CS839: AI for Scientific Computing
Neural Operators

Misha Khodak

University of Wisconsin-Madison

3 February 2026

Outline

- Motivation
- Basic architectures
- Successes and challenges
- Ongoing directions

Outline

- Motivation
- Basic architectures
- Successes and challenges
- Ongoing directions

Motivation

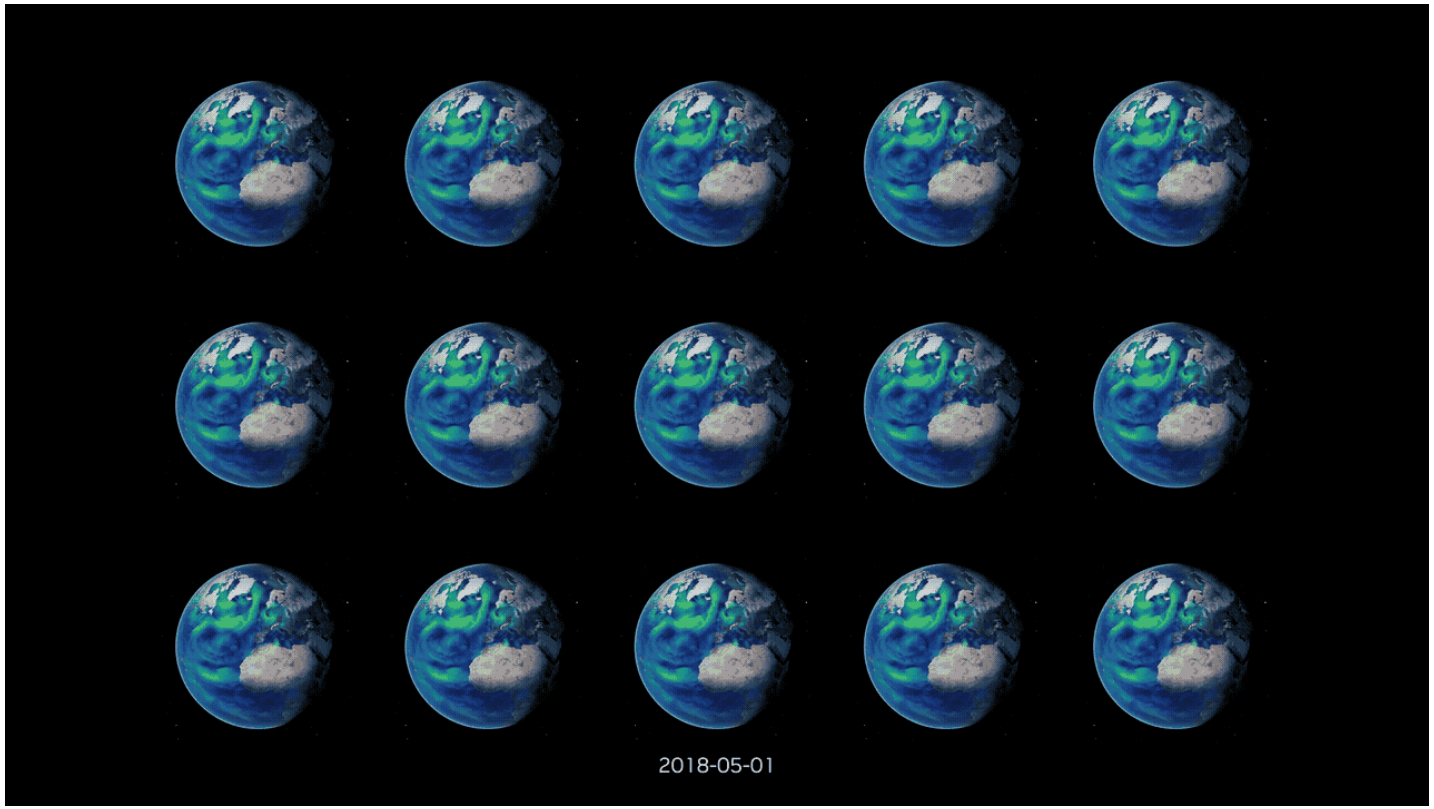
Say we have a time-dependent PDE

$$\partial_t u(t,x) = L[u](t,x) + c(t,x)$$

over domain Ω, and suppose we want to solve it for many different initial conditions $a(x) = u(0,\mathrm{x})$, boundary conditions $b(t,x)$, $x \in \partial\Omega$, and forcing functions $c(t,x)$, $t \geq 0, x \in \Omega$.

When do we need to solve a parameterized PDE multiple times?

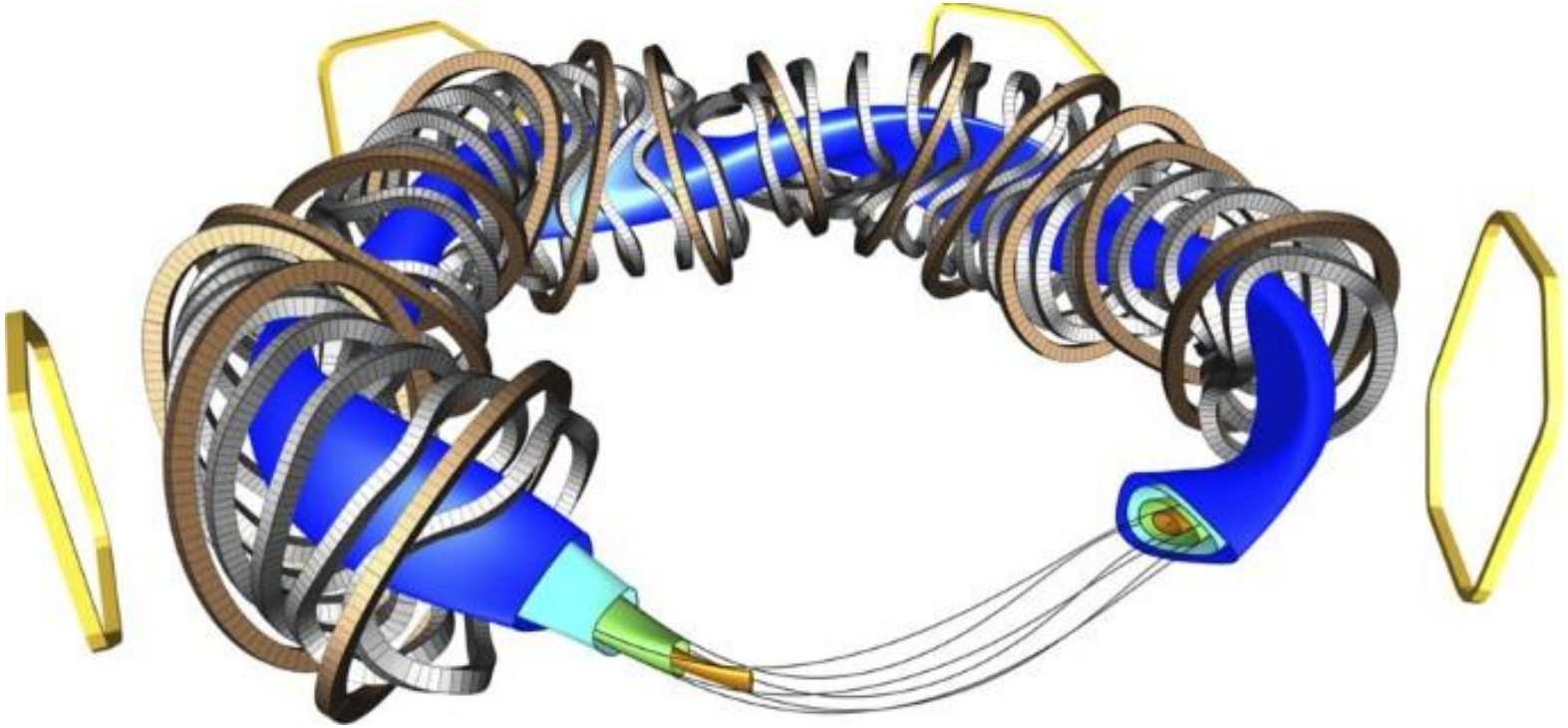
Forecasting: multiple initial / boundary conditions



NVIDIA

When do we need to solve a parameterized PDE multiple times?

Inverse problems: multiple forcing functions and initial conditions



Wendelstein 7-X

Classical approach

Run the simulation multiple times, perhaps at different accuracies

Advantages:

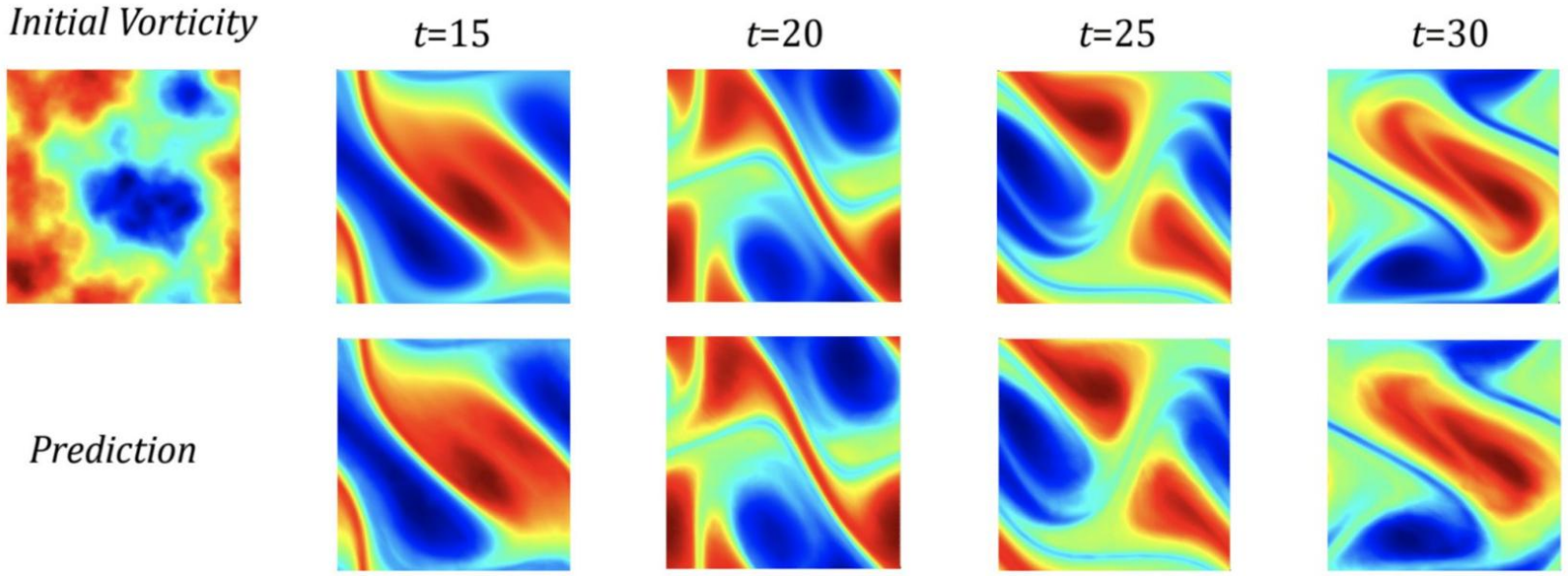
- lots of existing solvers and software
- correctness guarantees

Disadvantages:

- can be very expensive

The neural operator approach

1. generate solutions $u_{a,b,c}(t,x)$ for many different a, b, c
2. use the data to learn a neural network $f_\theta(a,b,c) \mapsto u_{a,b,c}$
3. generate future solutions $u_{a,b,c}$ using f_θ



Zongyi Li

The neural operator approach

Advantages: fast generation

- just forwarded passes with f_θ
- possibly fewer timesteps

Disadvantages

- no correctness guarantees
- original data can be expensive to generate

Comparing to PINNs (next lecture)

PINNs pose a neural network ansatz $u = f_\theta$ and train it to minimize the PDE residual:

$$\min_\theta \ \|\partial_t f_\theta - L[f_\theta] - c\|^2$$

Neural operators train f_θ to match classically generated solutions:

$$\min_\theta \sum_{i=1}^{n} \left\| f_\theta(a_i, b_i, c_i) - u_{a_i,b_i,c_i} \right\|^2$$

Comparing to PINNs (next lecture)

Both:
- Advantage: can incorporate experimental data
- Disadvantage: non-guaranteed optimization

PINNs:
- Advantage: no need for classically generated training data
- Disadvantage: training requires higher-order derivatives

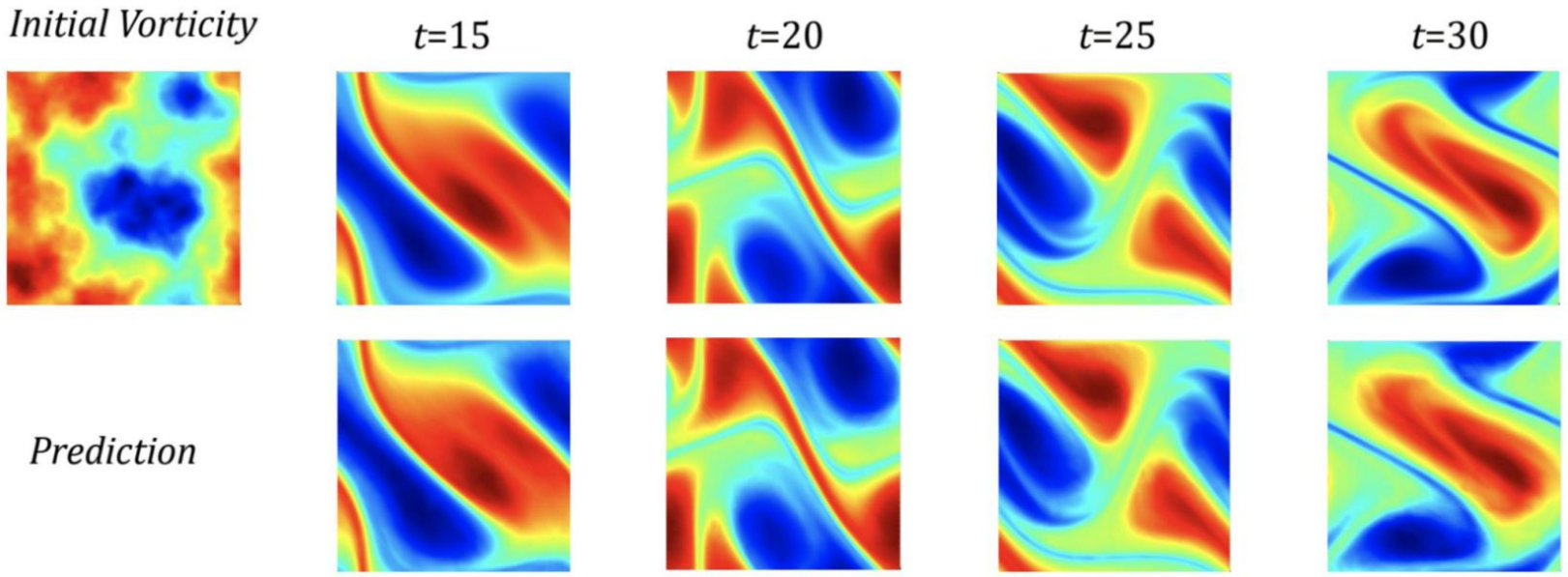
Neural operators:
- Advantage: very efficient when trained, can take large timesteps
- Disadvantage: requires classically generated training data

Outline

- Motivation
- Basic architectures
- Successes and challenges
- Ongoing directions

What kind of architecture do we need?

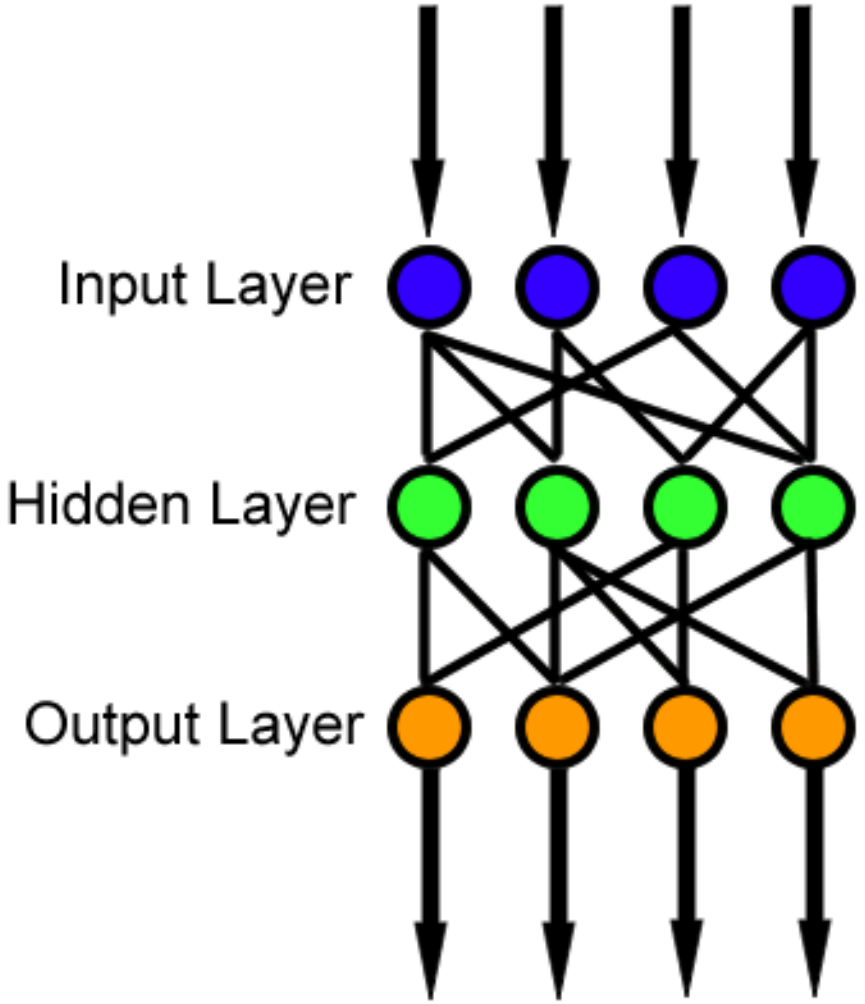
- output size scales with input size
- works with high-resolution multi-dimensional data



Zongyi Li

Recall: A simple feedforward neural network

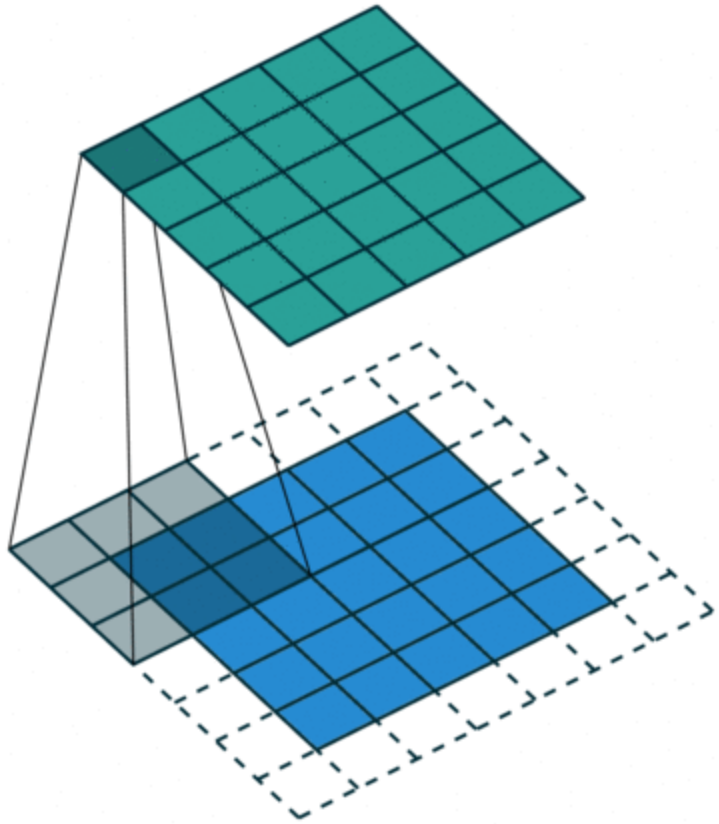
- inputs and outputs constrained by the parameter sizes ⊗
  - can only use if we always have the same size input and output
- parameter sizes scale with the number of grid points ⊗
  - massive parameter counts even for moderate resolutions



Wikipedia

Recall: Convolutional neural networks

- replaces linear layer by convolutions with small, parameterized kernels
- can handle variable input/output shapes ✓
- parameter count is independent of the input resolution ✓

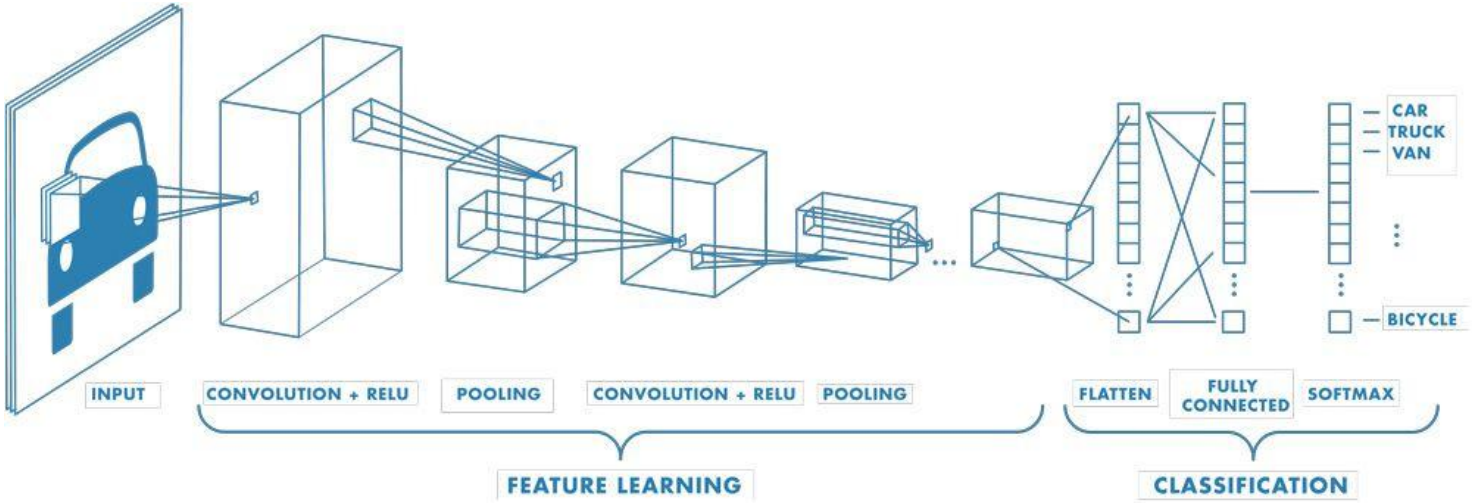


Vincent Dumoulin

What kind of CNN architecture is suitable?

Basic CNNs (e.g. ResNet):

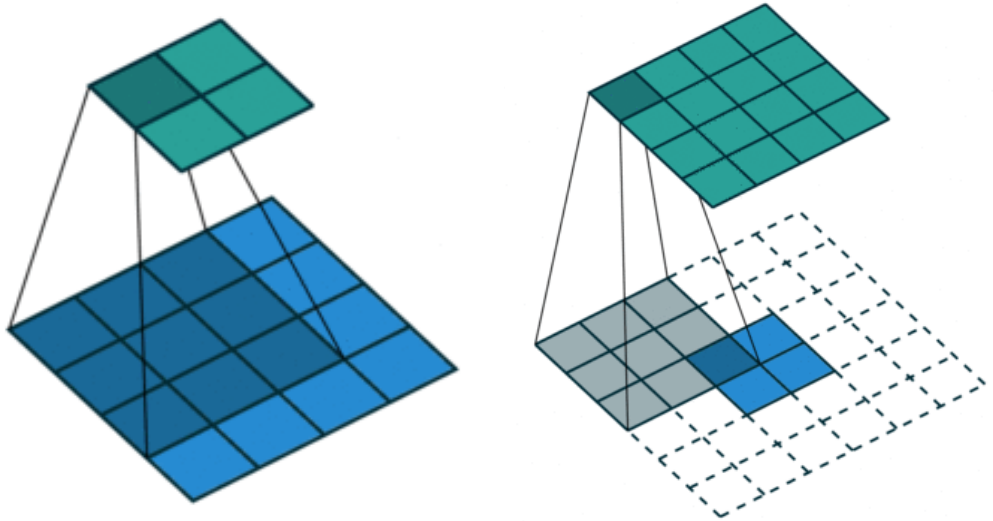
- built for image classification
  - input size does not scale with output size ⊗
  - reduces the representation size across convolutional layers
- keeping the representation size the same across layers is too costly ⊗



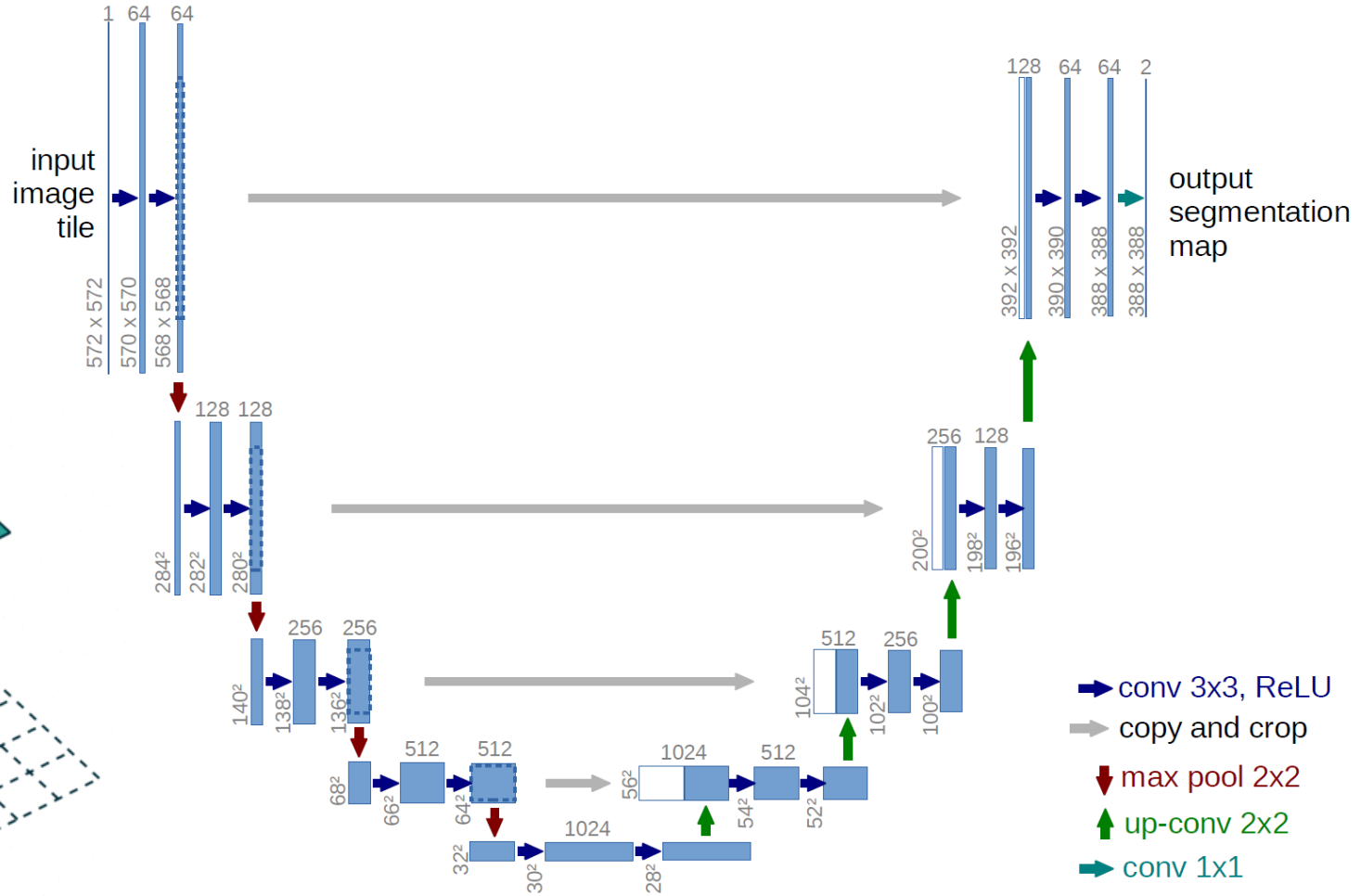
MATLAB

Early approach: U-Net [Raffenberger et al.]

- Originally for image segmentation
- Key feature is the **transposed convolution**

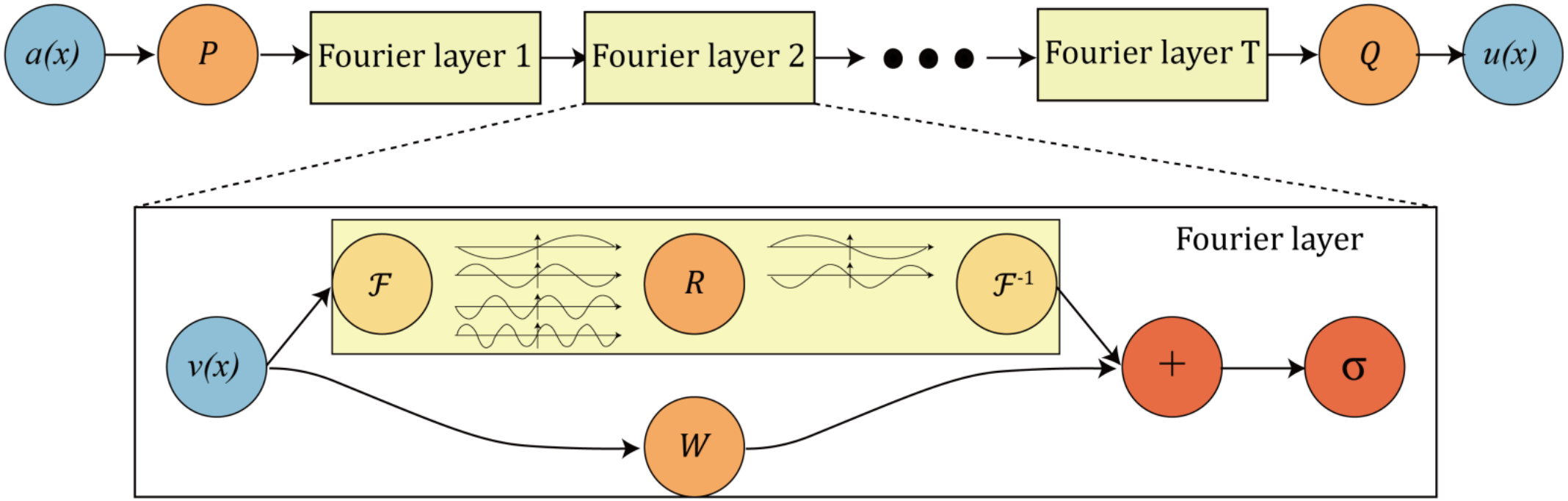


Vincent Dumoulin

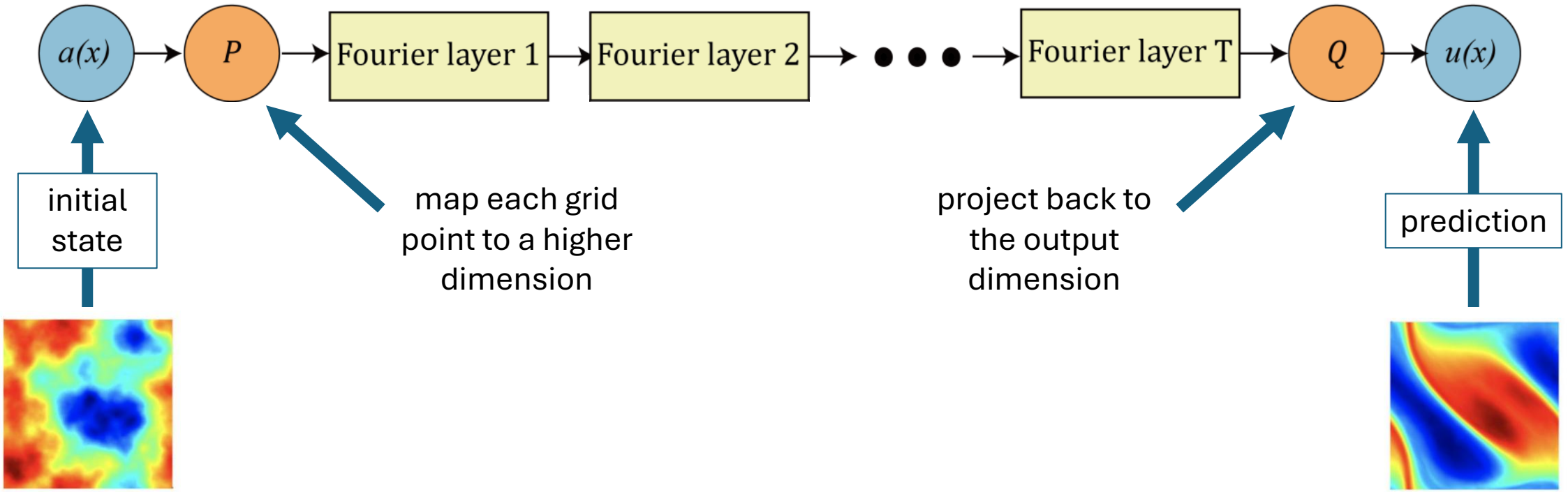


Popular choice: Fourier Neural Operator (FNO) [Li et al.]

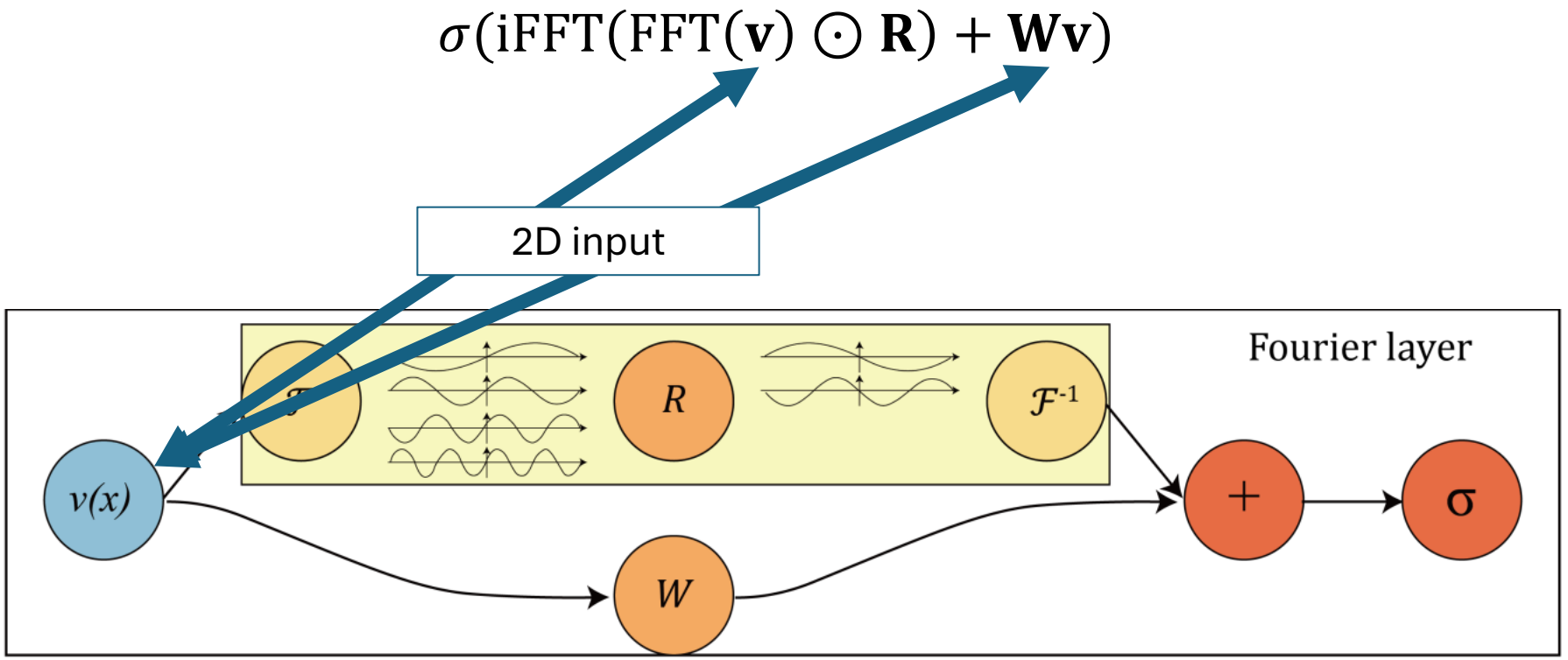
Main idea: replace convolutional layers by Fourier layers



FNO layer-by-layer



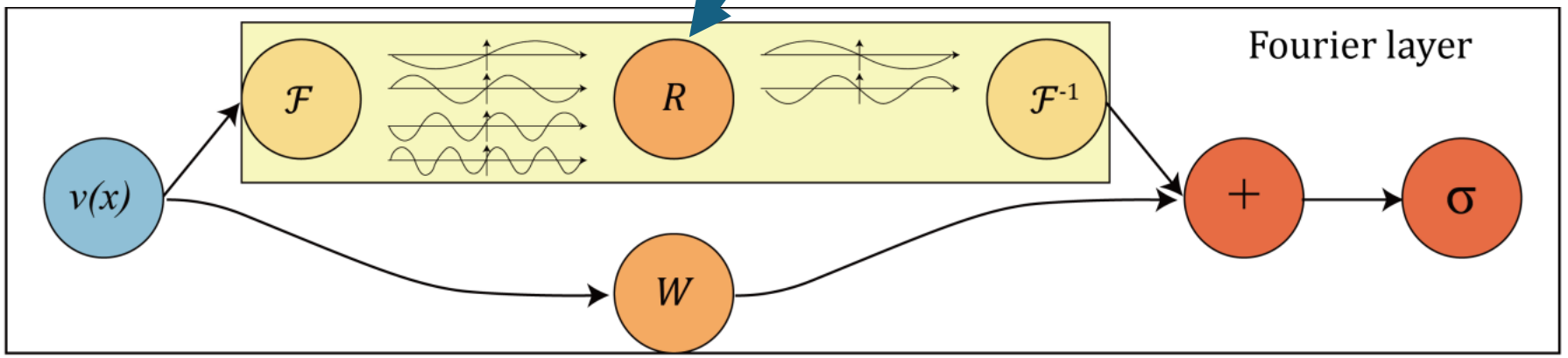
FNO layer-by-layer



FNO layer-by-layer

$$\sigma(\text{iFFT}(\text{FFT}(\mathbf{v}) \odot \mathbf{R}) + \mathbf{W}\mathbf{v})$$

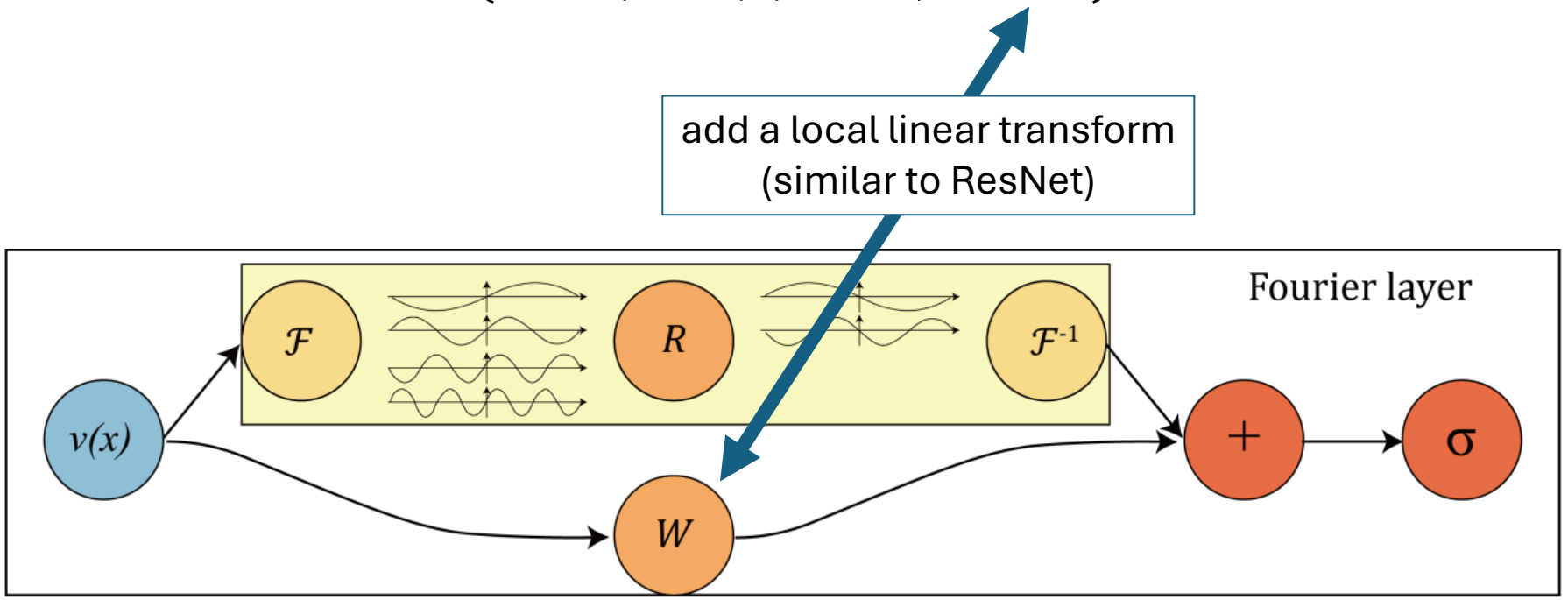
linear transform on lower modes; filter the rest



FNO layer-by-layer

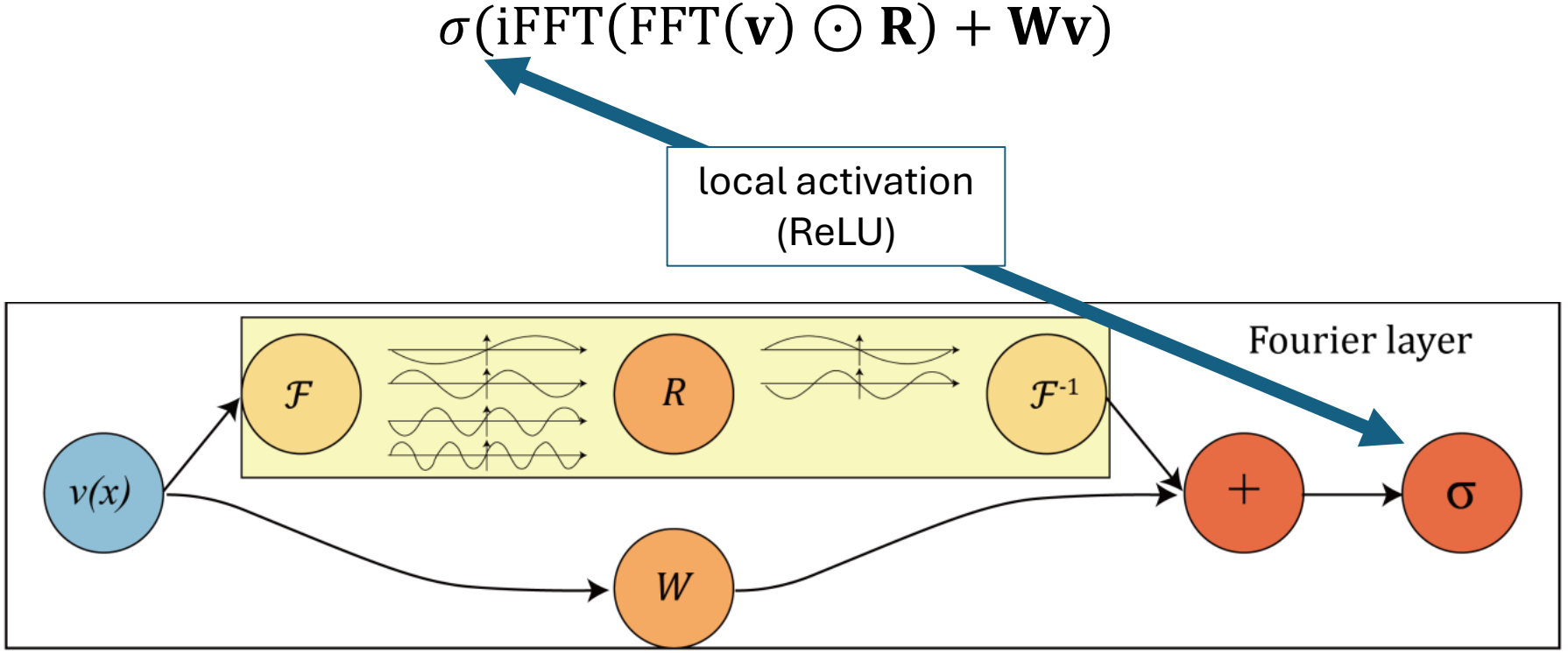
Only difference with a regular CNN:
$\sigma(\text{iFFT}(\text{FFT}(\mathbf{v}) \odot \text{FFT}(\mathbf{r})) + \mathbf{W}\mathbf{v})$

$$\sigma(\text{iFFT}(\text{FFT}(\mathbf{v}) \odot \mathbf{R}) + \mathbf{W}\mathbf{v})$$

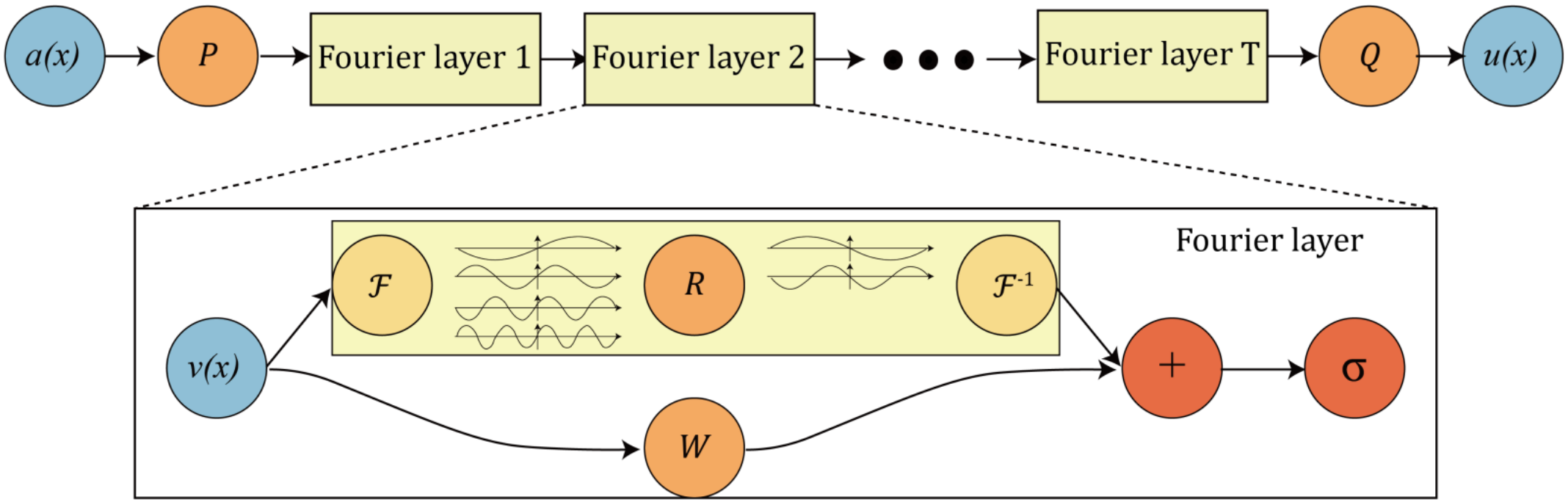


FNO layer-by-layer

$$\sigma(\text{iFFT}(\text{FFT}(\mathbf{v}) \odot \mathbf{R}) + \mathbf{W}\mathbf{v})$$

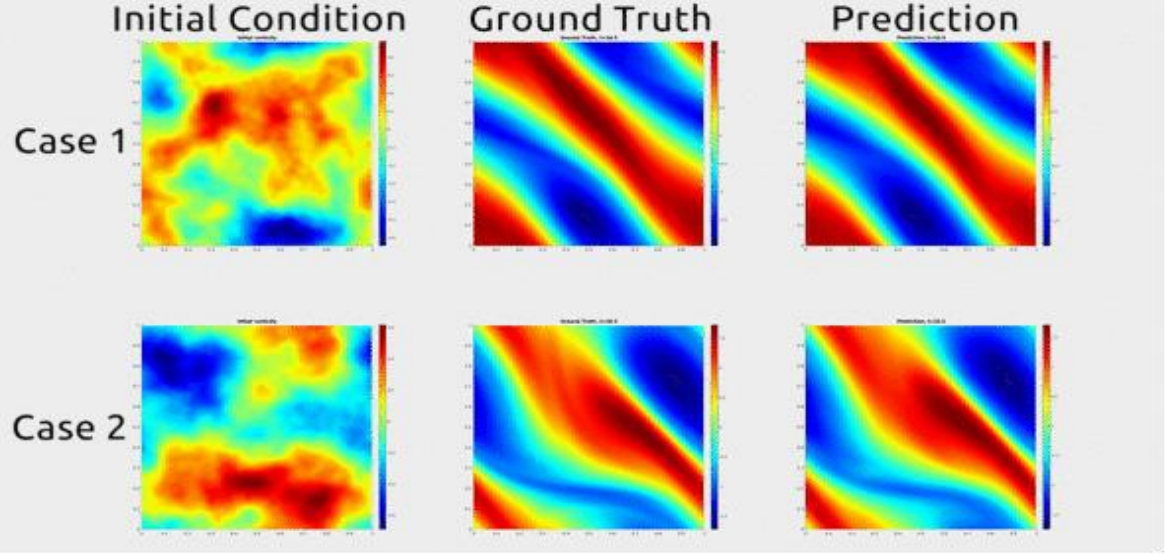


Recap: FNOs


a(x)
P
Fourier layer 1
Fourier layer 2
Fourier layer T
Q
u(x)
Fourier layer
v(x)
$\mathcal{F}$
R
$\mathcal{F}^{-1}$
W
+
σ

U-Net vs. FNO

Both are popular approaches



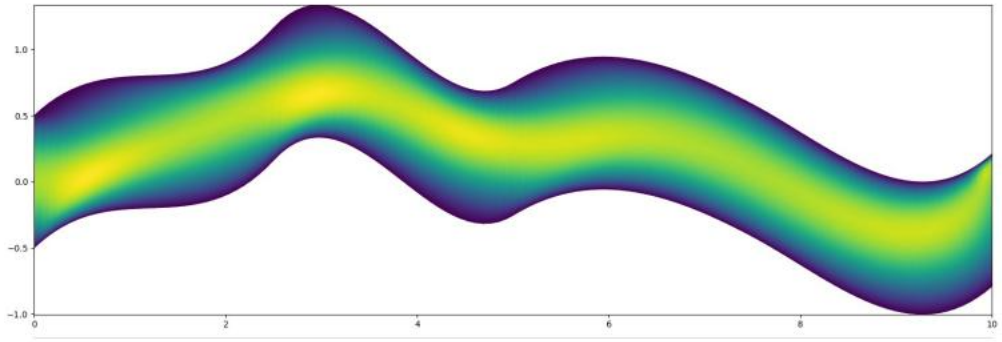
Zongyi Li

FNO has been developed on more, because

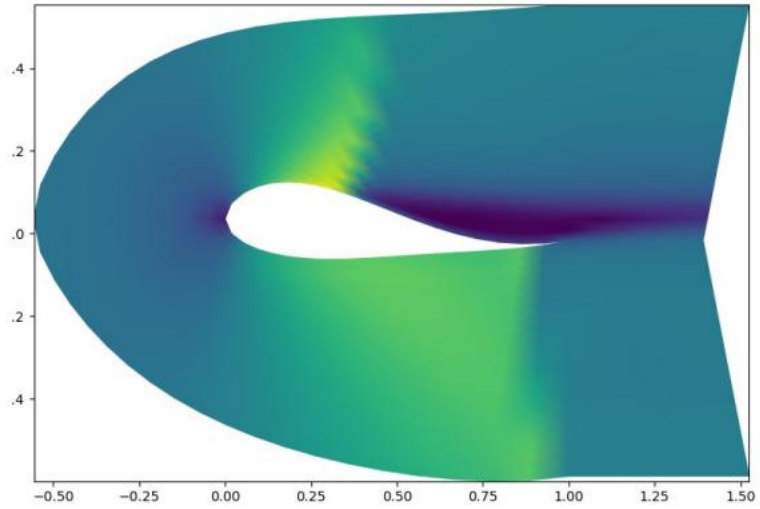
- it is claimed to be resolution invariant, enabling super resolution
- it was specifically developed for PDEs
- it does better on most (but not all) tasks in PDEBench [Takamoto et al.]

FNO variants

- PINO [Li et al.]: combines with a PINN-style equation loss
- Geo-FNO [Li et al.]: go beyond FFT to handle irregular geometry



Pipe

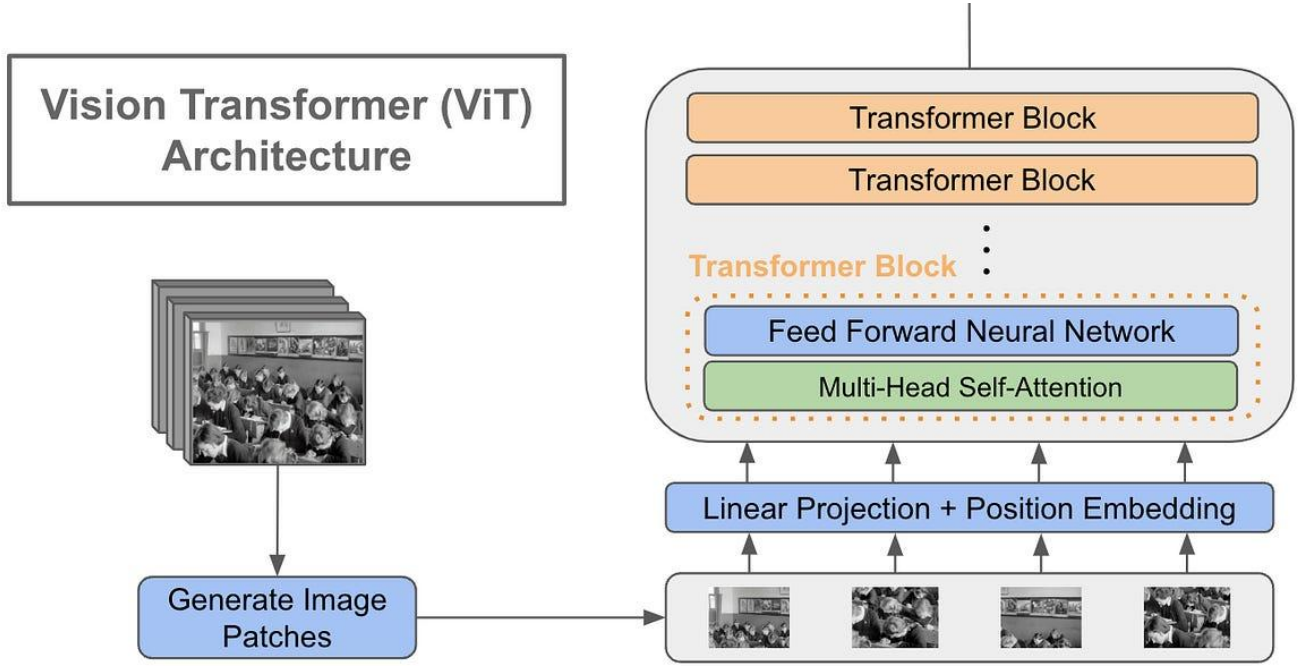


Airfoils

- FFNO [Tran et al.]: separate FFTs across dimensions for efficiency

Transformer architectures

- powerful architectures built around multi-head attention blocks
- developed for sequence tasks but in-principle can take any set of embeddings (e.g. grid points or patches of grid points) given the right positional embedding
- PDE Transformers are built around vision models such as ViT or SWIN



Outline

- Motivation
- Basic architectures
- Successes and challenges
- Ongoing directions

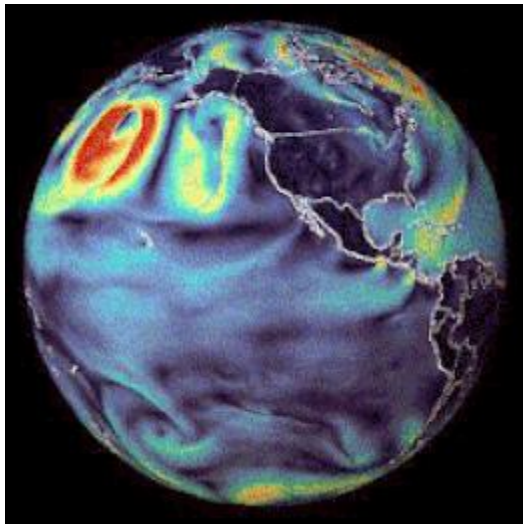
Success story: Weather

NVIDIA's FourCastNet:

- neural network trained on historical weather data
- FNO adapted to a spherical domain
- 60x faster 15-day forecast than traditional simulation

Key reasons for success:

- availability of historical data
- imperfect PDE model
- CPU-bound competitors



Challenge:
Fair comparisons to classical solvers

- If no historical data, trained models better be much faster than similar-quality classical solvers to justify the cost of generating the training data

- Unfortunately, meta-evaluations have found speedups to be overstated due to
  - comparing speedup relative to much more accurate solvers
  - using the wrong solver for the type of PDE

Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations

Nick McGreivy & Ammar Hakim

Nature Machine Intelligence volume **6**, pages 1256–1269 (2024)

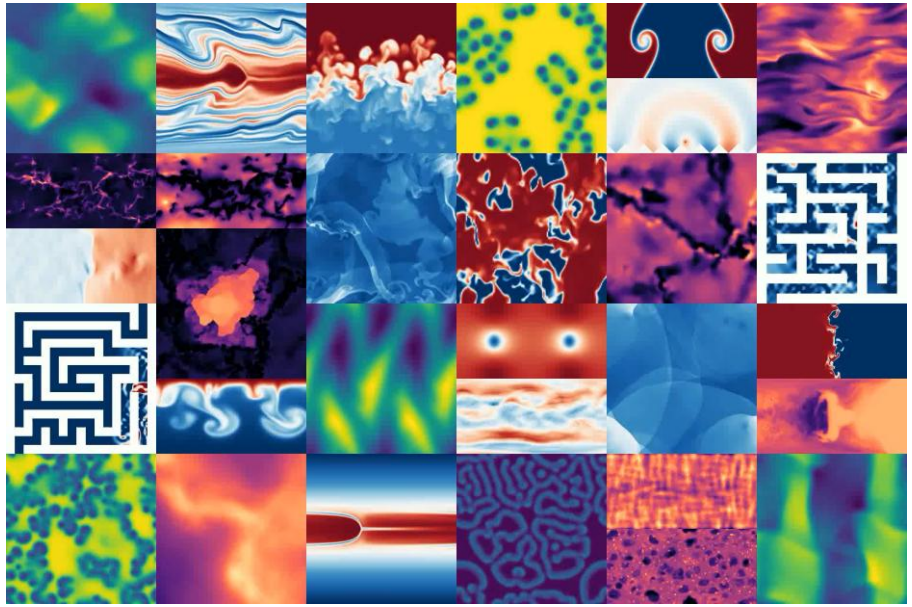
Outline

- Motivation
- Basic architectures
- Successes and challenges
- Ongoing directions

Direction: Benchmarks and evaluations

New benchmarks are constantly being developed:

1. PDEBench (2022):
   - 11 different PDEs
   - Hundreds of GB
2. The Well (2024):
   - 16 different PDEs,
   - Tens of thousands of GB

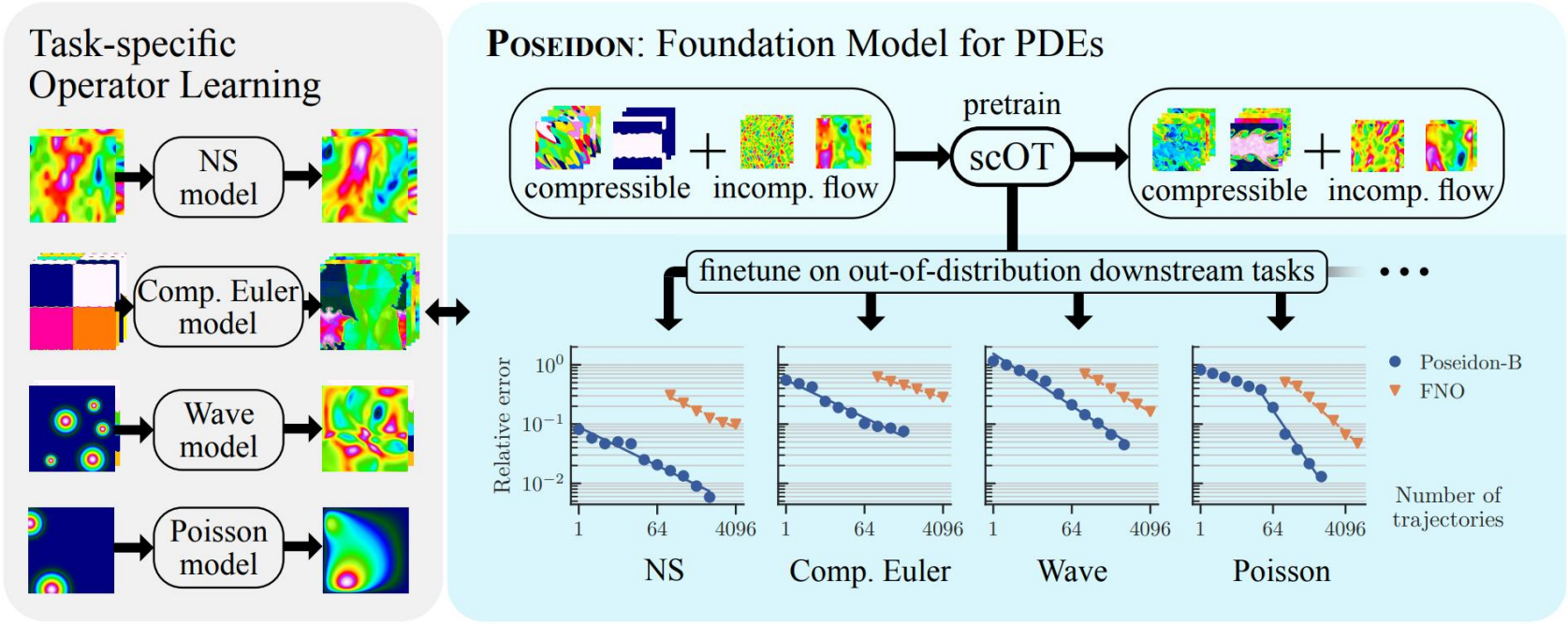


TheWell

Direction: Foundation models

Idea: reduce the need for simulation data by

1. pretraining a Transformer on tens of thousands of generic simulations
2. fine-tuning it on only a few examples from the target simulation



Direction: Theoretical understanding

Approximation theory [Marwah et al.]: what is the size of the smallest neural network needed to approximate a solution or solution operator?

Sample complexity [Boullé et al.]: how many training examples suffice to learn a neural operator to sufficient accuracy?

Results often depend on the kind of PDE (linear, elliptic, etc.).

Direction: Long-term forecasting

Neural operators are often trained auto-regressively

1. errors compound over time
2. difficult to train with long context windows
3. even on weather, neural PDE do worse on longer (subseasonal) forecasting than simple baselines that use averages of previous years' weather (even classical simulation struggles here)

This problem remains challenging, with some attempts at a solution.

Summary of neural operators for PDEs

Map forcing , initial, and boundary, conditions of to the solution.

Significant amount of architecture development, training, and benchmarking effort in the machine learning community.

Successful results in domains with significant observational data.

Skepticism from the scientific computing community due to the need for training data and underperformance relative to traditional numerical solvers.

Thanks Everyone!