIA Jetson Platforms

Python wheels for NVIDIA's Jetson Nano, Jetson TX1/TX2, Jetson Xavier NX/AGX, and Jetson AGX Orin are provided here and the L4T container is published here

They require JetPack 4.2 and above, and @dusty-nv and @ptrblck are maintaining them.

From Source

Prerequisites

If you are installing from source, you will need:

- Python 3.8 or later (for Linux, Python 3.8.1+ is needed)

- A compiler that fully supports C++17, such as clang or gcc (especially for aarch64, gcc 9.4.0 or newer is required)

We highly recommend installing an Anaconda environment. You will get a high-quality BLAS library (MKL) and you get controlled dependency versions regardless of your Linux distro.

If you want to compile with CUDA support, select a supported version of CUDA from our support matrix, then install the following:

- NVIDIA CUDA
- NVIDIA cuDNN v8.5 or above
- Compiler compatible with CUDA

Note: You could refer to the cuDNN Support Matrix for cuDNN versions with the various supported CUDA, CUDA driver and NVIDIA hardware

If you want to disable CUDA support, export the environment variable `USE_CUDA=0` . Other potentially useful environment variables may be found in `setup.py` .

If you are building for NVIDIA's Jetson platforms (Jetson Nano, TX1, TX2, AGX Xavier), Instructions to install PyTorch for Jetson Nano are available here

If you want to compile with ROCm support, install

- AMD ROCm 4.0 and above installation
- ROCm is currently supported only for Linux systems.

If you want to disable ROCm support, export the environment variable `USE_ROCM=0` . Other potentially useful environment variables may be found in `setup.py` .

Install Dependencies

Common

```
conda install cmake ninja
# Run this command from the PyTorch directory after cloning the source code using the “Get the PyTorch Source“ se
pip install -r requirements.txt
```

On Linux

```
conda install intel::mkl-static intel::mkl-include
# CUDA only: Add LAPACK support for the GPU if needed
conda install -c pytorch magma-cuda110  # or the magma-cuda* that matches your CUDA version from https://anaconda

# (optional) If using torch.compile with inductor/triton, install the matching version of triton
# Run from the pytorch directory after cloning
make triton
```

On MacOS

```
# Add this package on intel x86 processor machines only
conda install intel::mkl-static intel::mkl-include
# Add these packages if torch.distributed is needed
conda install pkg-config libuv
```

On Windows

```
conda install intel::mkl-static intel::mkl-include
# Add these packages if torch.distributed is needed.
```

```
# Distributed package support on Windows is a prototype feature and is subject to changes.
conda install -c conda-forge libuv=1.39
```

Get the PyTorch Source

```
git clone --recursive https://github.com/pytorch/pytorch
cd pytorch
# if you are updating an existing checkout
git submodule sync
git submodule update --init --recursive
```

Install PyTorch

On Linux

If you would like to compile PyTorch with new C++ ABI enabled, then first run this command:

```
export _GLIBCXX_USE_CXX11_ABI=1
```

If you're compiling for AMD ROCm then first run this command:

```
# Only run this if you're compiling for ROCm
python tools/amd_build/build_amd.py
```

Install PyTorch

```
export CMAKE_PREFIX_PATH=${CONDA_PREFIX:-"$(dirname $(which conda))/../"}
python setup.py develop
```

> *Aside:* If you are using Anaconda, you may experience an error caused by the linker:
>
> ```
> build/temp.linux-x86_64-3.7/torch/csrc/stub.o: file not recognized: file format not recognized
> collect2: error: ld returned 1 exit status
> error: command 'g++' failed with exit status 1
> ```
>
> This is caused by `ld` from the Conda environment shadowing the system `ld`. You should use a newer version of Python that fixes this issue. The recommended Python version is 3.8.1+.

On macOS

```
python3 setup.py develop
```

On Windows

Choose Correct Visual Studio Version.

PyTorch CI uses Visual C++ BuildTools, which come with Visual Studio Enterprise, Professional, or Community Editions. You can also install the build tools from https://visualstudio.microsoft.com/visual-cpp-build-tools/. The build tools *do not* come with Visual Studio Code by default.

If you want to build legacy python code, please refer to Building on legacy code and CUDA

CPU-only builds

In this mode PyTorch computations will run on your CPU, not your GPU

```
conda activate
python setup.py develop
```

Note on OpenMP: The desired OpenMP implementation is Intel OpenMP (iomp). In order to link against iomp, you'll need to manually download the library and set up the building environment by tweaking `CMAKE_INCLUDE_PATH` and `LIB`. The instruction here is an example for setting up both MKL and Intel OpenMP. Without these configurations for CMake, Microsoft Visual C OpenMP runtime (vcomp) will be used.

CUDA based build

In this mode PyTorch computations will leverage your GPU via CUDA for faster number crunching

NVTX is needed to build Pytorch with CUDA. NVTX is a part of CUDA distributive, where it is called "Nsight Compute". To install it onto an already installed CUDA run CUDA installation once again and check the corresponding checkbox. Make sure that CUDA with Nsight Compute is installed after Visual Studio.

Currently, VS 2017 / 2019, and Ninja are supported as the generator of CMake. If `ninja.exe` is detected in `PATH`, then Ninja will be used as the default generator, otherwise, it will use VS 2017 / 2019.
If Ninja is selected as the generator, the latest MSVC will get selected as the underlying toolchain.

Additional libraries such as Magma, oneDNN, a.k.a. MKLDNN or DNNL, and Sccache are often needed. Please refer to the installation-helper to install them.

You can refer to the build_pytorch.bat script for some other environment variables configurations

```cmd
:: Set the environment variables after you have downloaded and unzipped the mkl package,
:: else CMake would throw an error as `Could NOT find OpenMP`.
set CMAKE_INCLUDE_PATH={Your directory}\mkl\include
set LIB={Your directory}\mkl\lib;%LIB%

:: Read the content in the previous section carefully before you proceed.
:: [Optional] If you want to override the underlying toolset used by Ninja and Visual Studio with CUDA, please ru
:: "Visual Studio 2019 Developer Command Prompt" will be run automatically.
:: Make sure you have CMake >= 3.12 before you do this when you use the Visual Studio generator.
set CMAKE_GENERATOR_TOOLSET_VERSION=14.27
set DISTUTILS_USE_SDK=1
for /f "usebackq tokens=*" %i in (`"%ProgramFiles(x86)%\Microsoft Visual Studio\Installer\vswhere.exe" -version [

:: [Optional] If you want to override the CUDA host compiler
set CUDAHOSTCXX=C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.27.29110\bin\HostX

python setup.py develop
```

Adjust Build Options (Optional)

You can adjust the configuration of cmake variables optionally (without building first), by doing the following. For example, adjusting the pre-detected directories for CuDNN or BLAS can be done with such a step.

On Linux

```
export CMAKE_PREFIX_PATH=${CONDA_PREFIX:-"$(dirname $(which conda))/../"}
python setup.py build --cmake-only
ccmake build  # or cmake-gui build
```

On macOS

```
export CMAKE_PREFIX_PATH=${CONDA_PREFIX:-"$(dirname $(which conda))/../"}
MACOSX_DEPLOYMENT_TARGET=10.9 CC=clang CXX=clang++ python setup.py build --cmake-only
ccmake build  # or cmake-gui build
```

Docker Image

Using pre-built images

You can also pull a pre-built docker image from Docker Hub and run with docker v19.03+

```
docker run --gpus all --rm -ti --ipc=host pytorch/pytorch:latest
```

README Code of conduct License Security

data loaders) the default shared memory segment size that container runs with is not enough, and you should increase shared memory size either with `--ipc=host` or `--shm-size` command line options to `nvidia-docker run`.

Building the image yourself

NOTE: Must be built with a docker version > 18.06

The `Dockerfile` is supplied to build images with CUDA 11.1 support and cuDNN v8. You can pass `PYTHON_VERSION=x.y` make variable to specify which Python version is to be used by Miniconda, or leave it unset to use the default.

```
make -f docker.Makefile
# images are tagged as docker.io/${your_docker_username}/pytorch
```

You can also pass the `CMAKE_VARS="..."` environment variable to specify additional CMake variables to be passed to CMake during the build. See setup.py for the list of available variables.

```
CMAKE_VARS="BUILD_CAFFE2=ON BUILD_CAFFE2_OPS=ON" make -f docker.Makefile
```

Building the Documentation

To build documentation in various formats, you will need Sphinx and the readthedocs theme.

```
cd docs/
pip install -r requirements.txt
```

You can then build the documentation by running `make <format>` from the `docs/` folder. Run `make` to get a list of all available output formats.

If you get a katex error run `npm install katex`. If it persists, try `npm install -g katex`

> Note: if you installed `nodejs` with a different package manager (e.g., `conda`) then `npm` will probably install a version of `katex` that is not compatible with your version of `nodejs` and doc builds will fail. A combination of versions that is known to work is

> is not compatible with your version of `nodejs` and doc builds will fail. A combination of versions that is known to work is `node@6.13.1` and `katex@0.13.18` . To install the latter with `npm` you can run `npm install -g katex@0.13.18`

Previous Versions

Installation instructions and binaries for previous PyTorch versions may be found on our website.

Getting Started

Three-pointers to get you started:

- Tutorials: get you started with understanding and using PyTorch
- Examples: easy to understand PyTorch code across all domains
- The API Reference
- Glossary

Resources

- PyTorch.org
- PyTorch Tutorials
- PyTorch Examples
- PyTorch Models
- Intro to Deep Learning with PyTorch from Udacity
- Intro to Machine Learning with PyTorch from Udacity
- Deep Neural Networks with PyTorch from Coursera
- PyTorch Twitter
- PyTorch Blog
- PyTorch YouTube

Communication

- Forums: Discuss implementations, research, etc. https://discuss.pytorch.org
- GitHub Issues: Bug reports, feature requests, install issues, RFCs, thoughts, etc.
- Slack: The PyTorch Slack hosts a primary audience of moderate to experienced PyTorch users and developers for general chat, online discussions, collaboration, etc. If you are a beginner looking for help, the primary medium is PyTorch Forums. If you need a slack invite, please fill this form: https://goo.gl/forms/PP1AGvNHpSaJP8to1
- Newsletter: No-noise, a one-way email newsletter with important announcements about PyTorch. You can sign-up here: https://eepurl.com/cbG0rv
- Facebook Page: Important announcements about PyTorch. https://www.facebook.com/pytorch
- For brand guidelines, please visit our website at pytorch.org

Releases and Contributing

Typically, PyTorch has three minor releases a year. Please let us know if you encounter a bug by filing an issue.

We appreciate all contributions. If you are planning to contribute back bug-fixes, please do so without any further discussion.

If you plan to contribute new features, utility functions, or extensions to the core, please first open an issue and discuss the feature with us. Sending a PR without discussion might end up resulting in a rejected PR because we might be taking the core in a different direction than you might be aware of.

To learn more about making a contribution to Pytorch, please see our Contribution page. For more information about PyTorch releases, see Release page.

The Team

PyTorch is a community-driven project with several skillful engineers and researchers contributing to it.

PyTorch is currently maintained by Soumith Chintala, Gregory Chanan, Dmytro Dzhulgakov, Edward Yang, and Nikita Shulga with major contributions coming from hundreds of talented individuals in various forms and means. A non-exhaustive but growing list needs to mention: Trevor Killeen, Sasank Chilamkurthy, Sergey Zagoruyko, Adam Lerer, Francisco Massa, Alykhan Tejani, Luca Antiga, Alban Desmaison, Andreas Koepf, James Bradbury, Zeming Lin, Yuandong Tian, Guillaume Lample, Marat Dukhan, Natalia Gimelshein, Christian Sarofeen, Martin Raison, Edward Yang, Zachary Devito.

Note: This project is unrelated to hughperkins/pytorch with the same name. Hugh is a valuable contributor to the Torch community and has helped with many things Torch and PyTorch.