**Context and goals**

**Mission:** Accelerate research on transformers [5]

- Researchers first
- Efficient components
- Customizable blocks
- Open source library

https://github.com/facebookresearch/xformers

**Getting started**

Option 1 (recommended): Use conda binaries if available

```
$ conda install xformers -c xformers/label/dev
```

Option 2 (slower): Build from source. See our README

```
$ pip install -U git+https://github.com/facebookresearch/xformers.git@main#egg=xformers
```

**Fast & memory-efficient attention**

```
def self_attention_block(self, x):
    QKV = self.qkv(x).reshape(B, N, 3, H, K)
    QKV = QKV.permute(2, 0, 3, 1, 4)
    Q, K, V = torch.unbind(QKV, 0)
    attn = Q @ K.transpose(-2, -1)
    attn = attn.softmax(dim=-1)
    y = (attn @ V).transpose(1, 2)
    y = y.reshape(B, N, C)
    return y
```

```
def self_attention_block(self, x):
    QKV = self.qkv(x).reshape(B, N, 3, H, K)
    Q, K, V = xops.unbind(QKV, 2)
    y = xops.memory_efficient_attention(Q, K, V,
            attn_bias=None,
            op=None)
    return y
```

**Supported configurations**

xFormers automatically dispatches to the most efficient operator for the given input shape/dtype, options and GPU generation.

<table>
<thead>
<tr>
<th>Operator</th>
<th>GPU support</th>
<th>Datatype</th>
<th>Dropout</th>
<th>Cross-attention</th>
<th>Causal support</th>
<th>Nested tensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUTLASS</td>
<td>6.0+ (P100)</td>
<td>F16/F32/16/128</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Flash</td>
<td>7.5+ (T4)</td>
<td>F16/F32</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

More coming

We can also mix operations by using CUTLASS’s forward with Flash’s backward for instance

**References**

[1] Markus N Rabe and Charles Staats: Self-attention does not need $O(n^2)$ memory


