Learning Operations for Neural PDE Solvers

ICLR 2021 Workshop Deep Learning for Simulation (simDL)

Nicholas Roberts*, Mikhail Khodak*, Tri Dao, Liam Li, Christopher Ré, Ameet Talwalkar
* Equal contribution; CMU, Stanford, Determined AI

Contributions

XD-operations:
A neural architecture search-inspired efficient relaxation of convolutions based on their DFT diagonalization.

Theoretical properties:
- efficient to store and apply (log-linear in the input size)
- can express numerous different operations, including convolutions, graph convolutions (Kipf & Welling, 2017), and FNOs (Li et al., 2021)

Empirical properties:
- when initialized as random convolutions, XD-operations learn a more accurate PDE solver than FNOs on three sets of equations

XD-operations

Key idea: replace the DFTs in the standard diagonalization of a convolution by a more general family of efficient matrices

filter weights input

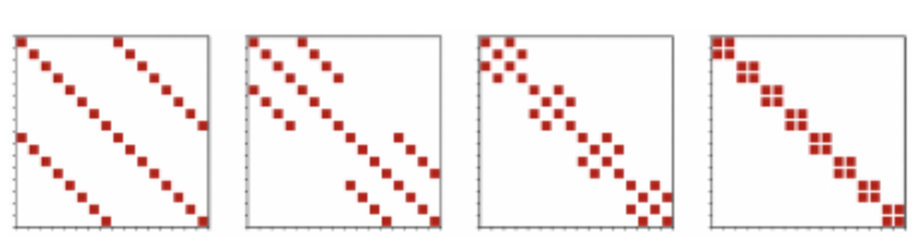
$$\mathbf{Conv}(\mathbf{w})(\mathbf{x}) = \mathbf{A}_{\mathbf{w}}\mathbf{x}$$

DFT

$$= \mathbf{F}^{-1}\,\mathrm{diag}\,(\mathbf{F}\underline{\mathbf{w}})\,\mathbf{F}\mathbf{x}$$

$$\mathbf{XD}^{1}_{\alpha}(\mathbf{w})(\mathbf{x}) = \mathrm{Real}\,(\mathbf{K}\,\mathrm{diag}\,(\mathbf{L}\underline{\mathbf{w}})\,\mathbf{M}\mathbf{x})$$

FFT-like product of small factor matrices



Provably expresses any efficient matrix-vector operation:
- sparse
- low-rank
- permutation
- DFT
- DCT
- wavelet transform

Specifically, Kaleidoscope (K-) matrices (Dao et al., 2020)

Properties

Efficiency:
- asymptotically, takes time log-linear in the input size to store and apply
- empirically, 3-5x slower than convolutions, with scope for improving performance.

Expressivity:
If viewed as an architecture search space, XD-operations contain
- convolutions of all types
- graph convolutions for fixed graphs
- average pooling
- skip-connections
- Fourier neural operator
- convolution composed with permutation
- infinitely many more operations

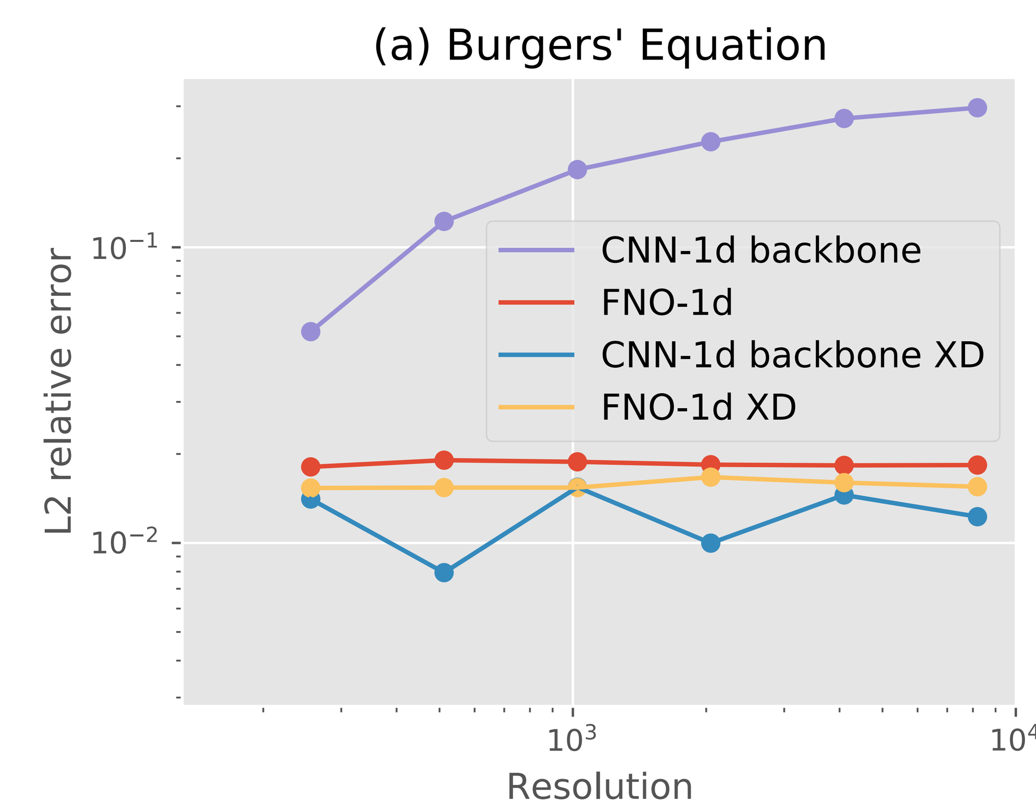
Experimental setup

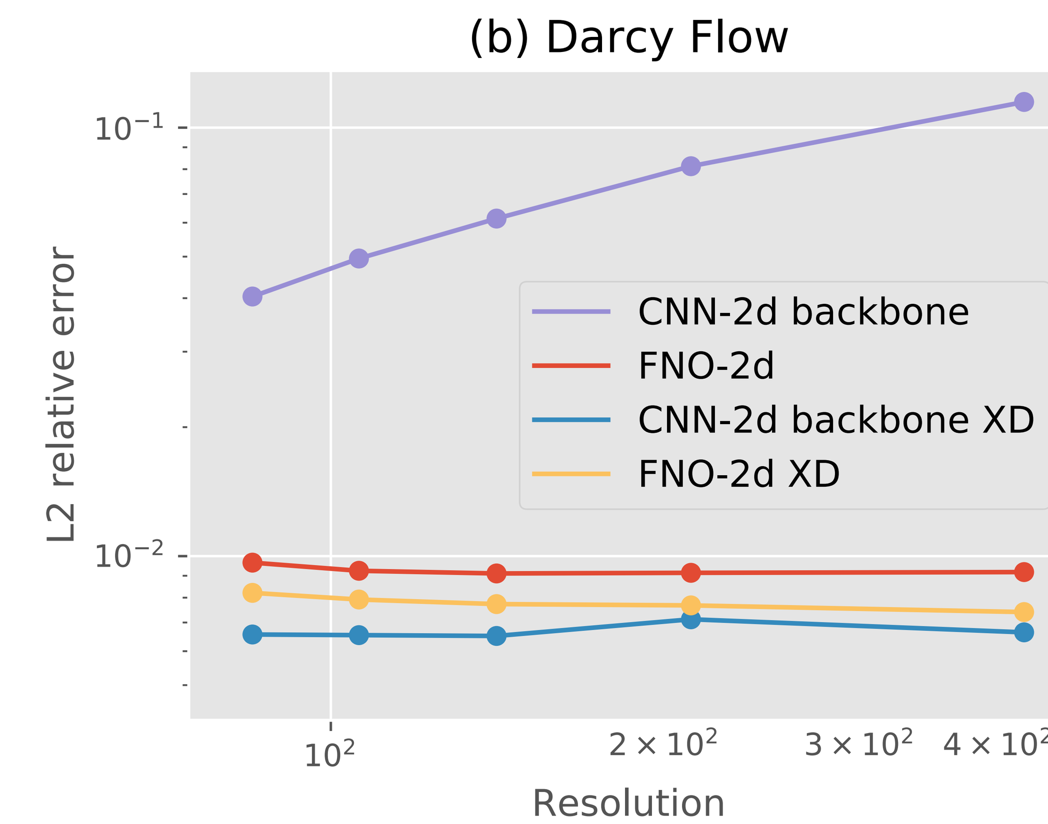
Model: ResNet-like backbone model from Li et al. (2021)

Tasks: learn a mapping from initial conditions to solutions using data generated by a classical PDE solver for
- Burgers' equations (1d)
- Darcy flow (2d)
- 2d Navier-Stokes (3d)

Training:
- train regular model weights using routine from Li et al. (2021)
- simultaneously train K-matrices using SGD with momentum

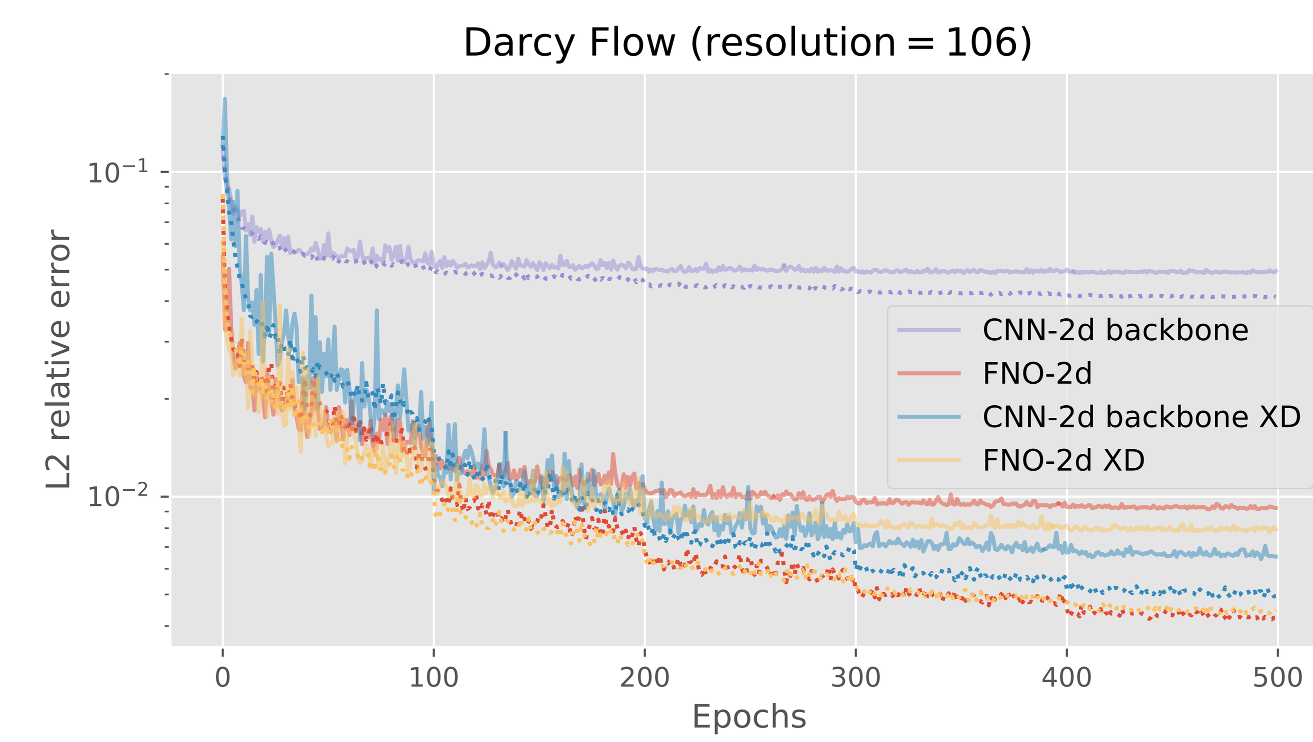
Results





Navier-Stokes

Method	$\nu = 10^{-4}$ $T = 30$	$\nu = 10^{-5}$ $T = 20$
CNN-3d backbone	0.325	0.278
FNO-3d (reproduced)	0.182	0.177
CNN-3d backbone XD	**0.172**	**0.168**



Next steps

Efficiency:
- improve efficiency by improving padding and ablating individual K-matrices
- handling transfer to different grids and irregular meshes
- initializing with other operations such as graph convolutions

References

Dao et al. Kaleidoscope: An efficient, learnable representation for all structured linear maps. ICLR 2020.
Kipf & Welling. Semi-supervised classification with graph convolutional networks. ICLR 2017.
Li et al. Fourier neural operator for parametric partial differential equations. ICLR 2021.