pytorch / **pytorch** Public

Tensors and Dynamic neural networks in Python with strong GPU acceleration

pytorch.org

View license

90.5k stars 24.3k forks Branches Tags Activity

Star Notifications

Code Issues 5k+ Pull requests 1.3k Actions Projects 12 Wiki Security 2 Insights

main 5231 Branches 1260 Tags Go to file Go to file Code

davidberard98 and **pytorchmergebot** [triton pin][test] relax codecache test checks for number of triton a...

6[illegible]da6 22 minutes ago

.ci	[CI][CUDA] Migrate remaining cu118 jobs to cu...	4 hours ago
.circleci	Fix vs2022 caused AVX512 illegal instruction is...	2 weeks ago
.ctags.d	Add a .ctags.d/ toplevel directory (#18827)	6 years ago
.devcontainer	Devcontainer: Optimize apt-get commands to ...	last month
.github	[CI] Reuse old whl on PRs (#154662)	1 hour ago
.vscode	Remove lint dependency `ufmt` (#132573)	11 months ago
android	[pytorch] Delete TorchScript based Android de...	2 weeks ago
aten	Enable ConvTranspose3D for FP32 and Compl...	8 hours ago
benchmarks	Vary batch size when running dynamic shapes ...	6 hours ago
binaries	Remove binaries/benchmark_args.h (#147920)	4 months ago
c10	[BE] Introduce torch.AcceleratorError (#152023)	yesterday
caffe2	Revert "[BE] Cleanup old ExecuTorch codegen ...	8 hours ago
cmake	Revert "Use 3.27 as the minimum CMake versi...	3 days ago
docs	add reference to stances from dynamic shape...	6 hours ago
functorch	Removing conda references from PyTorch Doc...	2 weeks ago
mypy_plugins	Revert "Use absolute path `path.resolve()` -> ...	5 months ago
scripts	Update lint_urls.sh (#154838)	10 hours ago
test	[triton pin][test] relax codecache test checks f...	22 minutes ago
third_party	Bump pocketfft submodule to the latest (#154...	10 hours ago
tools	Revert "[BE] Cleanup old ExecuTorch codegen ...	8 hours ago
torch	[dynamo] fix pruning locals with ShapeEnvSou...	40 minutes ago
torchgen	Revert "[BE] Cleanup old ExecuTorch codegen ...	8 hours ago
.bazelignore	[rfc] opentelemetry in pytorch (#122999)	last year
.bazelrc	[bazel] add python targets (#101003)	2 years ago
	[BE] Update bazelisk and bazel versions (#140...	7 months ago

.bazelversion		
.clang-format	Enable more C++ warnings (#143355)	6 months ago
.clang-tidy	Enable modernize-use-default-member-init (#...	2 months ago
.cmakelintrc	Fix/relax CMake linter rules (#35574)	5 years ago
.coveragerc	Use JIT Plug-in for coverage to cover JIT'd fun...	4 years ago
.dockerignore	Add .dockerignore. (#3333)	8 years ago
.editorconfig	Add .editorconfig (#149193)	3 months ago
.flake8	Revert "[BE]: Enable RUFF TRY400 rule - log.e...	3 weeks ago
.gdbinit	gdb special command to print tensors (#54339)	4 years ago
.git-blame-ignore-revs	Add ignorable commits on run_test.py to git bl...	5 months ago
.gitattributes	[export] Implement cpp deserializer. (#136398)	7 months ago
.gitignore	[Monitoring] Add util for linux build (#153456)	2 weeks ago
.gitmodules	Revert "Delete TorchScript based Android dem...	3 weeks ago
.lintrunner.toml	[BE][Ez]: Enable ClangFormat aten/src/core/Fo...	3 days ago
.lldbinit	Add helpful pretty pretting summaries to torch ...	2 years ago
BUCK.oss	Revert "Fix clang-tidy warnings in Caffe2 code...	9 months ago
BUILD.bazel	ROCm Sparsity through HipSparseLT (#150578)	3 days ago
CITATION.cff	Fix more URLs (#153277)	3 weeks ago
CMakeLists.txt	Revert "[BE] Cleanup old ExecuTorch codegen ...	8 hours ago
CODEOWNERS	Add torch/header_only_apis.txt and enforce th...	2 weeks ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	5 years ago
CONTRIBUTING.md	Removing conda references from PyTorch Doc...	2 weeks ago
Dockerfile	docker: Use miniforge, install from pip (#13427...	10 months ago
GLOSSARY.md	Add remaining ToCs to ToC lint (#56487)	4 years ago
LICENSE	[ROCm] CK Flash Attention Backend (#143695)	5 months ago
MANIFEST.in	fix citation file in MANIFEST (#89994)	3 years ago
Makefile	[Easy] Add ROCm support to nightly pull tool (...	6 months ago
NOTICE	[submodule] Revert "Adds support for acceler...	7 months ago
README.md	Update README.md - James has the wrong gi...	yesterday
RELEASE.md	[Docs] Mention `version.txt` change for patch ...	2 weeks ago
SECURITY.md	[Security] Advise against loading untrusted Tor...	2 months ago
WORKSPACE	[nativert] port semaphore to c10 util (#153504)	5 days ago
aten.bzl	Remove unnecessary use of ctx.resolve_tools. ...	last year
buckbuild.bzl	[nativert] port semaphore to c10 util (#153504)	5 days ago
build.bzl	[MTIA Aten Backend][2/n] Migrate clamp ops(...	last week
build_variables.bzl	[nativert] Move file_util to pytorch core (#1531...	last week

defs.bzl	[modefile free][long tail] selectify fbcode/caffe...	2 months ago
docker.Makefile	docker: Use miniforge, install from pip (#13427...	10 months ago
mypy-strict.ini	[1/N] Apply py39 ruff fixes (#138578)	7 months ago
mypy.ini	Add support for prototype affine quantization i...	6 months ago
pt_ops.bzl	[lint] Remove unnecessary BUCKRESTRICTED...	11 months ago
pt_template_srcs.bzl	Reify view_func() closures as ViewFuncs (#118...	last year
pyproject.toml	Revert "[BE] Cleanup old ExecuTorch codegen ...	8 hours ago
pyrefly.toml	Add pyrefly.toml (#154144)	last week
pytest.ini	Remove color in CI (#133517)	10 months ago
requirements.txt	[BE]: Use undocumented temp shim to restore...	3 weeks ago
setup.py	Revert "[BE] Cleanup old ExecuTorch codegen ...	8 hours ago
ubsan.supp	Upgrade Pybind submodule to 2.10.4 (#103989)	2 years ago
ufunc_defs.bzl	move build_variables.bzl and ufunc_defs.bzl fr...	3 years ago
version.txt	Update nightly PyTorch version to 2.8.0 (#149...	3 months ago

README Code of conduct License Security

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
  - A GPU-Ready Tensor Library
  - Dynamic Neural Networks: Tape-Based Autograd
  - Python First
  - Imperative Experiences
  - Fast and Lean
  - Extensions Without Pain
- Installation
  - Binaries
    - NVIDIA Jetson Platforms
  - From Source
    - Prerequisites

        - NVIDIA CUDA Support
        - AMD ROCm Support
        - Intel GPU Support
      - Get the PyTorch Source
      - Install Dependencies
      - Install PyTorch
        - Adjust Build Options (Optional)
    - Docker Image
      - Using pre-built images
      - Building the image yourself
    - Building the Documentation
      - Building a PDF
    - Previous Versions
- Getting Started
- Resources
- Communication
- Releases and Contributing
- The Team
- License

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

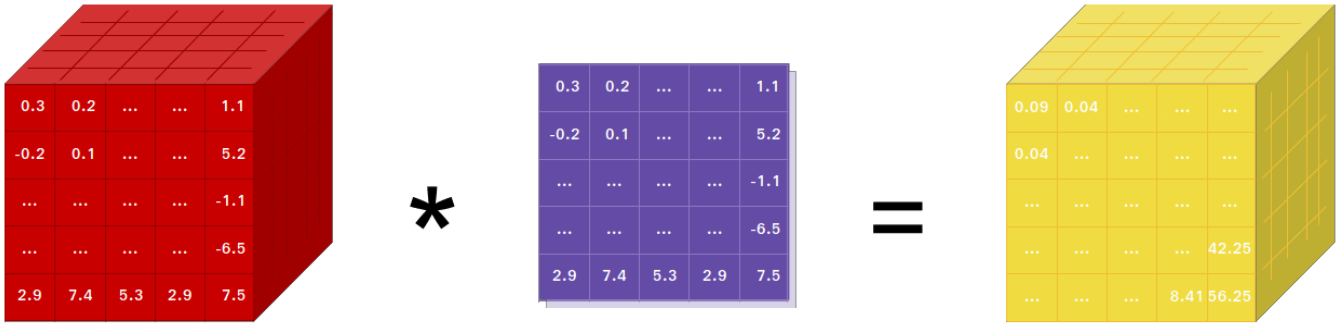
Usually, PyTorch is used either as:

- 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!

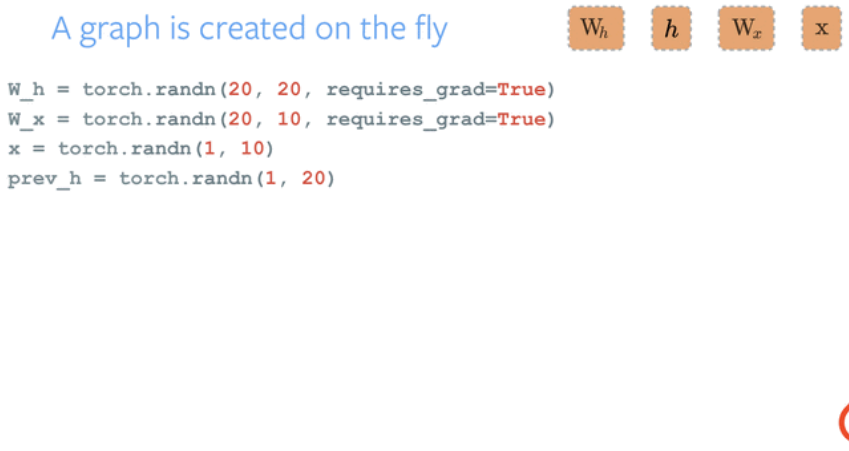
Dynamic Neural Networks: Tape-Based Autograd

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 3.9 or later
- A compiler that fully supports C++17, such as clang or gcc (gcc 9.4.0 or newer is required, on Linux)
- Visual Studio or Visual Studio Build Tool (Windows only)

* 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.

An example of environment setup is shown below:

- Linux:

```
$ source <CONDA_INSTALL_DIR>/bin/activate
$ conda create -y -n <CONDA_NAME>
$ conda activate <CONDA_NAME>
```

- Windows:

```
$ source <CONDA_INSTALL_DIR>\Scripts\activate.bat
$ conda create -y -n <CONDA_NAME>
$ conda activate <CONDA_NAME>
$ call "C:\Program Files\Microsoft Visual Studio\<VERSION>\Community\VC\Auxiliary\Build\vcvarsall.bat" x64
```

NVIDIA CUDA Support

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

AMD ROCm Support

If you want to compile with ROCm support, install

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

By default the build system expects ROCm to be installed in `/opt/rocm`. If ROCm is installed in a different directory, the `ROCM_PATH` environment variable must be set to the ROCm installation directory. The build system automatically detects the AMD GPU architecture. Optionally, the AMD GPU architecture can be explicitly set with the `PYTORCH_ROCM_ARCH` environment variable AMD GPU architecture

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`.

Intel GPU Support

If you want to compile with Intel GPU support, follow these

- PyTorch Prerequisites for Intel GPUs instructions.
- Intel GPU is supported for Linux and Windows.

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

Get the PyTorch Source

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

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“ secti
pip install -r requirements.txt
```

On Linux

```
pip install mkl-static mkl-include
# CUDA only: Add LAPACK support for the GPU if needed
# magma installation: run with active conda environment. specify CUDA version to install
.ci/docker/common/install_magma_conda.sh 12.4

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

```
# For Intel GPU support, please explicitly `export USE_XPU=1` before running command.
make triton
```

On MacOS

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

On Windows

```
pip install mkl-static 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
```

Install PyTorch

On Linux

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 $(which conda))/../'}:${CMAKE_PREFIX_PATH}"
python setup.py develop
```

On macOS

```
python3 setup.py develop
```

On Windows

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.

```
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 run tl
:: "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 [15^

:: [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\HostX64\

python setup.py develop
```

Intel GPU builds

In this mode PyTorch with Intel GPU support will be built.

Please make sure the common prerequisites as well as the prerequisites for Intel GPU are properly installed and the environment variables are configured prior to starting the build. For build tool support, `Visual Studio 2022` is required.

Then PyTorch can be built with the command:

```
:: CMD Commands:
:: Set the CMAKE_PREFIX_PATH to help find corresponding packages
:: %CONDA_PREFIX% only works after `conda activate custom_env`

if defined CMAKE_PREFIX_PATH (
    set "CMAKE_PREFIX_PATH=%CONDA_PREFIX%\Library;%CMAKE_PREFIX_PATH%"
) else (
    set "CMAKE_PREFIX_PATH=%CONDA_PREFIX%\Library"
)

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

On macOS

```
export CMAKE_PREFIX_PATH="${CONDA_PREFIX:-'$(dirname $(which conda))/../'}:${CMAKE_PREFIX_PATH}"
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
```

Please note that PyTorch uses shared memory to share data between processes, so if torch multiprocessing is used (e.g. for multithreaded 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.

```
make -f docker.Makefile
```

Building the Documentation

To build documentation in various formats, you will need Sphinx and the pytorch_sphinx_theme2.

Before you build the documentation locally, ensure `torch` is installed in your environment. For small fixes, you can install the nightly version as described in Getting Started.

For more complex fixes, such as adding a new module and docstrings for the new module, you might need to install torch from source. See Docstring Guidelines for docstring conventions.

```
cd docs/
pip install -r requirements.txt
make html
make serve
```

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 `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`

> **Note**
>
> If you see a numpy incompatibility error, run:
>
> ```
> pip install 'numpy<2'
> ```

When you make changes to the dependencies run by CI, edit the `.ci/docker/requirements-docs.txt` file.

Building a PDF

To compile a PDF of all PyTorch documentation, ensure you have `texlive` and LaTeX installed. On macOS, you can install them using:

```
brew install --cask mactex
```

To create the PDF:

1. Run:

   ```
   make latexpdf
   ```

   This will generate the necessary files in the `build/latex` directory.

2. Navigate to this directory and execute:

   ```
   make LATEXOPTS="-interaction=nonstopmode"
   ```

   This will produce a `pytorch.pdf` with the desired content. Run this command one more time so that it generates the correct table of contents and index.

> **Note**
>
> To view the Table of Contents, switch to the **Table of Contents** view in your PDF viewer.

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

Releases 59

PyTorch 2.7.0 Release Latest
on Apr 23

+ 58 releases

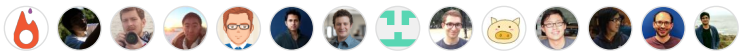
Packages

No packages published

Used by 780k

+ 780,204

Contributors 3,847



+ 3,833 contributors

Languages

Python 57.7% C++ 34.1% Cuda 3.0% C 1.5% Objective-C++ 1.1% CMake 0.7% Other 1.9%