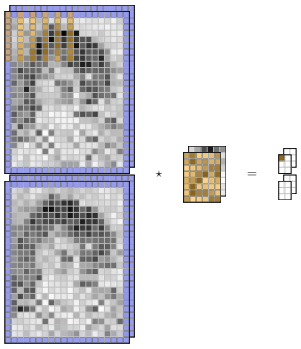
Convolutions and More as Einsum

Felix Dangel

NeurIPS 2024

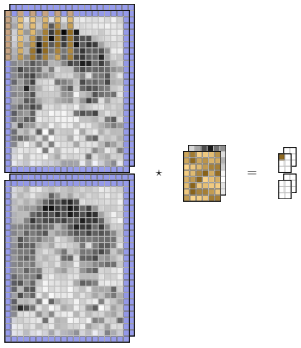
Conceptual Overview

Convolution

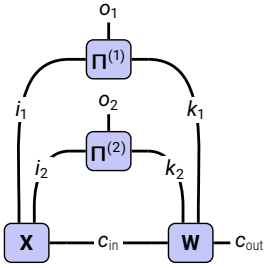


Conceptual Overview

Convolution

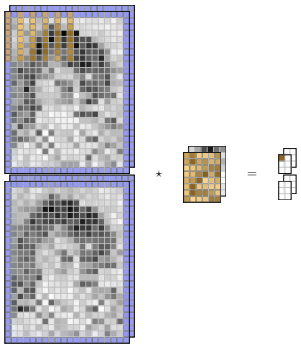


Tensor networks

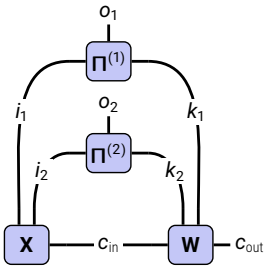


Conceptual Overview

Convolution



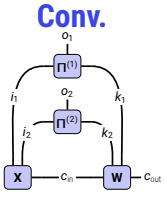
Tensor networks



Evaluation

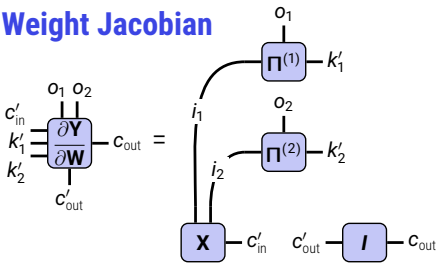
```
einsum("c_in i1 i2, k1 o1 i1, k2 o2 i2, c_out c_in k1 k2 -> c_out o1 o2", X, Pi1, Pi2, W)
```

Derivatives & Autodiff Routines

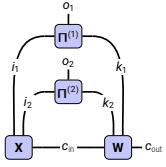


Derivatives & Autodiff Routines

Weight Jacobian

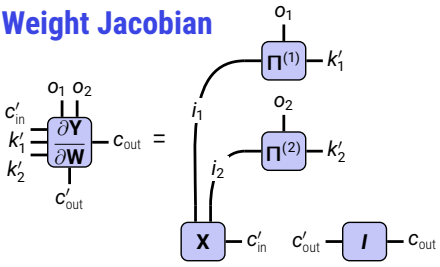


Conv.

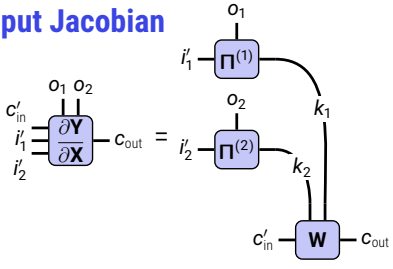


Derivatives & Autodiff Routines

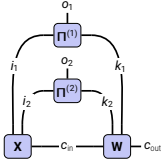
Weight Jacobian



Input Jacobian

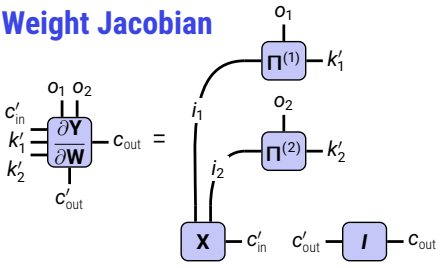


Conv.

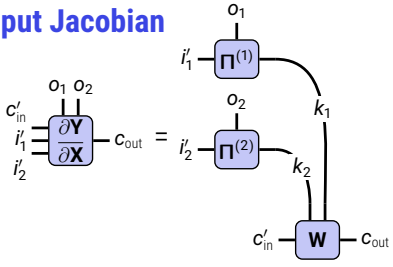


Derivatives & Autodiff Routines

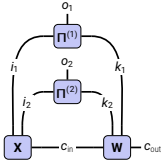
Weight Jacobian



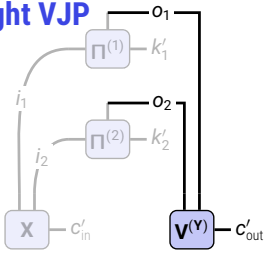
Input Jacobian



Conv.

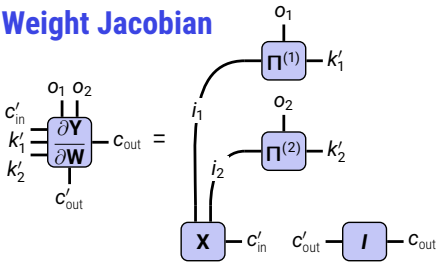


Weight VJP

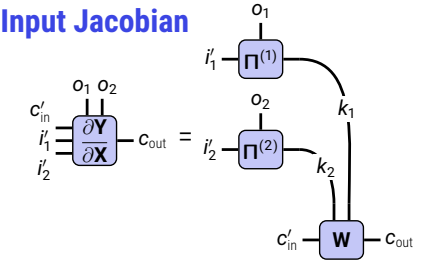


Derivatives & Autodiff Routines

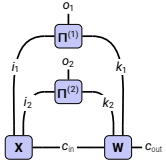
Weight Jacobian



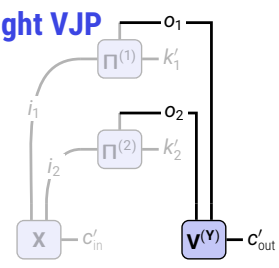
Input Jacobian



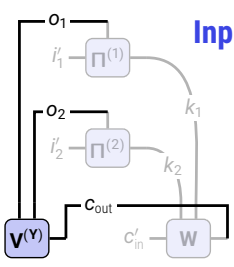
Conv.



Weight VJP



Input VJP



Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

State of the art

x

Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

State of the art

$\mathbf{X}$

$\rightarrow [\![\mathbf{X}]\!]$

Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

State of the art

$\mathbf{X}$

$\rightarrow [\![\mathbf{X}]\!]$

$\rightarrow \mathbf{1}^\top [\![\mathbf{X}]\!]$

Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

State of the art

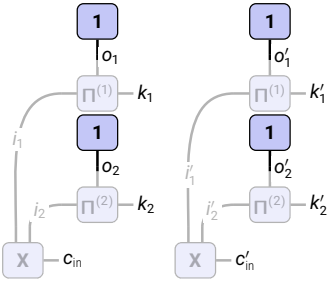
$$\mathbf{X}$$
$$\rightarrow [\![\mathbf{X}]\!]$$
$$\rightarrow \mathbf{1}^\top [\![\mathbf{X}]\!]$$
$$\rightarrow \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)^\top \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)$$

Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

State of the art

$$\mathbf{X}$$
$$\to [\![\mathbf{X}]\!]$$
$$\to \mathbf{1}^\top [\![\mathbf{X}]\!]$$
$$\to \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)^\top \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)$$

Tensor Network



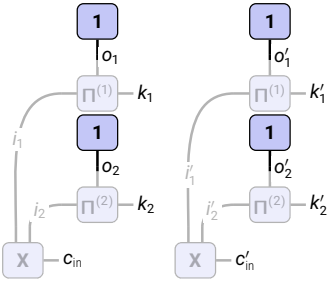
Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

State of the art

$$
\begin{aligned}
&\mathbf{X} \\
&\rightarrow [\![\mathbf{X}]\!] \\
&\rightarrow \mathbf{1}^\top [\![\mathbf{X}]\!] \\
&\rightarrow \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)^\top \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)
\end{aligned}
$$

Time: **9.87 ms**

Tensor Network



Time: **2.69 ms (3.7 x)**

(`features.1.0.block.0` convolution of ConvNeXt-base with (32, 3, 256, 256) input)

Improving KFAC-reduce [Eschenhagen et al., 2023, NeurIPS]

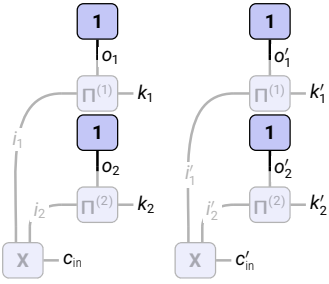
Non-conventional operations can be much cheaper with tensor networks

State of the art

$$
\begin{aligned}
&\mathbf{X} \\
&\rightarrow [\![\mathbf{X}]\!] \\
&\rightarrow \mathbf{1}^\top [\![\mathbf{X}]\!] \\
&\rightarrow \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)^\top \left(\mathbf{1}^\top [\![\mathbf{X}]\!]\right)
\end{aligned}
$$

Time: **9.87 ms**
Extra memory: **3.07 GiB**

Tensor Network



Time: **2.69 ms (3.7 x)**
Extra memory: **0 MiB**

(`features.1.0.block.0` convolution of ConvNeXt-base with (32, 3, 256, 256) input)

Improving Second-Order Optimizers

[Lin, Dangel, Eschenhagen, Neklyudov, Kristiadi, Turner, and Makhzani, 2024, ICML]

ImageNet GPU benchmark (NVIDIA A40) with (128, 3, 256, 256) inputs

Net	Optimizer	Per-iteration [s]	Peak memory [GiB]
ResNet18	SGD	$1.17 \cdot 10^{-1}$ (1.00 x)	3.62 (1.00 x)
VGG19	SGD	$6.90 \cdot 10^{-1}$ (1.00 x)	14.1 (1.00 x)

Improving Second-Order Optimizers

[Lin, Dangel, Eschenhagen, Neklyudov, Kristiadi, Turner, and Makhzani, 2024, ICML]

ImageNet GPU benchmark (NVIDIA A40) with (128, 3, 256, 256) inputs

Net	Optimizer	Per-iteration [s]	Peak memory [GiB]
ResNet18	SGD	$1.17 \cdot 10^{-1}$ (1.00 x)	3.62 (1.00 x)
	SINGD	$1.94 \cdot 10^{-1}$ **(1.67 x)**	4.54 **(1.25 x)**
VGG19	SGD	$6.90 \cdot 10^{-1}$ (1.00 x)	14.1 (1.00 x)
	SINGD	1.02 **(1.48 x)**	32.1 **(2.28 x)**

(SINGD uses KFAC-reduce and diagonal pre-conditioners which are updated every step)

Improving Second-Order Optimizers

[Lin, Dangel, Eschenhagen, Neklyudov, Kristiadi, Turner, and Makhzani, 2024, ICML]

ImageNet GPU benchmark (NVIDIA A40) with (128, 3, 256, 256) inputs

Net	Optimizer	Per-iteration [s]	Peak memory [GiB]
ResNet18	SGD	$1.17 \cdot 10^{-1}$ (1.00 x)	3.62 (1.00 x)
	SINGD	$1.94 \cdot 10^{-1}$ **(1.67 x)**	4.54 **(1.25 x)**
	SINGD+TN	$1.56 \cdot 10^{-1}$ **(1.33 x)**	3.63 **(1.00 x)**
VGG19	SGD	$6.90 \cdot 10^{-1}$ (1.00 x)	14.1 (1.00 x)
	SINGD	1.02 **(1.48 x)**	32.1 **(2.28 x)**
	SINGD+TN	$8.01 \cdot 10^{-1}$ **(1.16 x)**	16.1 **(1.14 x)**

(SINGD uses KFAC-reduce and diagonal pre-conditioners which are updated every step)

Significantly reduces the computational gap of second-order methods.

Convolutions and More as `einsum`

- TN perspective simplifies the transfer of algorithmic ideas
- Enables flexible/faster implementations of black box routines
- Relies on automatically efficient evaluation inside `einsum`

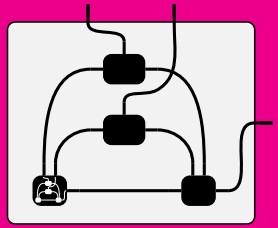
Convolutions and More as `einsum`

- TN perspective simplifies the transfer of algorithmic ideas
- Enables flexible/faster implementations of black box routines
- Relies on automatically efficient evaluation inside `einsum`

Try it out!

```
from einconv.expressions import kfac_reduce
from torch import einsum

# create the tensor network
equation, operands, shape = kfac_reduce.
    einsum_expression(..., simplify=True)
# evaluate it
einsum(equation, *operands).reshape(shape)
```



`pip install einconv`

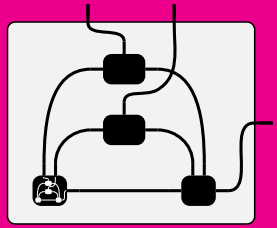
Convolutions and More as `einsum`

- TN perspective simplifies the transfer of algorithmic ideas
- Enables flexible/faster implementations of black box routines
- Relies on automatically efficient evaluation inside `einsum`

Try it out!

```
from einconv.expressions import kfac_reduce
from torch import einsum

# create the tensor network
equation, operands, shape = kfac_reduce.
    einsum_expression(..., simplify=True)
# evaluate it
einsum(equation, *operands).reshape(shape)
```



pip install einconv

Paper: arxiv/2307.02275

Code: github.com/f-dangel/einconv

Thank you 🙏 questions?

References I

Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In **Advances in Neural Information Processing Systems (NeurIPS)**, 2023.

Wu Lin, Felix Dangel, Runa Eschenhagen, Kirill Neklyudov, Agustinus Kristiadi, Richard E. Turner, and Alireza Makhzani. Structured inverse-free natural gradient descent: Memory-efficient & numerically-stable KFAC. In **International Conference on Machine Learning (ICML)**, 2024.