Source:

https://pytorch.org/docs/stable/torch.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch

Shortcuts

torch

The torch package contains data structures for multi-dimensional tensors and defines mathematical operations over these tensors. Additionally, it provides many utilities for efficient serialization of Tensors and arbitrary types, and other useful utilities.

It has a CUDA counterpart, that enables you to run your tensor computations on an NVIDIA GPU with compute capability >= 3.0.

Tensor

Source:

https://pytorch.org/docs/stable/nn.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch.nn

Shortcuts

torch.nn

These are the basic building blocks for graphs:

torch.nn

Containers

Convolution Layers

Pooling layers

Padding Layers

Non-linear Activations (weighted sum, nonlinearity)

Non-linear Activations (other)

Normalization Layers

Recurrent Layers

Transformer Layers

Linear Layers

Dropout Layers

Sparse Layers

Distance Functions

Loss Functions

Vision Layers

Shuffle Layers

Data

Source:

https://pytorch.org/docs/stable/nn.functional.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch.nn.functional

Shortcuts

torch.nn.functional

Convolution functions

conv1d

Applies a 1D convolution over an input signal composed of several input planes.

conv2d

Applies a 2D convolution over an input image composed of several input planes.

conv3d

Applies a 3D convolution over an input image composed of several input planes.

conv_transpose1d

Applies a 1D transposed convolution o

Source:

https://pytorch.org/docs/stable/tensors.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch.Tensor

Shortcuts

torch.Tensor

A torch.Tensor is a multi-dimensional matrix containing elements of a single data type.

Data types

Torch defines tensor types with the following data types:

Data type

dtype

32-bit floating point

torch.float32 or torch.float

64-bit floating point

torch.float64 or torch.double

16-bit floating point 1

torch.float16 or torch.half

16-bit floatin

Source:

https://pytorch.org/docs/stable/tensor_attributes.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Tensor Attributes

Shortcuts

Tensor Attributes

Each torch.Tensor has a torch.dtype, torch.device, and torch.layout.

torch.dtype

CLASS

torch.dtype

A torch.dtype is an object that represents the data type of a torch.Tensor. PyTorch has twelve different data types:

Data type

dtype

Legacy Constructors

32-bit floating point

torch.float32 or torch.float

torch.*.FloatTensor

64-bit floating p

Source:

https://pytorch.org/docs/stable/tensor_view.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Tensor Views

Tensor Views

PyTorch allows a tensor to be a View of an existing tensor. View tensor shares the same underlying data with its base tensor. Supporting View avoids explicit data copy, thus allows us to do fast and memory efficient reshaping, slicing and element-wise operations.

For example, to get a view of an existing tensor t, you can call t.view(...).

```
>>> t = torch.rand(4, 4)
>>> b = t.view(2,
```

Source:

https://pytorch.org/docs/stable/amp.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Automatic Mixed Precision package - torch.amp

Shortcuts

Automatic Mixed Precision package - torch.amp

torch.amp provides convenience methods for mixed precision, where some operations use the torch.float32 (float) datatype and other operations use lower precision floating point datatype (lower_precision_fp): torch.float16 (half) or torch.bfloat16. Some ops, like linear layers and convolutions, are much faster

Source:

https://pytorch.org/docs/stable/autograd.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Automatic differentiation package - torch.autograd

Shortcuts

Automatic differentiation package - torch.autograd

torch.autograd provides classes and functions implementing automatic differentiation of arbitrary scalar valued functions.

It requires minimal changes to the existing code - you only need to declare Tensor s for which gradients should be computed with the requires_grad=True keyword. As of now, we on

Source:

https://pytorch.org/docs/stable/library.html

Landing Page for more details on how to effectively use these APIs.

Testing custom ops

Use torch.library.opcheck() to test custom ops for incorrect usage of the Python torch.library and/or C++ TORCH_LIBRARY APIs. Also, if your operator supports training, use torch.autograd.gradcheck() to test that the gradients are mathematically correct.

torch.library.opcheck(op, args, kwargs=None, *, test_utils=('test_schema', 'test_autograd_registration', 'test_faketensor', 'test_aot_dispatch_dynamic'), raise_exception=True) [SOURCE]

Given an operator and some sample arguments, tests if the operator is registered correctly.

That is, when you use the torch.library/TORCH_LIBRARY APIs to create a custom op, you specified metadata (e.g. mutability info) about the custom op and these APIs require that the functions you pass them satisfy certain properties (e.g. no data pointer access in the fake/meta/abstract kernel) opcheck tests these metadata and properties.

Concretely, we test the following:

t

Source:

https://pytorch.org/docs/stable/cpu.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch.cpu

Shortcuts

torch.cpu

This package implements abstractions found in torch.cuda to facilitate writing device-agnostic code.

current_device

Returns current device for cpu.

current_stream

Returns the currently selected Stream for a given device.

is_available

Returns a bool indicating if CPU is currently available.

synchronize

Waits for all kernels in all streams on the CPU dev

Source:

https://pytorch.org/docs/stable/cuda.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch.cuda

Shortcuts

torch.cuda

This package adds support for CUDA tensor types.

It implements the same function as CPU tensors, but they utilize GPUs for computation.

It is lazily initialized, so you can always import it, and use is_available() to determine if your system supports CUDA.

CUDA semantics has more details about working with CUDA.

StreamContext

Context-manager that selects a given stream.

can_device_access_peer

Check if peer access between two devices is possible.

current_blas_handle

Return cublasHandle_t pointer to current cuBLAS handle

current_device

Return the index of a currently selected device.

current_stream

Return the currently selected Stream for a given device.

cudart

Retrieves the CUDA runtime API module.

default_stream

Return the default Stream for a given device.

device

Context-manager that changes the selected device.

device_count

Return the number of GPUs available.

device_of

Context-manager that changes the current device to that of given object.

get_arch_list

Return list CUDA architectures this library was compiled for.

get_device_capability

Get the cuda capability of a device.

get_device_name

Get the name of a device.

get_device_properties

Get the properties of a device.

get_gencode_flags

Return NVCC gencode flags this library was compiled wi

Source:

https://pytorch.org/docs/stable/torch_cuda_memory.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Understanding CUDA Memory Usage

Shortcuts

Understanding CUDA Memory Usage

To debug CUDA memory use, PyTorch provides a way to generate memory snapshots that record the state of allocated CUDA memory at any point in time, and optionally record the history of allocation events that led up to that snapshot.

The generated snapshots can then be drag and dropped onto the interactiver viewer hosted at pytorch.org/me

Source:

https://pytorch.org/docs/stable/torch_cuda_memory.html#generating-a-snapshot

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Understanding CUDA Memory Usage

Shortcuts

Understanding CUDA Memory Usage

To debug CUDA memory use, PyTorch provides a way to generate memory snapshots that record the state of allocated CUDA memory at any point in time, and optionally record the history of allocation events that led up to that snapshot.

The generated snapshots can then be drag and dropped onto the interactiver viewer hosted at pytorch.org/me

Source:

https://pytorch.org/docs/stable/torch_cuda_memory.html#using-the-visualizer

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Understanding CUDA Memory Usage

Shortcuts

Understanding CUDA Memory Usage

To debug CUDA memory use, PyTorch provides a way to generate memory snapshots that record the state of allocated CUDA memory at any point in time, and optionally record the history of allocation events that led up to that snapshot.

The generated snapshots can then be drag and dropped onto the interactiver viewer hosted at pytorch.org/me

Source:

https://pytorch.org/docs/stable/torch_cuda_memory.html#snapshot-api-reference

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Understanding CUDA Memory Usage

Shortcuts

Understanding CUDA Memory Usage

To debug CUDA memory use, PyTorch provides a way to generate memory snapshots that record the state of allocated CUDA memory at any point in time, and optionally record the history of allocation events that led up to that snapshot.

The generated snapshots can then be drag and dropped onto the interactiver viewer hosted at pytorch.org/me

Source:

https://pytorch.org/docs/stable/export.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > torch.export

Shortcuts

torch.export

WARNING

This feature is a prototype under active development and there WILL BE BREAKING CHANGES in the future.

Overview

torch.export.export() takes an arbitrary Python callable (a torch.nn.Module, a function or a method) and produces a traced graph representing only the Tensor computation of the function in an Ahead-of-Time (AOT) fashion, which can subsequently be execute

Source:

https://pytorch.org/docs/stable/distributed.html

---

cation Hooks

Quantization

Distributed RPC Framework

torch.random

torch.masked

torch.nested

torch.Size

torch.sparse

torch.Storage

torch.testing

torch.utils

torch.utils.benchmark

torch.utils.bottleneck

torch.utils.checkpoint

torch.utils.cpp_extension

torch.utils.data

torch.utils.deterministic

torch.utils.jit

torch.utils.dlpack

torch.utils.mobile_optimizer

torch.utils.model_zoo

torch.utils.tensorboard

torch.utils.module_tracker

Type Info

Named Tensors

Named Tensors operator coverage

torch.__config__

torch.__future__

torch._logging

Torch Environment Variables

Libraries[+]

Docs > Distributed communication package - torch.distributed

Shortcuts

Distributed communication package - torch.distributed

NOTE

Please refer to PyTorch Distributed Overview for a brief introduction to all features related to distributed training.

Backends

torch.distributed supports three built-in backends, each with different capabilities. The table below shows which functions are available for use with CPU / CU