torch.Tensor

A `torch.Tensor` is a multi- imensional matrix containing elements of a single ata type.

Data ty es

Torch efines tensor types with the following ata types:

Data type	type
32- it floating point	`torch.float32` or `torch.float`
64- it floating point	`torch.float64` or `torch.double`
16- it floating point 1	`torch.float16` or `torch.half`
16- it floating point 2	`torch.bfloat16`
32- it complex	`torch.complex32` or `torch.chalf`
64- it complex	`torch.complex64` or `torch.cfloat`
128- it complex	`torch.complex128` or `torch.cdouble`
8- it integer (unsigne)	`torch.uint8`
16- it integer (unsigne)	`torch.uint16` (limite support) 4
32- it integer (unsigne)	`torch.uint32` (limite support) 4
64- it integer (unsigne)	`torch.uint64` (limite support) 4
8- it integer (signe)	`torch.int8`
16- it integer (signe)	`torch.int16` or `torch.short`
32- it integer (signe)	`torch.int32` or `torch.int`
64- it integer (signe)	`torch.int64` or `torch.long`
Boolean	`torch.bool`
quantize 8- it integer (unsigne)	`torch.quint8`
quantize 8- it integer (signe)	`torch.qint8`
quantize 32- it integer (signe)	`torch.qint32`
quantize 4- it integer (unsigne) 3	`torch.quint4x2`
8- it floating point, e4m3 5	`torch.float8_e4m3fn` (limite support)
8- it floating point, e5m2 5	`torch.float8_e5m2` (limite support)

1

Sometimes referre to as inary16: uses 1 sign, 5 exponent, an 10 significan its. seful when precision is important at the expense of range.

2

Sometimes referre to as Brain Floating Point: uses 1 sign, 8 exponent, an 7 significan its. seful when range is important, since it has the same num er of exponent its as `float32`

3

quantize 4- it integer is store as a 8- it signe integer. Currently it's only supporte in Em e ingBag operator.

4(1,2,3)

nsigne types asi es from `uint8` are currently planne to only have limite support in eager mo e (they primarily exist to assist usage with torch.compile); if you nee eager support an the extra range is not nee e , we recommen using their signe variants instea . See https://githu .com/pytorch/pytorch/issues/58734 for more etails.

5(1,2)

`torch.float8_e4m3fn` an `torch.float8_e5m2` implement the spec for 8- it floating point types from https://arxiv.org/a s/2209.05433. The op support is very limite .

For ackwar s compati ility, we support the following alternate class names for these ata types:

Data type	CP tensor	GP tensor

64- it floating point	`torch.DoubleTensor`	`torch.cuda.DoubleTensor`
16- it floating point	`torch.HalfTensor`	`torch.cuda.HalfTensor`
16- it floating point	`torch.BFloat16Tensor`	`torch.cuda.BFloat16Tensor`
8- it integer (unsigne)	`torch.ByteTensor`	`torch.cuda.ByteTensor`
8- it integer (signe)	`torch.CharTensor`	`torch.cuda.CharTensor`
16- it integer (signe)	`torch.ShortTensor`	`torch.cuda.ShortTensor`
32- it integer (signe)	`torch.IntTensor`	`torch.cuda.IntTensor`
64- it integer (signe)	`torch.LongTensor`	`torch.cuda.LongTensor`
Boolean	`torch.BoolTensor`	`torch.cuda.BoolTensor`

However, to construct tensors, we recommen using factory functions such as `torch.empty()` with the `dtype` argument instea . The `torch.Tensor` constructor is an alias for the efault tensor type (`torch.FloatTensor`).

Initializing an asic o erations

A tensor can e constructe from a Python `list` or sequence using the `torch.tensor()` constructor:

```
>>> torch.tensor([[1., -1.], [1., -1.]])
tensor([[ 1.0000, -1.0000],
        [ 1.0000, -1.0000]])
>>> torch.tensor(np.array([[1, 2, 3], [4, 5, 6]]))
tensor([[ 1,  2,  3],
        [ 4,  5,  6]])
```

• WARNING

`torch.tensor()` always copies `data`. If you have a Tensor `data` an ust want to change its `requires_grad` flag, use `requires_grad_()` or `detach()` to avoi a copy. If you have a numpy array an want to avoi a copy, use `torch.as_tensor()`.

A tensor of specific ata type can e constructe y passing a `torch.dtype` an /or a `torch.device` to a constructor or tensor creation op:

```
>>> torch.zeros([2, 4], dtype=torch.int32)
tensor([[ 0,  0,  0,  0],
        [ 0,  0,  0,  0]], dtype=torch.int32)
>>> cuda0 = torch.device('cuda:0')
>>> torch.ones([2, 4], dtype=torch.float64, device=cuda0)
tensor([[ 1.0000,  1.0000,  1.0000,  1.0000],
        [ 1.0000,  1.0000,  1.0000,  1.0000]], dtype=torch.float64, device='cuda:0')
```

For more information a out uil ing Tensors, see Creation Ops

The contents of a tensor can e accesse an mo ifie using Python's in exing an slicing notation:

```
>>> x = torch.tensor([[1, 2, 3], [4, 5, 6]])
>>> print(x[1][2])
tensor(6)
>>> x[0][1] = 8
>>> print(x)
tensor([[ 1,  8,  3],
        [ 4,  5,  6]])
```

se `torch.Tensor.item()` to get a Python num er from a tensor containing a single value:

```
>>> x = torch.tensor([[1]])
>>> x
tensor([[ 1]])
>>> x.item()
1
>>> x = torch.tensor(2.5)
>>> x
tensor(2.5000)
>>> x.item()
2.5
```

For more information a out in exing, see In exing, Slicing, Joining, Mutating Ops

A tensor can e create with `requires_grad=True` so that `torch.autograd` recor s operations on them for automatic ifferentiation.

```
>>> x = torch.tensor([[1., -1.], [1., 1.]], requires_grad=True)
>>> out = x.pow(2).sum()
>>> out.backward()
>>> x.grad
tensor([[ 2.0000, -2.0000],
        [ 2.0000,  2.0000]])
```

Each tensor has an associate `torch.Storage`, which hol s its ata. The tensor class also provi es multi- imensional, stri e view of a storage an efines numeric operations on it.

• NOTE

For more information on tensor views, see Tensor Views.

• NOTE

For more information on the `torch.dtype`, `torch.device`, an `torch.layout` attri utes of a `torch.Tensor`, see Tensor Attri utes.

• NOTE

• NOTE

To change an existing tensor's `torch.device` an /or `torch.dtype`, consi er using `to()` metho on the tensor.

• WARNING

Current implementation of `torch.Tensor` intro uces memory overhea , thus it might lea to unexpecte ly high memory usage in the applications with many tiny tensors. If this is your case, consi er using one large structure.

Tensor class reference

CLASS torch.Tensor

There are a few main ways to create a tensor, epen ing on your use case.

- To create a tensor with pre-existing ata, use `torch.tensor()`.
- To create a tensor with specific size, use `torch.*` tensor creation ops (see Creation Ops).
- To create a tensor with the same size (an similar types) as another tensor, use `torch.*_like` tensor creation ops (see Creation Ops).
- To create a tensor with similar type ut ifferent size as another tensor, use `tensor.new_*` creation ops.
- There is a legacy constructor `torch.Tensor` whose use is iscourage . se `torch.tensor()` instea .

Tensor.__init__(*self*, *data*)

This constructor is eprecate , we recommen using `torch.tensor()` instea . What this constructor oes epen s on the type of `data`.

- If `data` is a Tensor, returns an alias to the original Tensor. nlike `torch.tensor()`, this tracks autogra an will propagate gra ients to the original Tensor. `device` kwarg is not supporte for this `data` type.
- If `data` is a sequence or neste sequence, create a tensor of the efault type (typically `torch.float32`) whose ata is the values in the sequences, performing coercions if necessary. Nota ly, this iffers from `torch.tensor()` in that this constructor will always construct a float tensor, even if the inputs are all integers.
- If `data` is a `torch.Size`, returns an empty tensor of that size.

This constructor oes not support explicitly specifying `dtype` or `device` of the returne tensor. We recommen using `torch.tensor()` which provi es this functionality.

Args:

ata (array_like): The tensor to construct from.

Keywor args:

evice (`torch.device`, optional): the esire evice of returne tensor.
Default: if None, same `torch.device` as this tensor.

Tensor.T

Returns a view of this tensor with its imensions reverse .

If `n` is the num er of imensions in `x`, `x.T` is equivalent to `x.permute(n-1, n-2, ..., 0)`.

• WARNING

The use of `Tensor.T()` on tensors of imension other than 2 to reverse their shape is eprecate an it will throw an error in a future release. Consi er `mT` to transpose atches of matrices or *x.permute(*torch.arange(x.ndim - 1, -1, -1))* to reverse the imensions of a tensor.

Tensor.H

Returns a view of a matrix (2-D tensor) con ugate an transpose .

`x.H` is equivalent to `x.transpose(0, 1).conj()` for complex matrices an `x.transpose(0, 1)` for real matrices.

• SEE ALSO

`mH`: An attri ute that also works on atches of matrices.

Tensor.mT

Returns a view of this tensor with the last two imensions transpose .

`x.mT` is equivalent to `x.transpose(-2, -1)`.

Tensor.mH

Accessing this property is equivalent to calling `adjoint()`.

`Tensor.new_tensor`	Returns a new Tensor with `data` as the tensor ata.
`Tensor.new_full`	Returns a Tensor of size `size` fille with `fill_value`.
`Tensor.new_empty`	Returns a Tensor of size `size` fille with uninitialize ata.
`Tensor.new_ones`	Returns a Tensor of size `size` fille with `1`.
`Tensor.new_zeros`	Returns a Tensor of size `size` fille with `0`.
`Tensor.is_cuda`	Is `True` if the Tensor is store on the GP , `False` otherwise.
`Tensor.is_quantized`	Is `True` if the Tensor is quantize , `False` otherwise.
`Tensor.is_meta`	Is `True` if the Tensor is a meta tensor, `False` otherwise.
`Tensor.device`	Is the `torch.device` where this Tensor is.
`Tensor.grad`	This attri ute is `None` y efault an ecomes a Tensor the first time a call to `backward()` computes gra ients for `self`.

`Tensor.real`	Returns a new tensor containing real values of the `self` tensor for a complex-value input tensor.
`Tensor.imag`	Returns a new tensor containing imaginary values of the `self` tensor.
`Tensor.nbytes`	Returns the number of bytes consumed by the "view" of elements of the Tensor if the Tensor does not use sparse storage layout.
`Tensor.itemsize`	Alias for `element_size()`
`Tensor.abs`	See `torch.abs()`
`Tensor.abs_`	In-place version of `abs()`
`Tensor.absolute`	Alias for `abs()`
`Tensor.absolute_`	In-place version of `absolute()` Alias for `abs_()`
`Tensor.acos`	See `torch.acos()`
`Tensor.acos_`	In-place version of `acos()`
`Tensor.arccos`	See `torch.arccos()`
`Tensor.arccos_`	In-place version of `arccos()`
`Tensor.add`	A a scalar or tensor to `self` tensor.
`Tensor.add_`	In-place version of `add()`
`Tensor.addbmm`	See `torch.addbmm()`
`Tensor.addbmm_`	In-place version of `addbmm()`
`Tensor.addcdiv`	See `torch.addcdiv()`
`Tensor.addcdiv_`	In-place version of `addcdiv()`
`Tensor.addcmul`	See `torch.addcmul()`
`Tensor.addcmul_`	In-place version of `addcmul()`
`Tensor.addmm`	See `torch.addmm()`
`Tensor.addmm_`	In-place version of `addmm()`
`Tensor.sspaddmm`	See `torch.sspaddmm()`
`Tensor.addmv`	See `torch.addmv()`
`Tensor.addmv_`	In-place version of `addmv()`
`Tensor.addr`	See `torch.addr()`
`Tensor.addr_`	In-place version of `addr()`
`Tensor.adjoint`	Alias for `adjoint()`
`Tensor.allclose`	See `torch.allclose()`
`Tensor.amax`	See `torch.amax()`
`Tensor.amin`	See `torch.amin()`
`Tensor.aminmax`	See `torch.aminmax()`
`Tensor.angle`	See `torch.angle()`
`Tensor.apply_`	Applies the function `callable` to each element in the tensor, replacing each element with the value returned by `callable`.
`Tensor.argmax`	See `torch.argmax()`
`Tensor.argmin`	See `torch.argmin()`

`Tensor.asin`	See `torch.asin()`
`Tensor.asin_`	In-place version of `asin()`
`Tensor.arcsin`	See `torch.arcsin()`
`Tensor.arcsin_`	In-place version of `arcsin()`
`Tensor.as_strided`	See `torch.as_strided()`
`Tensor.atan`	See `torch.atan()`
`Tensor.atan_`	In-place version of `atan()`
`Tensor.arctan`	See `torch.arctan()`
`Tensor.arctan_`	In-place version of `arctan()`
`Tensor.atan2`	See `torch.atan2()`
`Tensor.atan2_`	In-place version of `atan2()`
`Tensor.arctan2`	See `torch.arctan2()`
`Tensor.arctan2_`	atan2_(other) -> Tensor
`Tensor.all`	See `torch.all()`
`Tensor.any`	See `torch.any()`
`Tensor.backward`	Computes the gra ient of current tensor wrt graph leaves.
`Tensor.baddbmm`	See `torch.baddbmm()`
`Tensor.baddbmm_`	In-place version of `baddbmm()`
`Tensor.bernoulli`	Returns a result tensor where each $\texttt{result[i]}$ is in epen ently sample from $\text{Bernoulli}(\texttt{self[i]})$.
`Tensor.bernoulli_`	Fills each location of `self` with an in epen ent sample from $\text{Bernoulli}(\texttt{p})$.
`Tensor.bfloat16`	`self.bfloat16()` is equivalent to `self.to(torch.bfloat16)`.
`Tensor.bincount`	See `torch.bincount()`
`Tensor.bitwise_not`	See `torch.bitwise_not()`
`Tensor.bitwise_not_`	In-place version of `bitwise_not()`
`Tensor.bitwise_and`	See `torch.bitwise_and()`
`Tensor.bitwise_and_`	In-place version of `bitwise_and()`
`Tensor.bitwise_or`	See `torch.bitwise_or()`
`Tensor.bitwise_or_`	In-place version of `bitwise_or()`
`Tensor.bitwise_xor`	See `torch.bitwise_xor()`
`Tensor.bitwise_xor_`	In-place version of `bitwise_xor()`
`Tensor.bitwise_left_shift`	See `torch.bitwise_left_shift()`
`Tensor.bitwise_left_shift_`	In-place version of `bitwise_left_shift()`
`Tensor.bitwise_right_shift`	See `torch.bitwise_right_shift()`
`Tensor.bitwise_right_shift_`	In-place version of `bitwise_right_shift()`
`Tensor.bmm`	See `torch.bmm()`
`Tensor.bool`	`self.bool()` is equivalent to `self.to(torch.bool)`.
`Tensor.byte`	`self.byte()` is equivalent to `self.to(torch.uint8)`.

`Tensor.cauchy_`	Fills the tensor with numbers drawn from the Cauchy distribution:
`Tensor.ceil`	See `torch.ceil()`
`Tensor.ceil_`	In-place version of `ceil()`
`Tensor.char`	`self.char()` is equivalent to `self.to(torch.int8)`.
`Tensor.cholesky`	See `torch.cholesky()`
`Tensor.cholesky_inverse`	See `torch.cholesky_inverse()`
`Tensor.cholesky_solve`	See `torch.cholesky_solve()`
`Tensor.chunk`	See `torch.chunk()`
`Tensor.clamp`	See `torch.clamp()`
`Tensor.clamp_`	In-place version of `clamp()`
`Tensor.clip`	Alias for `clamp()`.
`Tensor.clip_`	Alias for `clamp_()`.
`Tensor.clone`	See `torch.clone()`
`Tensor.contiguous`	Returns a contiguous in memory tensor containing the same data as `self` tensor.
`Tensor.copy_`	Copies the elements from `src` into `self` tensor and returns `self`.
`Tensor.conj`	See `torch.conj()`
`Tensor.conj_physical`	See `torch.conj_physical()`
`Tensor.conj_physical_`	In-place version of `conj_physical()`
`Tensor.resolve_conj`	See `torch.resolve_conj()`
`Tensor.resolve_neg`	See `torch.resolve_neg()`
`Tensor.copysign`	See `torch.copysign()`
`Tensor.copysign_`	In-place version of `copysign()`
`Tensor.cos`	See `torch.cos()`
`Tensor.cos_`	In-place version of `cos()`
`Tensor.cosh`	See `torch.cosh()`
`Tensor.cosh_`	In-place version of `cosh()`
`Tensor.corrcoef`	See `torch.corrcoef()`
`Tensor.count_nonzero`	See `torch.count_nonzero()`
`Tensor.cov`	See `torch.cov()`
`Tensor.acosh`	See `torch.acosh()`
`Tensor.acosh_`	In-place version of `acosh()`
`Tensor.arccosh`	acosh() -> Tensor
`Tensor.arccosh_`	acosh_() -> Tensor
`Tensor.cpu`	Returns a copy of this object in CPU memory.
`Tensor.cross`	See `torch.cross()`
`Tensor.cuda`	Returns a copy of this object in CUDA memory.
`Tensor.logcumsumexp`	See `torch.logcumsumexp()`

`Tensor.cummin`	See `torch.cummin()`
`Tensor.cumprod`	See `torch.cumprod()`
`Tensor.cumprod_`	In-place version of `cumprod()`
`Tensor.cumsum`	See `torch.cumsum()`
`Tensor.cumsum_`	In-place version of `cumsum()`
`Tensor.chalf`	`self.chalf()` is equivalent to `self.to(torch.complex32)`.
`Tensor.cfloat`	`self.cfloat()` is equivalent to `self.to(torch.complex64)`.
`Tensor.cdouble`	`self.cdouble()` is equivalent to `self.to(torch.complex128)`.
`Tensor.data_ptr`	Returns the a ress of the first element of `self` tensor.
`Tensor.deg2rad`	See `torch.deg2rad()`
`Tensor.dequantize`	Given a quantize Tensor, equantize it an return the equantize float Tensor.
`Tensor.det`	See `torch.det()`
`Tensor.dense_dim`	Return the num er of ense imensions in a sparse tensor `self`.
`Tensor.detach`	Returns a new Tensor, etache from the current graph.
`Tensor.detach_`	Detaches the Tensor from the graph that create it, making it a leaf.
`Tensor.diag`	See `torch.diag()`
`Tensor.diag_embed`	See `torch.diag_embed()`
`Tensor.diagflat`	See `torch.diagflat()`
`Tensor.diagonal`	See `torch.diagonal()`
`Tensor.diagonal_scatter`	See `torch.diagonal_scatter()`
`Tensor.fill_diagonal_`	Fill the main iagonal of a tensor that has at least 2- imensions.
`Tensor.fmax`	See `torch.fmax()`
`Tensor.fmin`	See `torch.fmin()`
`Tensor.diff`	See `torch.diff()`
`Tensor.digamma`	See `torch.digamma()`
`Tensor.digamma_`	In-place version of `digamma()`
`Tensor.dim`	Returns the num er of imensions of `self` tensor.
`Tensor.dim_order`	Returns the uniquely etermine tuple of int escri ing the im or er or physical layout of `self`.
`Tensor.dist`	See `torch.dist()`
`Tensor.div`	See `torch.div()`
`Tensor.div_`	In-place version of `div()`
`Tensor.divide`	See `torch.divide()`
`Tensor.divide_`	In-place version of `divide()`
`Tensor.dot`	See `torch.dot()`
`Tensor.double`	`self.double()` is equivalent to `self.to(torch.float64)`.
`Tensor.dsplit`	See `torch.dsplit()`

`Tensor.eq_`	In-place version of `eq()`
`Tensor.equal`	See `torch.equal()`
`Tensor.erf`	See `torch.erf()`
`Tensor.erf_`	In-place version of `erf()`
`Tensor.erfc`	See `torch.erfc()`
`Tensor.erfc_`	In-place version of `erfc()`
`Tensor.erfinv`	See `torch.erfinv()`
`Tensor.erfinv_`	In-place version of `erfinv()`
`Tensor.exp`	See `torch.exp()`
`Tensor.exp_`	In-place version of `exp()`
`Tensor.expm1`	See `torch.expm1()`
`Tensor.expm1_`	In-place version of `expm1()`
`Tensor.expand`	Returns a new view of the `self` tensor with singleton imensions expan e to a larger size.
`Tensor.expand_as`	Expan this tensor to the same size as `other`.
`Tensor.exponential_`	Fills `self` tensor with elements rawn from the PDF (pro a ility ensity function):
`Tensor.fix`	See `torch.fix()`.
`Tensor.fix_`	In-place version of `fix()`
`Tensor.fill_`	Fills `self` tensor with the specifie value.
`Tensor.flatten`	See `torch.flatten()`
`Tensor.flip`	See `torch.flip()`
`Tensor.fliplr`	See `torch.fliplr()`
`Tensor.flipud`	See `torch.flipud()`
`Tensor.float`	`self.float()` is equivalent to `self.to(torch.float32)`.
`Tensor.float_power`	See `torch.float_power()`
`Tensor.float_power_`	In-place version of `float_power()`
`Tensor.floor`	See `torch.floor()`
`Tensor.floor_`	In-place version of `floor()`
`Tensor.floor_divide`	See `torch.floor_divide()`
`Tensor.floor_divide_`	In-place version of `floor_divide()`
`Tensor.fmod`	See `torch.fmod()`
`Tensor.fmod_`	In-place version of `fmod()`
`Tensor.frac`	See `torch.frac()`
`Tensor.frac_`	In-place version of `frac()`
`Tensor.frexp`	See `torch.frexp()`
`Tensor.gather`	See `torch.gather()`
`Tensor.gcd`	See `torch.gcd()`
`Tensor.gcd_`	In-place version of `gcd()`

`Tensor.ge_`	In-place version of `ge()`.
`Tensor.greater_equal`	See `torch.greater_equal()`.
`Tensor.greater_equal_`	In-place version of `greater_equal()`.
`Tensor.geometric_`	Fills `self` tensor with elements rawn from the geometric istri ution:
`Tensor.geqrf`	See `torch.geqrf()`
`Tensor.ger`	See `torch.ger()`
`Tensor.get_device`	For C DA tensors, this function returns the evice or inal of the GP on which the tensor resi es.
`Tensor.gt`	See `torch.gt()`.
`Tensor.gt_`	In-place version of `gt()`.
`Tensor.greater`	See `torch.greater()`.
`Tensor.greater_`	In-place version of `greater()`.
`Tensor.half`	`self.half()` is equivalent to `self.to(torch.float16)`.
`Tensor.hardshrink`	See `torch.nn.functional.hardshrink()`
`Tensor.heaviside`	See `torch.heaviside()`
`Tensor.histc`	See `torch.histc()`
`Tensor.histogram`	See `torch.histogram()`
`Tensor.hsplit`	See `torch.hsplit()`
`Tensor.hypot`	See `torch.hypot()`
`Tensor.hypot_`	In-place version of `hypot()`
`Tensor.i0`	See `torch.i0()`
`Tensor.i0_`	In-place version of `i0()`
`Tensor.igamma`	See `torch.igamma()`
`Tensor.igamma_`	In-place version of `igamma()`
`Tensor.igammac`	See `torch.igammac()`
`Tensor.igammac_`	In-place version of `igammac()`
`Tensor.index_add_`	Accumulate the elements of `alpha` times `source` into the `self` tensor y a ing to the in ices in the or er given in `index`.
`Tensor.index_add`	Out-of-place version of `torch.Tensor.index_add_()`.
`Tensor.index_copy_`	Copies the elements of `tensor` into the `self` tensor y selecting the in ices in the or er given in `index`.
`Tensor.index_copy`	Out-of-place version of `torch.Tensor.index_copy_()`.
`Tensor.index_fill`	Fills the elements of the `self` tensor with value `value` y selecting the in ices in the or er given in `index`.
`Tensor.index_fill`	Out-of-place version of `torch.Tensor.index_fill_()`.
`Tensor.index_put_`	Puts values from the tensor `values` into the tensor `self` using the in ices specifie in `indices` (which is a tuple of Tensors).
`Tensor.index_put`	Out-place version of `index_put_()`.
`Tensor.index_reduce_`	Accumulate the elements of `source` into the `self` tensor y accumulating to the in ices in the or er given in `index` using the re uction given y the `reduce` argument.
`Tensor.index_reduce`	

`Tensor.indices`	Return the in ices tensor of a sparse COO tensor.
`Tensor.inner`	See `torch.inner()`.
`Tensor.int`	`self.int()` is equivalent to `self.to(torch.int32)`.
`Tensor.int_repr`	Given a quantize Tensor, `self.int_repr()` returns a CP Tensor with uint8_t as ata type that stores the un erlying uint8_t values of the given Tensor.
`Tensor.inverse`	See `torch.inverse()`
`Tensor.isclose`	See `torch.isclose()`
`Tensor.isfinite`	See `torch.isfinite()`
`Tensor.isinf`	See `torch.isinf()`
`Tensor.isposinf`	See `torch.isposinf()`
`Tensor.isneginf`	See `torch.isneginf()`
`Tensor.isnan`	See `torch.isnan()`
`Tensor.is_contiguous`	Returns True if `self` tensor is contiguous in memory in the or er specifie y memory format.
`Tensor.is_complex`	Returns True if the ata type of `self` is a complex ata type.
`Tensor.is_conj`	Returns True if the con ugate it of `self` is set to true.
`Tensor.is_floating_point`	Returns True if the ata type of `self` is a floating point ata type.
`Tensor.is_inference`	See `torch.is_inference()`
`Tensor.is_leaf`	All Tensors that have `requires_grad` which is `False` will e leaf Tensors y convention.
`Tensor.is_pinned`	Returns true if this tensor resi es in pinne memory.
`Tensor.is_set_to`	Returns True if oth tensors are pointing to the exact same memory (same storage, offset, size an stri e).
`Tensor.is_shared`	Checks if tensor is in share memory.
`Tensor.is_signed`	Returns True if the ata type of `self` is a signe ata type.
`Tensor.is_sparse`	Is `True` if the Tensor uses sparse COO storage layout, `False` otherwise.
`Tensor.istft`	See `torch.istft()`
`Tensor.isreal`	See `torch.isreal()`
`Tensor.item`	Returns the value of this tensor as a stan ar Python num er.
`Tensor.kthvalue`	See `torch.kthvalue()`
`Tensor.lcm`	See `torch.lcm()`
`Tensor.lcm_`	In-place version of `lcm()`
`Tensor.ldexp`	See `torch.ldexp()`
`Tensor.ldexp_`	In-place version of `ldexp()`
`Tensor.le`	See `torch.le()`.
`Tensor.le_`	In-place version of `le()`.
`Tensor.less_equal`	See `torch.less_equal()`.
`Tensor.less_equal_`	In-place version of `less_equal()`.
`Tensor.lerp`	See `torch.lerp()`
`Tensor.lerp_`	In-place version of `lerp()`

`Tensor.lgamma_`	In-place version of `lgamma()`
`Tensor.log`	See `torch.log()`
`Tensor.log_`	In-place version of `log()`
`Tensor.logdet`	See `torch.logdet()`
`Tensor.log10`	See `torch.log10()`
`Tensor.log10_`	In-place version of `log10()`
`Tensor.log1p`	See `torch.log1p()`
`Tensor.log1p_`	In-place version of `log1p()`
`Tensor.log2`	See `torch.log2()`
`Tensor.log2_`	In-place version of `log2()`
`Tensor.log_normal_`	Fills `self` tensor with num ers samples from the log-normal istri ution parameterize y the given mean μ an stan ar eviation σ.
`Tensor.logaddexp`	See `torch.logaddexp()`
`Tensor.logaddexp2`	See `torch.logaddexp2()`
`Tensor.logsumexp`	See `torch.logsumexp()`
`Tensor.logical_and`	See `torch.logical_and()`
`Tensor.logical_and_`	In-place version of `logical_and()`
`Tensor.logical_not`	See `torch.logical_not()`
`Tensor.logical_not_`	In-place version of `logical_not()`
`Tensor.logical_or`	See `torch.logical_or()`
`Tensor.logical_or_`	In-place version of `logical_or()`
`Tensor.logical_xor`	See `torch.logical_xor()`
`Tensor.logical_xor_`	In-place version of `logical_xor()`
`Tensor.logit`	See `torch.logit()`
`Tensor.logit_`	In-place version of `logit()`
`Tensor.long`	`self.long()` is equivalent to `self.to(torch.int64)`.
`Tensor.lt`	See `torch.lt()`.
`Tensor.lt_`	In-place version of `lt()`.
`Tensor.less`	lt(other) -> Tensor
`Tensor.less_`	In-place version of `less()`.
`Tensor.lu`	See `torch.lu()`
`Tensor.lu_solve`	See `torch.lu_solve()`
`Tensor.as_subclass`	Makes a `cls` instance with the same ata pointer as `self`.
`Tensor.map_`	Applies `callable` for each element in `self` tensor an the given `tensor` an stores the results in `self` tensor.
`Tensor.masked_scatter_`	Copies elements from `source` into `self` tensor at positions where the `mask` is True.
`Tensor.masked_scatter`	Out-of-place version of `torch.Tensor.masked_scatter_()`
`Tensor.masked_fill_`	Fills elements of `self` tensor with `value` where `mask` is True.

`Tensor.masked_select`	See `torch.masked_select()`
`Tensor.matmul`	See `torch.matmul()`
`Tensor.matrix_power`	• NOTE `matrix_power()` is eprecate , use `torch.linalg.matrix_power()` instea .
`Tensor.matrix_exp`	See `torch.matrix_exp()`
`Tensor.max`	See `torch.max()`
`Tensor.maximum`	See `torch.maximum()`
`Tensor.mean`	See `torch.mean()`
`Tensor.module_load`	Defines how to transform `other` when loa ing it into `self` in `load_state_dict()`.
`Tensor.nanmean`	See `torch.nanmean()`
`Tensor.median`	See `torch.median()`
`Tensor.nanmedian`	See `torch.nanmedian()`
`Tensor.min`	See `torch.min()`
`Tensor.minimum`	See `torch.minimum()`
`Tensor.mm`	See `torch.mm()`
`Tensor.smm`	See `torch.smm()`
`Tensor.mode`	See `torch.mode()`
`Tensor.movedim`	See `torch.movedim()`
`Tensor.moveaxis`	See `torch.moveaxis()`
`Tensor.msort`	See `torch.msort()`
`Tensor.mul`	See `torch.mul()`.
`Tensor.mul_`	In-place version of `mul()`.
`Tensor.multiply`	See `torch.multiply()`.
`Tensor.multiply_`	In-place version of `multiply()`.
`Tensor.multinomial`	See `torch.multinomial()`
`Tensor.mv`	See `torch.mv()`
`Tensor.mvlgamma`	See `torch.mvlgamma()`
`Tensor.mvlgamma_`	In-place version of `mvlgamma()`
`Tensor.nansum`	See `torch.nansum()`
`Tensor.narrow`	See `torch.narrow()`.
`Tensor.narrow_copy`	See `torch.narrow_copy()`.
`Tensor.ndimension`	Alias for `dim()`
`Tensor.nan_to_num`	See `torch.nan_to_num()`.
`Tensor.nan_to_num_`	In-place version of `nan_to_num()`.
`Tensor.ne`	See `torch.ne()`.
`Tensor.ne_`	In-place version of `ne()`.
`Tensor.not_equal`	See `torch.not_equal()`.

`Tensor.neg`	See `torch.neg()`
`Tensor.neg_`	In-place version of `neg()`
`Tensor.negative`	See `torch.negative()`
`Tensor.negative_`	In-place version of `negative()`
`Tensor.nelement`	Alias for `numel()`
`Tensor.nextafter`	See `torch.nextafter()`
`Tensor.nextafter_`	In-place version of `nextafter()`
`Tensor.nonzero`	See `torch.nonzero()`
`Tensor.norm`	See `torch.norm()`
`Tensor.normal_`	Fills `self` tensor with elements samples from the normal istri ution parameterize y `mean` an `std`.
`Tensor.numel`	See `torch.numel()`
`Tensor.numpy`	Returns the tensor as a NumPy `ndarray`.
`Tensor.orgqr`	See `torch.orgqr()`
`Tensor.ormqr`	See `torch.ormqr()`
`Tensor.outer`	See `torch.outer()`.
`Tensor.permute`	See `torch.permute()`
`Tensor.pin_memory`	Copies the tensor to pinne memory, if it's not alrea y pinne .
`Tensor.pinverse`	See `torch.pinverse()`
`Tensor.polygamma`	See `torch.polygamma()`
`Tensor.polygamma_`	In-place version of `polygamma()`
`Tensor.positive`	See `torch.positive()`
`Tensor.pow`	See `torch.pow()`
`Tensor.pow_`	In-place version of `pow()`
`Tensor.prod`	See `torch.prod()`
`Tensor.put_`	Copies the elements from `source` into the positions specifie y `index`.
`Tensor.qr`	See `torch.qr()`
`Tensor.qscheme`	Returns the quantization scheme of a given QTensor.
`Tensor.quantile`	See `torch.quantile()`
`Tensor.nanquantile`	See `torch.nanquantile()`
`Tensor.q_scale`	Given a Tensor quantize y linear(affine) quantization, returns the scale of the un erlying quantizer().
`Tensor.q_zero_point`	Given a Tensor quantize y linear(affine) quantization, returns the zero_point of the un erlying quantizer().
`Tensor.q_per_channel_scales`	Given a Tensor quantize y linear (affine) per-channel quantization, returns a Tensor of scales of the un erlying quantizer.
`Tensor.q_per_channel_zero_points`	Given a Tensor quantize y linear (affine) per-channel quantization, returns a tensor of zero_points of the un erlying quantizer.
`Tensor.q_per_channel_axis`	Given a Tensor quantize y linear (affine) per-channel quantization, returns the in ex of imension on which per-channel quantization is applie .
`Tensor.rad2deg`	See `torch.rad2deg()`

`Tensor.ravel`	see `torch.ravel()`
`Tensor.reciprocal`	See `torch.reciprocal()`
`Tensor.reciprocal_`	In-place version of `reciprocal()`
`Tensor.record_stream`	Marks the tensor as having een use y this stream.
`Tensor.register_hook`	Registers a ackwar hook.
`Tensor.register_post_accumulate_grad_hook`	Registers a ackwar hook that runs after gra accumulation.
`Tensor.remainder`	See `torch.remainder()`
`Tensor.remainder_`	In-place version of `remainder()`
`Tensor.renorm`	See `torch.renorm()`
`Tensor.renorm_`	In-place version of `renorm()`
`Tensor.repeat`	Repeats this tensor along the specifie imensions.
`Tensor.repeat_interleave`	See `torch.repeat_interleave()`.
`Tensor.requires_grad`	Is `True` if gra ients nee to e compute for this Tensor, `False` otherwise.
`Tensor.requires_grad_`	Change if autogra shoul recor operations on this tensor: sets this tensor's `requires_grad` attri ute in-place.
`Tensor.reshape`	Returns a tensor with the same ata an num er of elements as `self` ut with the specifie shape.
`Tensor.reshape_as`	Returns this tensor as the same shape as `other`.
`Tensor.resize_`	Resizes `self` tensor to the specifie size.
`Tensor.resize_as_`	Resizes the `self` tensor to e the same size as the specifie `tensor`.
`Tensor.retain_grad`	Ena les this Tensor to have their `grad` populate uring `backward()`.
`Tensor.retains_grad`	Is `True` if this Tensor is non-leaf an its `grad` is ena le to e populate uring `backward()`, `False` otherwise.
`Tensor.roll`	See `torch.roll()`
`Tensor.rot90`	See `torch.rot90()`
`Tensor.round`	See `torch.round()`
`Tensor.round_`	In-place version of `round()`
`Tensor.rsqrt`	See `torch.rsqrt()`
`Tensor.rsqrt_`	In-place version of `rsqrt()`
`Tensor.scatter`	Out-of-place version of `torch.Tensor.scatter_()`
`Tensor.scatter_`	Writes all values from the tensor `src` into `self` at the in ices specifie in the `index` tensor.
`Tensor.scatter_add_`	A s all values from the tensor `src` into `self` at the in ices specifie in the `index` tensor in a similar fashion as `scatter_()`.
`Tensor.scatter_add`	Out-of-place version of `torch.Tensor.scatter_add_()`
`Tensor.scatter_reduce_`	Re uces all values from the `src` tensor to the in ices specifie in the `index` tensor in the `self` tensor using the applie re uction efine via the `reduce` argument (`"sum"`, `"prod"`, `"mean"`, `"amax"`, `"amin"`).
`Tensor.scatter_reduce`	Out-of-place version of `torch.Tensor.scatter_reduce_()`
`Tensor.select`	See `torch.select()`
`Tensor.select_scatter`	See `torch.select_scatter()`

`Tensor.share_memory_`	Moves the un erlying storage to share memory.
`Tensor.short`	`self.short()` is equivalent to `self.to(torch.int16)`.
`Tensor.sigmoid`	See `torch.sigmoid()`
`Tensor.sigmoid_`	In-place version of `sigmoid()`
`Tensor.sign`	See `torch.sign()`
`Tensor.sign_`	In-place version of `sign()`
`Tensor.signbit`	See `torch.signbit()`
`Tensor.sgn`	See `torch.sgn()`
`Tensor.sgn_`	In-place version of `sgn()`
`Tensor.sin`	See `torch.sin()`
`Tensor.sin_`	In-place version of `sin()`
`Tensor.sinc`	See `torch.sinc()`
`Tensor.sinc_`	In-place version of `sinc()`
`Tensor.sinh`	See `torch.sinh()`
`Tensor.sinh_`	In-place version of `sinh()`
`Tensor.asinh`	See `torch.asinh()`
`Tensor.asinh_`	In-place version of `asinh()`
`Tensor.arcsinh`	See `torch.arcsinh()`
`Tensor.arcsinh_`	In-place version of `arcsinh()`
`Tensor.shape`	Returns the size of the `self` tensor.
`Tensor.size`	Returns the size of the `self` tensor.
`Tensor.slogdet`	See `torch.slogdet()`
`Tensor.slice_scatter`	See `torch.slice_scatter()`
`Tensor.softmax`	Alias for `torch.nn.functional.softmax()`.
`Tensor.sort`	See `torch.sort()`
`Tensor.split`	See `torch.split()`
`Tensor.sparse_mask`	Returns a new sparse tensor with values from a stri e tensor `self` filtere y the in ices of the sparse tensor `mask`.
`Tensor.sparse_dim`	Return the num er of sparse imensions in a sparse tensor `self`.
`Tensor.sqrt`	See `torch.sqrt()`
`Tensor.sqrt_`	In-place version of `sqrt()`
`Tensor.square`	See `torch.square()`
`Tensor.square_`	In-place version of `square()`
`Tensor.squeeze`	See `torch.squeeze()`
`Tensor.squeeze_`	In-place version of `squeeze()`
`Tensor.std`	See `torch.std()`
`Tensor.stft`	See `torch.stft()`
`Tensor.storage`	Returns the un erlying `TypedStorage`.

`Tensor.storage_offset`	Returns `self` tensor's offset in the un erlying storage in terms of num er of storage elements (not ytes).
`Tensor.storage_type`	Returns the type of the un erlying storage.
`Tensor.stride`	Returns the stri e of `self` tensor.
`Tensor.sub`	See `torch.sub()`.
`Tensor.sub_`	In-place version of `sub()`
`Tensor.subtract`	See `torch.subtract()`.
`Tensor.subtract_`	In-place version of `subtract()`.
`Tensor.sum`	See `torch.sum()`
`Tensor.sum_to_size`	Sum `this` tensor to `size`.
`Tensor.svd`	See `torch.svd()`
`Tensor.swapaxes`	See `torch.swapaxes()`
`Tensor.swapdims`	See `torch.swapdims()`
`Tensor.t`	See `torch.t()`
`Tensor.t_`	In-place version of `t()`
`Tensor.tensor_split`	See `torch.tensor_split()`
`Tensor.tile`	See `torch.tile()`
`Tensor.to`	Performs Tensor type an /or evice conversion.
`Tensor.to_mkldnn`	Returns a copy of the tensor in `torch.mkldnn` layout.
`Tensor.take`	See `torch.take()`
`Tensor.take_along_dim`	See `torch.take_along_dim()`
`Tensor.tan`	See `torch.tan()`
`Tensor.tan_`	In-place version of `tan()`
`Tensor.tanh`	See `torch.tanh()`
`Tensor.tanh_`	In-place version of `tanh()`
`Tensor.atanh`	See `torch.atanh()`
`Tensor.atanh_`	In-place version of `atanh()`
`Tensor.arctanh`	See `torch.arctanh()`
`Tensor.arctanh_`	In-place version of `arctanh()`
`Tensor.tolist`	Returns the tensor as a (neste) list.
`Tensor.topk`	See `torch.topk()`
`Tensor.to_dense`	Creates a stri e copy of `self` if `self` is not a stri e tensor, otherwise returns `self`.
`Tensor.to_sparse`	Returns a sparse copy of the tensor.
`Tensor.to_sparse_csr`	Convert a tensor to compresse row storage format (CSR).
`Tensor.to_sparse_csc`	Convert a tensor to compresse column storage (CSC) format.
`Tensor.to_sparse_bsr`	Convert a tensor to a lock sparse row (BSR) storage format of given locksize.
`Tensor.to_sparse_bsc`	Convert a tensor to a lock sparse column (BSC) storage format of given locksize.

`Tensor.transpose_`	In-place version of `transpose()`
`Tensor.triangular_solve`	See `torch.triangular_solve()`
`Tensor.tril`	See `torch.tril()`
`Tensor.tril_`	In-place version of `tril()`
`Tensor.triu`	See `torch.triu()`
`Tensor.triu_`	In-place version of `triu()`
`Tensor.true_divide`	See `torch.true_divide()`
`Tensor.true_divide_`	In-place version of `true_divide_()`
`Tensor.trunc`	See `torch.trunc()`
`Tensor.trunc_`	In-place version of `trunc()`
`Tensor.type`	Returns the type if *dtype* is not provi e , else casts this o ect to the specifie type.
`Tensor.type_as`	Returns this tensor cast to the type of the given tensor.
`Tensor.unbind`	See `torch.unbind()`
`Tensor.unflatten`	See `torch.unflatten()`.
`Tensor.unfold`	Returns a view of the original tensor which contains all slices of size `size` from `self` tensor in the imension `dimension`.
`Tensor.uniform_`	Fills `self` tensor with num ers sample from the continuous uniform istri ution:
`Tensor.unique`	Returns the unique elements of the input tensor.
`Tensor.unique_consecutive`	Eliminates all ut the first element from every consecutive group of equivalent elements.
`Tensor.unsqueeze`	See `torch.unsqueeze()`
`Tensor.unsqueeze_`	In-place version of `unsqueeze()`
`Tensor.values`	Return the values tensor of a sparse COO tensor.
`Tensor.var`	See `torch.var()`
`Tensor.vdot`	See `torch.vdot()`
`Tensor.view`	Returns a new tensor with the same ata as the `self` tensor ut of a ifferent `shape`.
`Tensor.view_as`	View this tensor as the same size as `other`.
`Tensor.vsplit`	See `torch.vsplit()`
`Tensor.where`	`self.where(condition, y)` is equivalent to `torch.where(condition, self, y)`.
`Tensor.xlogy`	See `torch.xlogy()`
`Tensor.xlogy_`	In-place version of `xlogy()`
`Tensor.xpu`	Returns a copy of this o ect in XP memory.
`Tensor.zero_`	Fills `self` tensor with zeros.

‹ Previous

Next ›

View Docs > View Tutorials > View Resources >

PyTorch

Get Starte
Features
Ecosystem
Blog
Contri uting

Resources

Tutorials
Docs
Discuss
Githu Issues
Bran Gui elines

Stay u to ate

Face ook
Twitter
YouTu e
Linke In

PyTorch Po casts

S otify
A le
Google
Amazon

Terms | Privacy