Complex Numbers

Complex numbers are numbers that can be expressed in the form $a + bj$, where a and b are real numbers, and j is called the imaginary unit, which satisfies the equation $j^2 = -1$. Complex numbers frequently occur in mathematics and engineering, especially in topics like signal processing. Traditionally many users and libraries (e.g., TorchAudio) have handled complex numbers by representing the data in float tensors with shape (..., 2) where the last dimension contains the real and imaginary values.

Tensors of complex dtypes provide a more natural user experience while working with complex numbers. Operations on complex tensors (e.g., :func:`torch.mv`, :func:`torch.matmul`) are likely to be faster and more memory efficient than operations on float tensors mimicking them. Operations involving complex numbers in PyTorch are optimized to use vectorized assembly instructions and specialized kernels (e.g. LAPACK, cuBlas).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 12); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 12); *backlink*

Unknown interpreted text role "func".

Note

Spectral operations in the torch.fft module support native complex tensors.

Warning

Complex tensors is a beta feature and subject to change.

Creating Complex Tensors

We support two complex dtypes: *torch.cfloat* and *torch.cdouble*

```
>>> x = torch.randn(2,2, dtype=torch.cfloat)
>>> x
tensor([[-0.4621-0.0303j, -0.2438-0.5874j],
        [ 0.7706+0.1421j,  1.2110+0.1918j]])
```

Note

The default dtype for complex tensors is determined by the default floating point dtype. If the default floating point dtype is *torch.float64* then complex numbers are inferred to have a dtype of *torch.complex128*, otherwise they are assumed to have a dtype of *torch.complex64*.

All factory functions apart from :func:`torch.linspace`, :func:`torch.logspace`, and :func:`torch.arange` are supported for complex tensors.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 42); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 42); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 42); *backlink*

Unknown interpreted text role "func".

Transition from the old representation

Users who currently worked around the lack of complex tensors with real tensors of shape (..., 2) can easily to switch using the complex tensors in their code using :func:`torch.view_as_complex` and :func:`torch.view_as_real`. Note that these functions donâ€™t perform any copy and return a view of the input tensor.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 48); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 48); *backlink*

Unknown interpreted text role "func".

```
>>> x = torch.randn(3, 2)
>>> x
tensor([[ 0.6125, -0.1681],
     [-0.3773,  1.3487],
     [-0.0861, -0.7981]])
>>> y = torch.view_as_complex(x)
>>> y
tensor([ 0.6125-0.1681j, -0.3773+1.3487j, -0.0861-0.7981j])
>>> torch.view_as_real(y)
tensor([[ 0.6125, -0.1681],
     [-0.3773,  1.3487],
     [-0.0861, -0.7981]])
```

Accessing real and imag

The real and imaginary values of a complex tensor can be accessed using the :attr:`real` and :attr:`imag`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 71); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 71); *backlink*

Unknown interpreted text role "attr".

Note

Accessing *real* and *imag* attributes doesn't allocate any memory, and in-place updates on the *real* and *imag* tensors will update the original complex tensor. Also, the returned *real* and *imag* tensors are not contiguous.

```
>>> y.real
tensor([ 0.6125, -0.3773, -0.0861])
>>> y.imag
tensor([-0.1681,  1.3487, -0.7981])

>>> y.real.mul_(2)
tensor([ 1.2250, -0.7546, -0.1722])
>>> y
tensor([ 1.2250-0.1681j, -0.7546+1.3487j, -0.1722-0.7981j])
>>> y.real.stride()
(2,)
```

Angle and abs

The angle and absolute values of a complex tensor can be computed using :func:`torch.angle` and :func:`torch.abs`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 96); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 96); *backlink*

Unknown interpreted text role "func".

```
>>> x1=torch.tensor([3j, 4+4j])
>>> x1.abs()
tensor([3.0000, 5.6569])
>>> x1.angle()
tensor([1.5708, 0.7854])
```

Linear Algebra

Many linear algebra operations, like :func:`torch.matmul`, :func:`torch.svd`, :func:`torch.solve` etc., support complex numbers. If you'd like to request an operation we don't currently support, please search if an issue has already been filed and if not, file one.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 110); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 110); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 110); *backlink*

Unknown interpreted text role "func".

Serialization

Complex tensors can be serialized, allowing data to be saved as complex values.

```
>>> torch.save(y, 'complex_tensor.pt')
>>> torch.load('complex_tensor.pt')
tensor([ 0.6125-0.1681j, -0.3773+1.3487j, -0.0861-0.7981j])
```

Autograd

PyTorch supports autograd for complex tensors. The gradient computed is the Conjugate Wirtinger derivative, the negative of which is precisely the direction of steepest descent used in Gradient Descent algorithm. Thus, all the existing optimizers work out of the box with complex parameters. For more details, check out the note :ref:`complex_autograd-doc`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\(pytorch-master)(docs)(source)complex_numbers.rst, line 130); *backlink*

Unknown interpreted text role "ref".

We do not fully support the following subsystems:

- Quantization
- JIT
- Sparse Tensors
- Distributed

If any of these would help your use case, please search if an issue has already been filed and if not, file one.