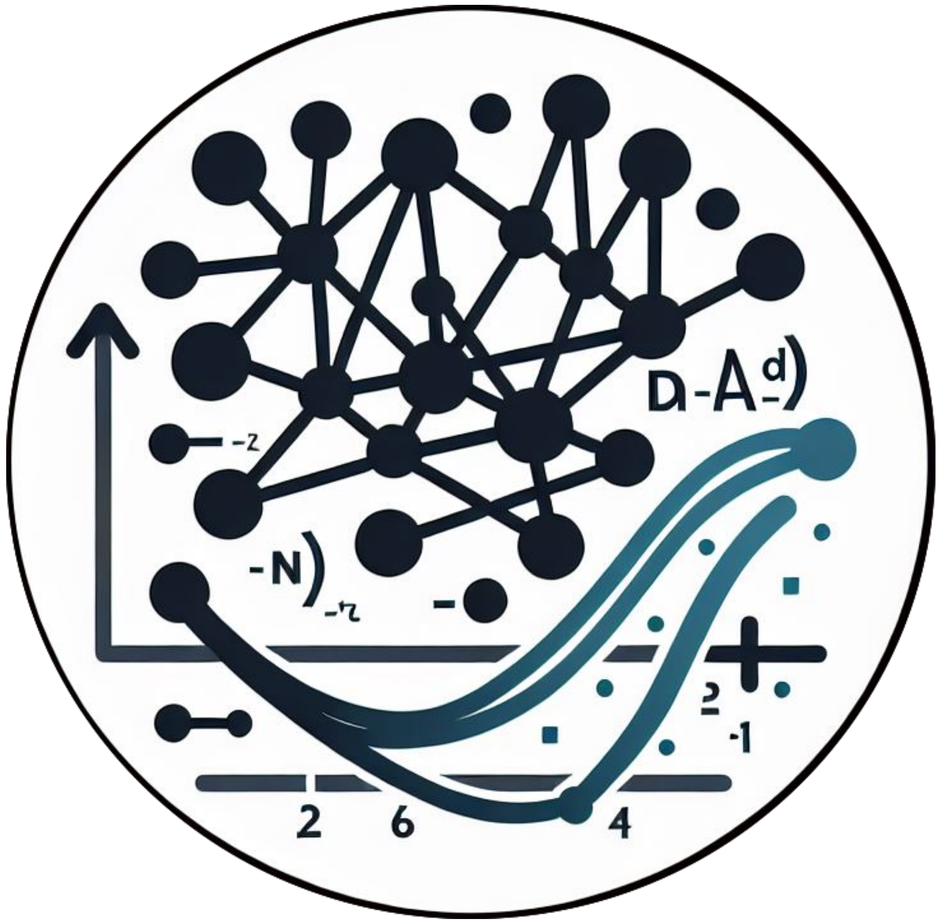
Neural Operators (FNO and DeepONet)



Deep Learning for Engineers
Andrew Ning
aning@byu.edu

FOURIER NEURAL OPERATOR FOR PARAMETRIC PARTIAL DIFFERENTIAL EQUATIONS

Zongyi Li
zongyili@caltech.edu

Nikola Kovachki
nkovachki@caltech.edu

Kamyar Azizzadenesheli
kamyar@purdue.edu

Burigede Liu
bgl@caltech.edu

Kaushik Bhattacharya
bhatta@caltech.edu

Andrew Stuart
astuart@caltech.edu

Anima Anandkumar
anima@caltech.edu

ABSTRACT

The classical development of neural networks has primarily focused on learning mappings between finite-dimensional Euclidean spaces. Recently, this has been generalized to neural operators that learn mappings between function spaces. For partial differential equations (PDEs), neural operators directly learn the mapping

input	function	output	example
x	$x \mapsto f(x)$	y	$x \mapsto x^2$

input		output	example
	function		
x	$x \mapsto f(x)$	y	$x \mapsto x^2$
f	functional $f \mapsto f(x)$	y	$f \mapsto \int_a^b f(x)dx$

input		output	example
x	function $x \mapsto f(x)$	y	$x \mapsto x^2$
f	functional $f \mapsto f(x)$	y	$f \mapsto \int_a^b f(x)dx$
f	operator $f \mapsto g$	g	$f \mapsto \nabla f$

PDE Solver

$a(x)$ $\mapsto$ $\mathcal{G}$ $\mapsto$ $u(x)$

Input function

initial/boundary conditions

operator

PDE solver

Output function

solution field

PDE Solver

$a(x) \mapsto \mathcal{G} \mapsto u(x)$

Input function	operator	Output function
initial/boundary conditions	PDE solver	solution field

---

initial temp T(x, 0)	$\frac{\partial T}{\partial t} = \alpha \nabla^2 T$	temp field T(x, t)
source current J(x, y)	Maxwell's equations	electric field E(x, y)

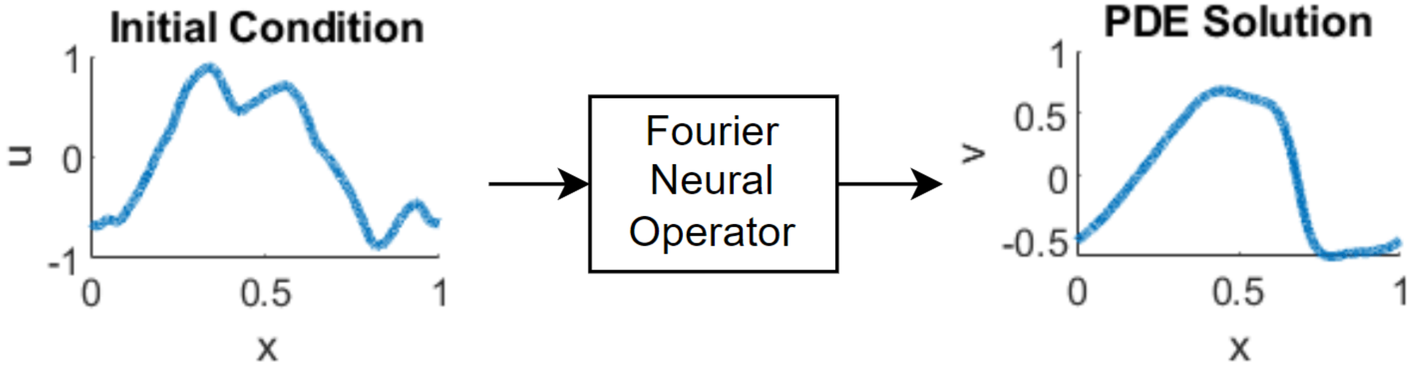
Neural Operator

$$a(x) \quad \mapsto \quad \mathcal{G} \quad \mapsto \quad u(x)$$

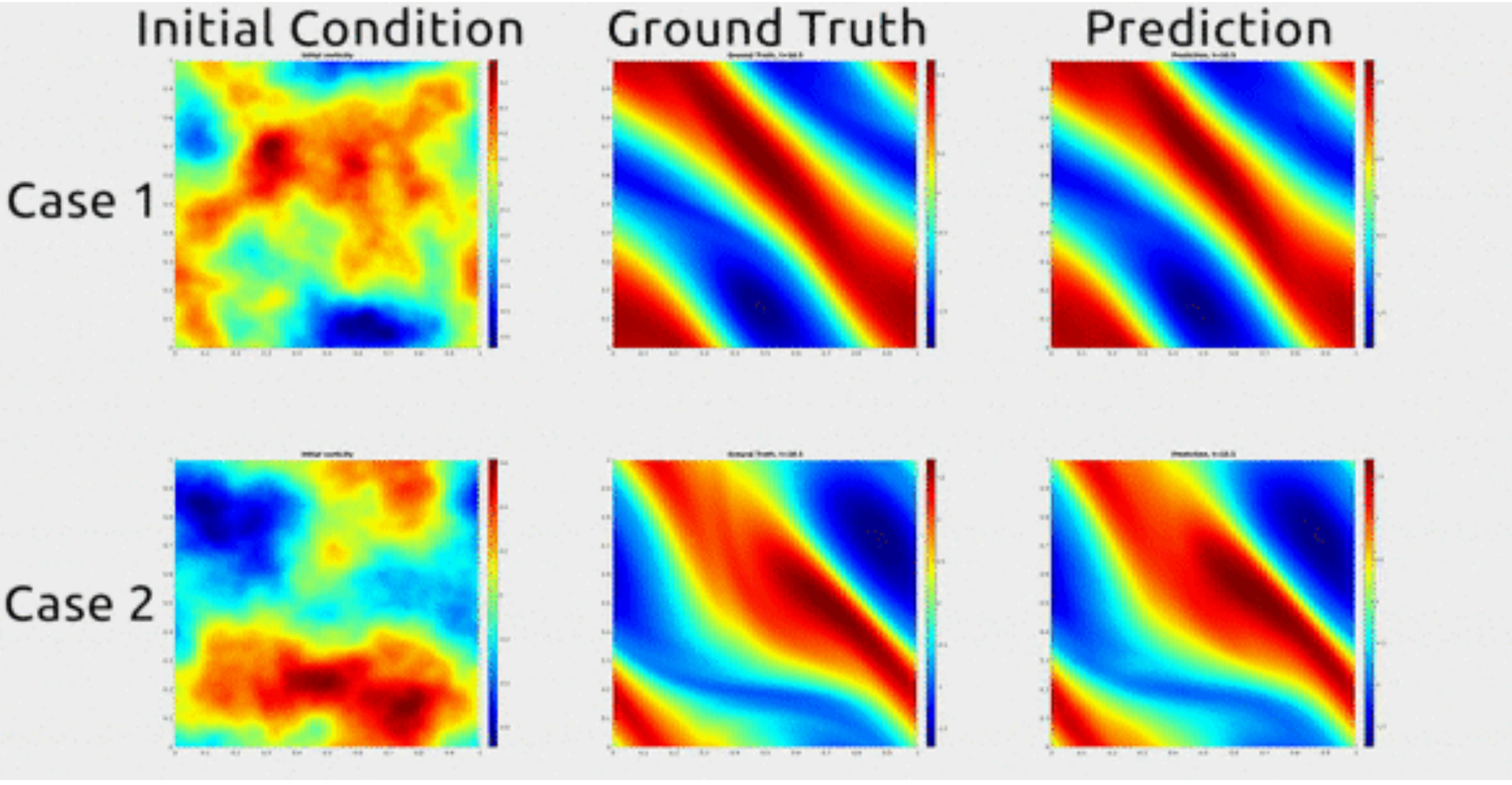
neural net learns operator $\mathcal{G}$

train with paired data: $\{a(x)^{(i)}, u(x)^{(i)}\}$

can be evaluated at any resolution (not tied to grid used in training)



mathworks


Initial Condition
Ground Truth
Prediction
Case 1
Case 2

Linear Boundary Value Problems

$$\mathscr{L}u(x) = f(x)$$

$$\mathscr{L} = a(x)\frac{d^2}{dx^2} + b(x)\frac{d}{dx} + c(x) \qquad \text{(for example)}$$

Linear Boundary Value Problems

$$\mathscr{L}u(x) = f(x)$$

$$\mathscr{L} = a(x)\frac{d^2}{dx^2} + b(x)\frac{d}{dx} + c(x) \qquad \text{(for example)}$$

$$\mathscr{L}G(x,y) = \delta(x-y)$$

Green's function

Dirac delta function

$$u(x) = \int G(x,y)f(y)dy$$

Moving towards a neural net approach

$$u(x) = \int G(x, y) f(y) dy$$ (linear PDEs)

Moving towards a neural net approach

$$u(x) = \int G(x, y) f(y) dy \qquad \text{(linear PDEs)}$$

for nonlinear PDEs we use iterative approaches

$$v_0 \mapsto v_1 \mapsto v_2 \mapsto \ldots \mapsto v_n$$

$a(x)$ $u(x)$

Moving towards a neural net approach

$$u(x) = \int G(x, y) f(y) dy \qquad \text{(linear PDEs)}$$

for nonlinear PDEs we use iterative approaches

$$v_0 \mapsto v_1 \mapsto v_2 \mapsto \ldots \mapsto v_n$$

$a(x)$ $\qquad\qquad\qquad\qquad\qquad\qquad$ $u(x)$

neural network

$$v_{t+1}(x) = \int K_\theta(x, y) v_t(y) dy$$

Moving towards a neural net approach

$$v_{t+1}(x) = \int K_\theta(x, y) v_t(y) dy$$

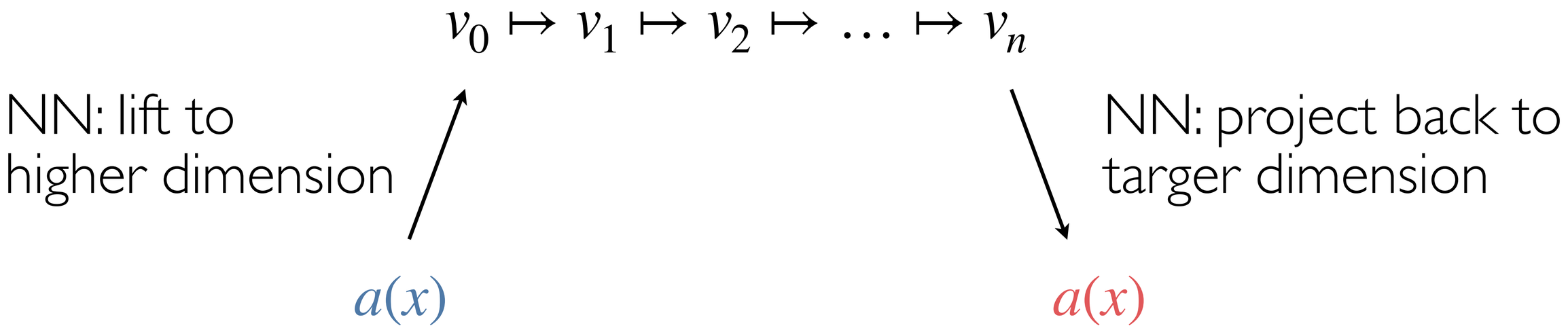
Moving towards a neural net approach

$$v_{t+1}(x) = \int K_\theta(x, y) v_t(y) dy$$

$$v_{t+1}(x) = \sigma\left(W v_t(x) + \int_D K_\theta(x, y) v_t(y) dy \right)$$

activation function

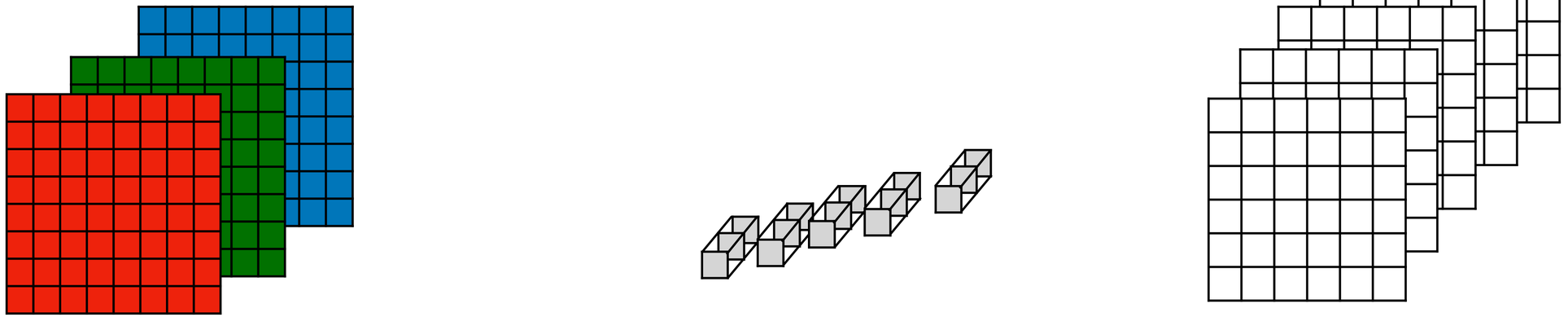
pointwise linear transform



NOTE: not an autoencoder

1x1 convolutions to lift in *channel dimension*

1x1 convolution



8 x 8 x 3

1 x 1 x 3 x 5

8 x 8 x 5

in channels

kernel size

out channels

```
nn.Conv2d(in_channels=3, out_channels=5, kernel_size=1)
```

Fourier Transform

$$\int K_\theta(x, y) v_t(y) dy$$

$$\int_D K_\theta(x - y) v_t(y) dy \qquad \text{(assumption)}$$

Fourier Transform

$$\int K_\theta(x, y) v_t(y) dy$$

$$\int_D K_\theta(x - y) v_t(y) dy \qquad \text{(assumption)}$$

$$= (K_\theta * v_t) \qquad \text{(convolution integral)}$$

$$= \mathcal{F}^{-1}\left(\mathcal{F}(K_\theta) \cdot \mathcal{F}(v_t)\right)$$

Fourier Transform

$$\int K_\theta(x, y) v_t(y) dy$$

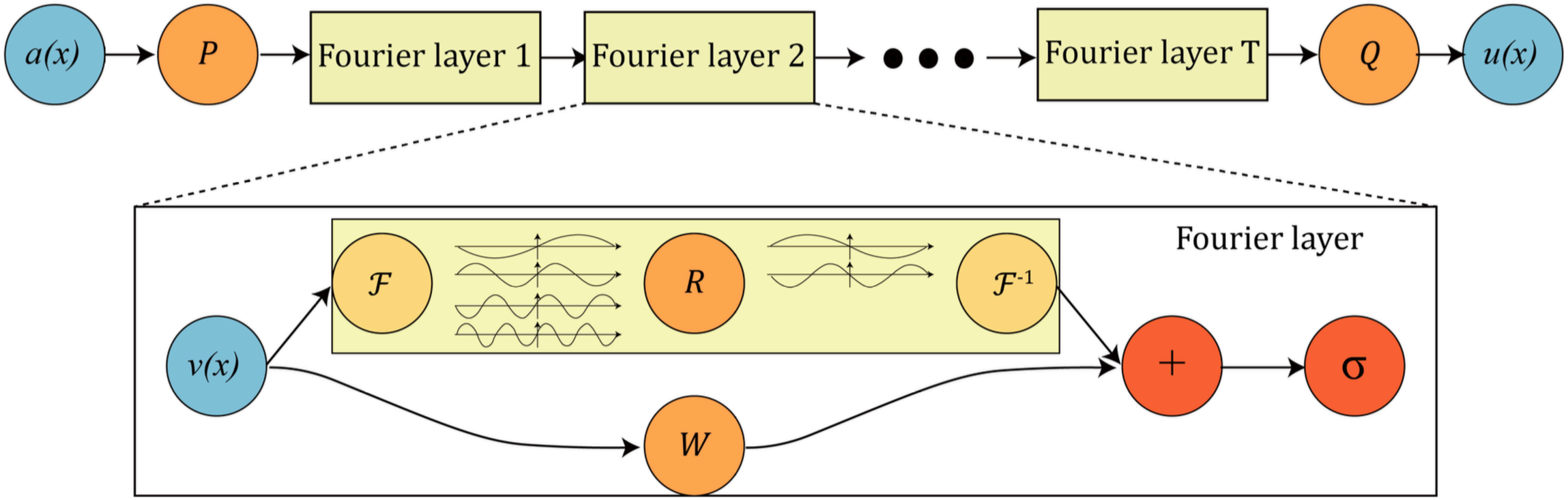
$$\int_D K_\theta(x - y) v_t(y) dy \qquad \text{(assumption)}$$

$$= (K_\theta * v_t) \qquad \text{(convolution integral)}$$

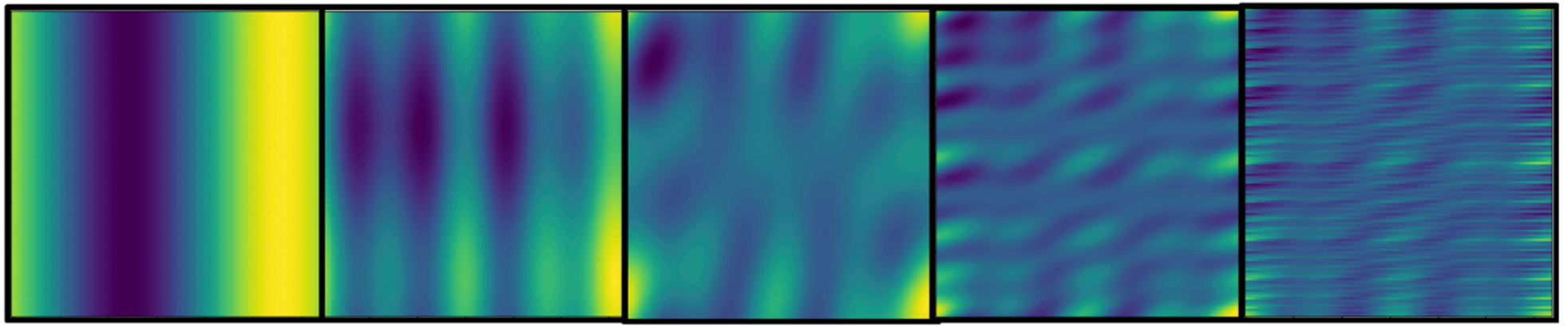
$$= \mathcal{F}^{-1}\left(\mathcal{F}(K_\theta) \cdot \mathcal{F}(v_t)\right)$$

$$= \mathcal{F}^{-1}\left(R_\theta \cdot \mathcal{F}(v_t)\right)$$

Summary


a(x)
P
Fourier layer 1
Fourier layer 2
Fourier layer T
Q
u(x)
Fourier layer
v(x)
W
+
σ

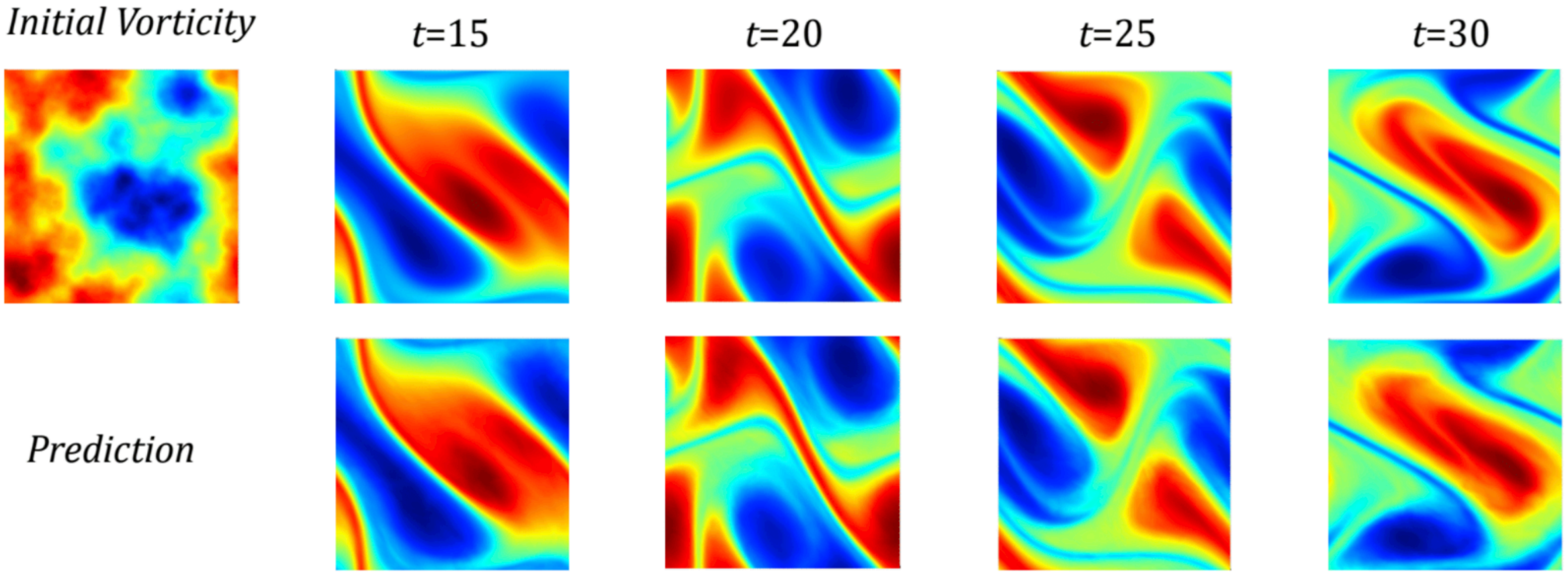
Filters in CNN



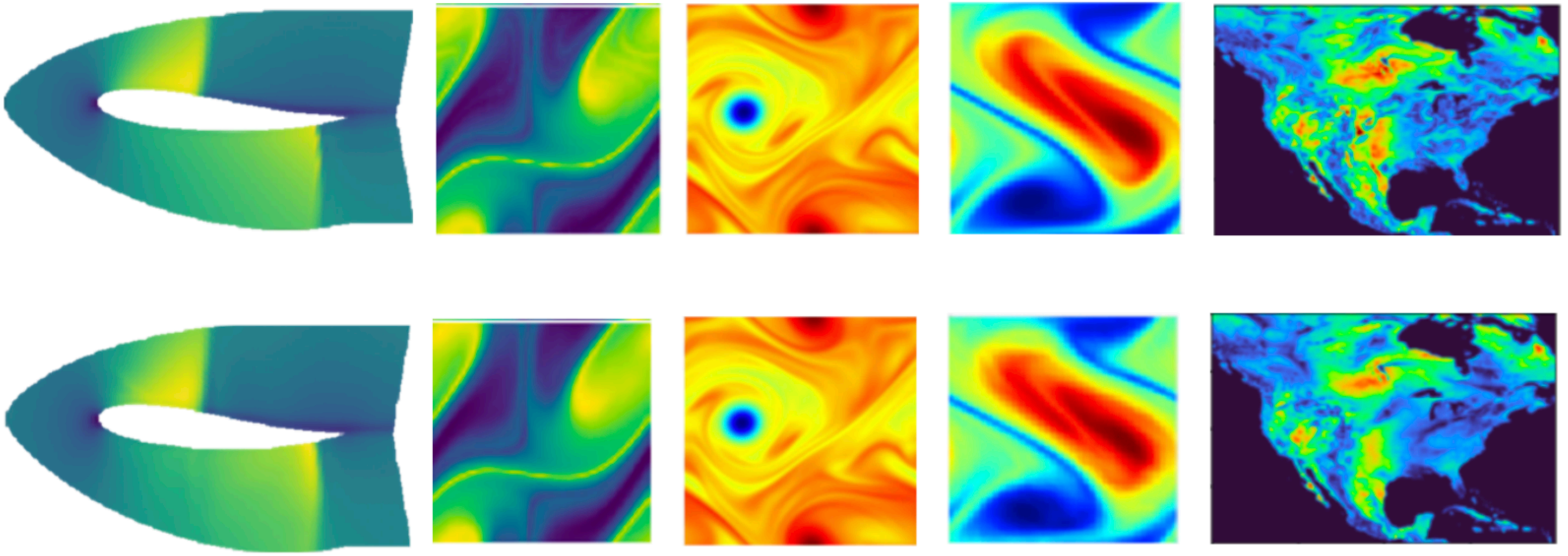
Fourier Filters

https://zongyi-li.github.io/blog/2020/fourier-pde/

Zero-shot Super Resolution



Many Other Variants



Airfoil Flow (Geo-FNO) | Darcy Flow (GANO) | Kolmogorov Flow (PINO) | Navier Stokes (FNO) | Weather model (FourcastNet)

DeepONet

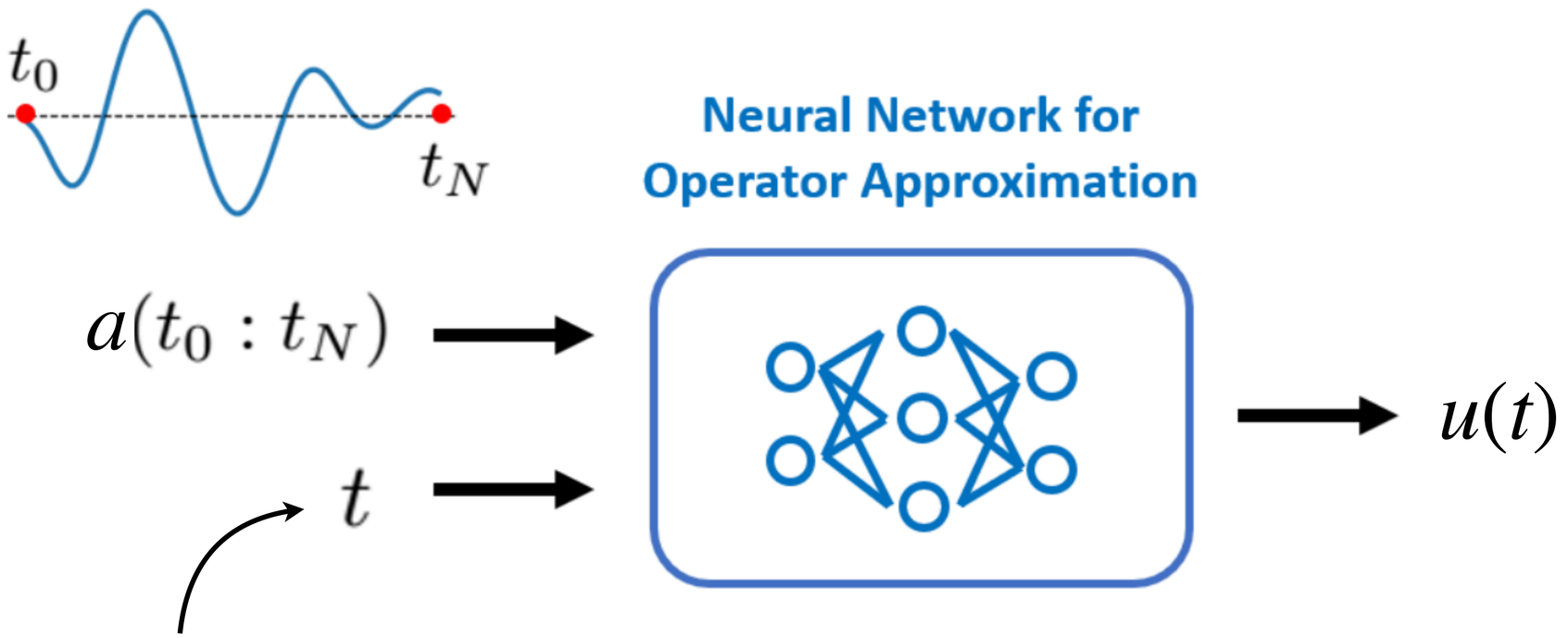
DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators

Lu Lu[1], Pengzhan Jin[2], and George Em Karniadakis[1]

[1]Division of Applied Mathematics, Brown University, Providence, RI 02912, USA
[2]LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

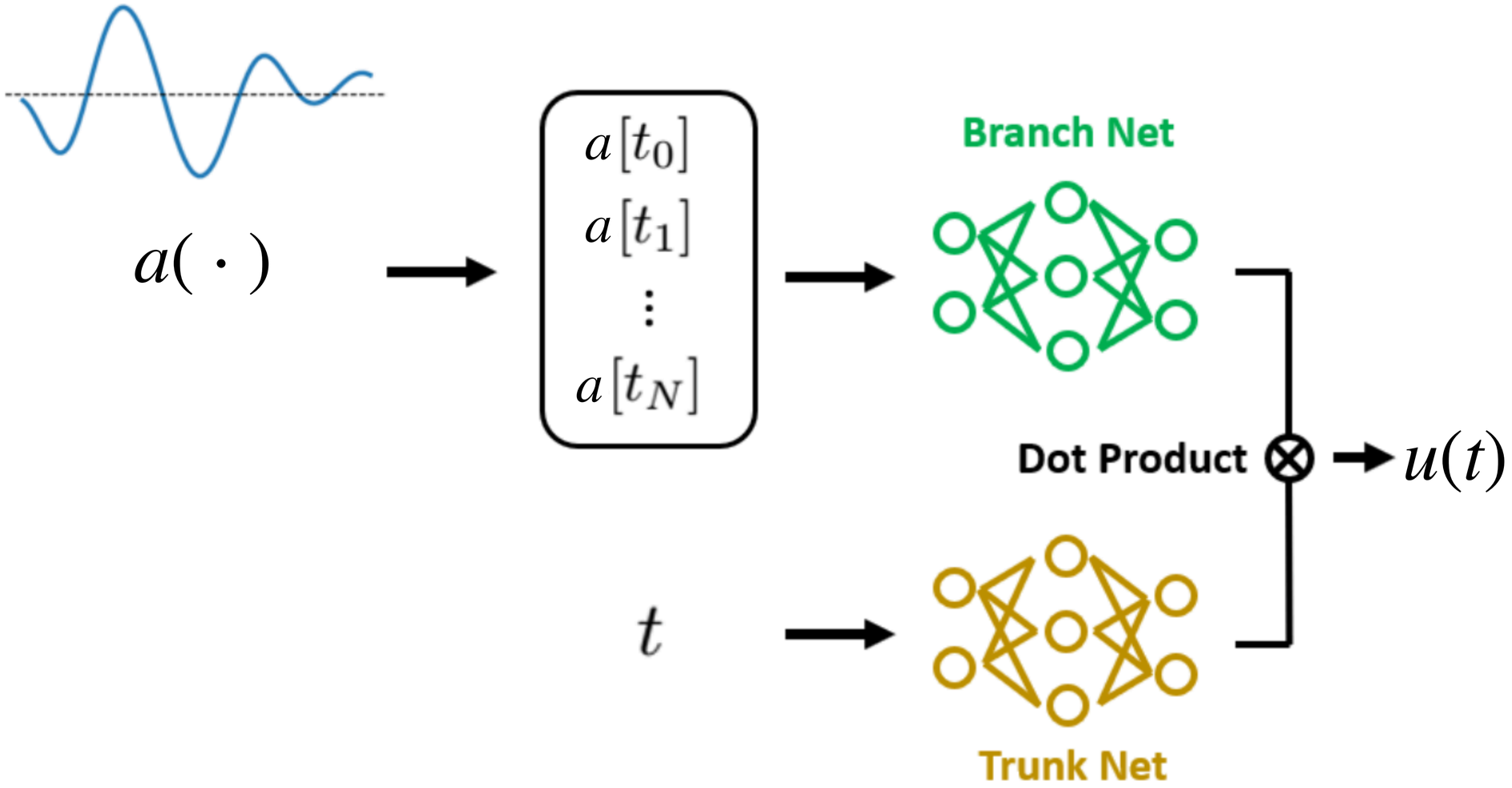
Abstract

While it is widely known that neural networks are universal approximators of continuous functions, a less known and perhaps more powerful result is that a neural network with a single hidden layer can



Pointwise prediction (like PINN)

figure adapted from Shuai Guo, Operator Learning via Physics-Informed DeepONet



Shuai Guo, Operator Learning via Physics-Informed DeepONet