# einops
# [Rogozhnikov, 2022]

Fedor Scholz

University of Tübingen

November 4, 2024

# Motivation

Output shape of pytorch's DataLoader: [batch, time, channel]
Input shape of pytorch's LSTM: [time, batch, channel]

# Motivation

Output shape of pytorch's DataLoader: [batch, time, channel]
Input shape of pytorch's LSTM: [time, batch, channel]

```
# pytorch solution
x = x.transpose(1, 0, 2)
```

# Motivation

Output shape of pytorch's DataLoader: [batch, time, channel]
Input shape of pytorch's LSTM: [time, batch, channel]

```
# pytorch solution
x = x.transpose(1, 0, 2)
# einops solution
x = einops.rearrange(x, 'b t c -> t b c')
```

# Introduction

From https://einops.rocks/:

"Flexible and powerful tensor operations for readable and reliable code. Supports numpy, pytorch, tensorflow, jax, and others."

Installation:

```
pip install einops
```

# Advantages

From https://einops.rocks/:

- Semantic information: "what is the input and output, not how the output is computed"
- Convenient checks of number and sizes of dimensions
- Result is strictly determined, e.g., in space2depth
- Uniformity, e.g., 1d/2d/3d pooling look similar
- Framework-independent behavior: `.flatten()` not always the same
- Independence of framework terminology: numpy's `.tile()` vs pytorch's `.repeat()`

# Operations

- `rearrange`: keeps number of elements, replaces `transpose`, `reshape`, `stack`, `concatenate`, `squeeze`, `expand_dims`
- `reduce`: `rearrange` with reductions: `mean`, `min`, `max`, `sum`, `prod`
- `repeat`: `rearrange` with repeating and tiling

# rearrange

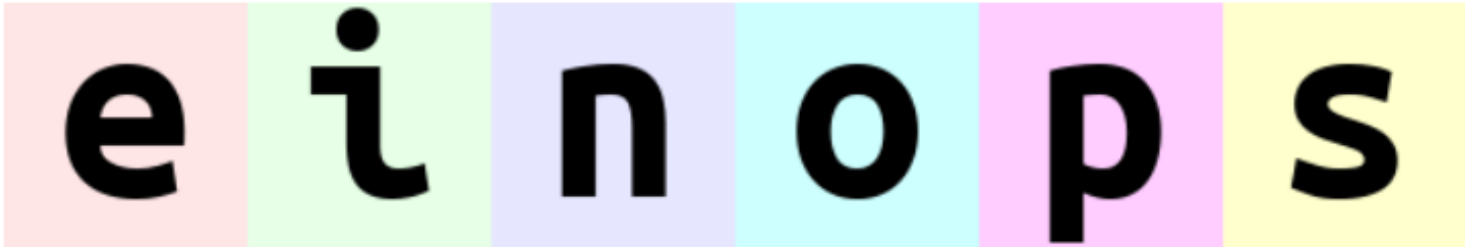
In [6]:
```
# rearrange, as its name suggests, rearranges elements
# below we swapped height and width.
# In other words, transposed first two axes (dimensions)
rearrange(ims[0], "h w c -> w h c")
```

Out[6]:

# rearrange

In [8]:
```
# or compose a new dimension of batch and width
rearrange(ims, "b h w c -> h (b w) c")
```

Out[8]:



# rearrange

In [12]:
```
# finally, combine composition and decomposition:
rearrange(ims, "(b1 b2) h w c -> (b1 h) (b2 w) c ", b1=2)
```

Out[12]:

# reduce

In [18]:
```
# average over batch
reduce(ims, "b h w c -> h w c", "mean")
```

Out[18]:

# repeat

In [33]:
```
# repeat along w (existing axis)
repeat(ims[0], "h w c -> h (repeat w) c", repeat=3)
```

Out[33]:

# Layer

Conveniently, einops provides pytorch layers:

```
from torch.nn import Sequential, Conv2d, MaxPool2d, Linear, ReLU
from einops.layers.torch import Rearrange

model = Sequential(
    ...,
    Conv2d(6, 16, kernel_size=5),
    MaxPool2d(kernel_size=2),
    # flattening without need to write forward
    Rearrange('b c h w -> b (c h w)'),
    Linear(16 * 5 * 5, 120),
    ReLU(),
    Linear(120, 10),
)
```

# Outlook

There is more: einsum, pack, unpack

# References I

Rogozhnikov, A. (2022).

Einops: Clear and reliable tensor manipulations with einstein-like notation.
In *International Conference on Learning Representations*.