# Aprendizaje profundo basado en la física

## Semana 8: Operadores Neuronales

**Docente: José I. Robledo - 25/05/2026**

# Operadores Neuronales

## Introducción

- Representan una familia de modelos de aprendizaje profundo diseñados para aprender **operadores** en lugar de funciones convencionales.

- Mientras que una red neuronal clásica aprende una relación:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Un operador neuronal aprende una relación entre funciones

$$\mathcal{G} : a(x) \rightarrow u(x)$$

# Operadores Neuronales

## Problemas de las NNs

*En lugar de resolver explícitamente las ecuaciones para cada nuevo escenario, las soluciones basadas en datos se entrenan con datos que emparejan problemas con sus soluciones.*

- Dependencia de la resolución y la calidad de los datos de entrada
- Costo agregado de resolver las PDEs repetidamente para generar los datos
- Mala generalización entre mallas. Si se entrenó para una resolución, capaz que no funciona bien para otra.
- Dificultad para capturar dinámicas continuas

# Operadores Neuronales

## Idea central

$$\mathcal{G} : a(x) \rightarrow u(x)$$

- $a(x)$: coeficientes, condiciones iniciales, términos de fuerzas externas, etc.
- $u(x)$: solución de la PDE

**Ventajas**

- Inferencia extremadamente rápida
- aprenden relaciones entre funciones y generalicen a través de diferentes discretizaciones
- podemos usar métodos numéricos para generar datos menos precisos y de baja resolución, y aun así el algoritmo entrenado puede producir predicciones precisas y de alta resolución
- tanto el entrenamiento como la evaluación se vuelven significativamente más eficientes y sencillos.

# Ejemplos

## Navier Stokes

$$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \Delta u$$

$$\mathcal{G} : u_0(x) \mapsto u(x, t)$$

## Poisson

$$-\nabla \cdot (a(x) \nabla u(x)) = f(x)$$

- $a(x)$: conductividad/permeabilidad espacial

- $u(x)$: temperatura o potencial

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

## Elasticidad Lineal

$$-\nabla \cdot (\mathbf{C}(x) : \nabla^s u) = f$$

- $\mathbf{C}(x)$ = tensor de elasticidad espacial

- $u(x)$ = desplazamiento

$$\mathcal{G} : \mathbf{C}(x) \mapsto u(x)$$

# Operadores Neuronales

## Teoría

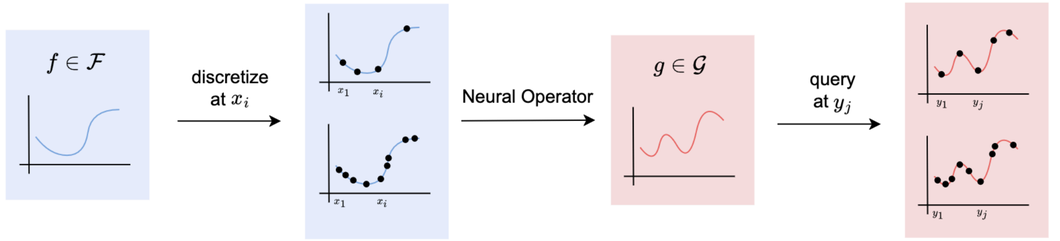



Figure 1: **Illustration of a neural operator.** The input is a function $f \in \mathcal{F}$ that can be given at different discretizations $(x_i)_{i=1}^n$. The output is a function $g \in \mathcal{G}$ that can be queried at different points $(y_j)_{j=1}^m$.

$$\mathcal{G} : a(x) \to u(x)$$

El término “operador” se refiere al mapeo entre dos espacios de funciones.
Ejemplos: diferenciación/integración

$$\mathcal{L}u = f$$

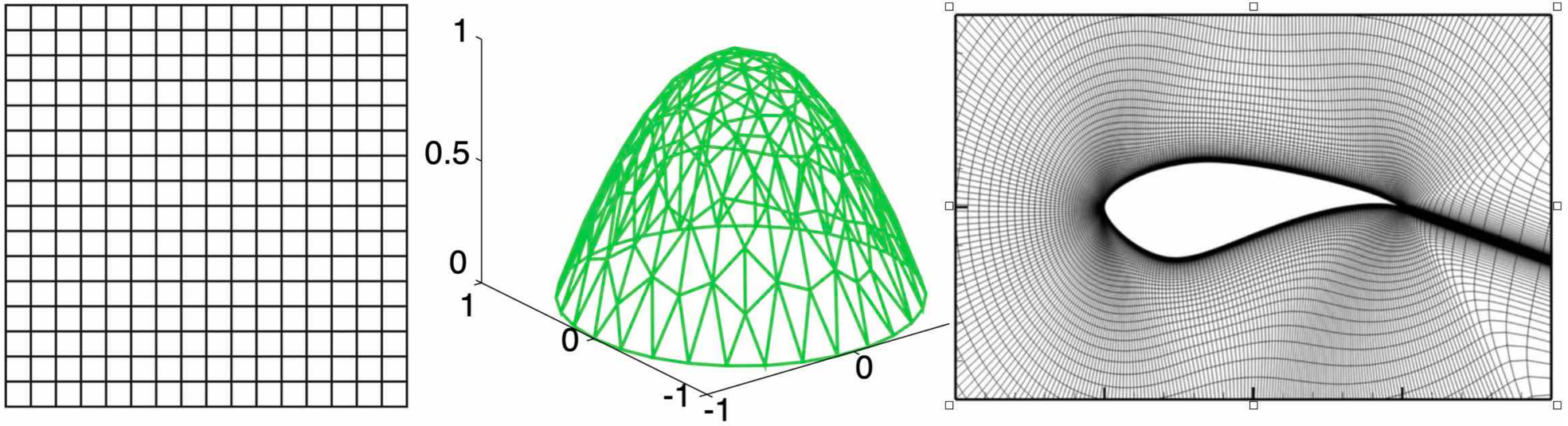
Se suele definir $\mathcal{L}$ y $f$, y se busca $u$ como solución. Se puede pensar que resolver una PDE en definitiva es una tarea de aprender un operador.
**Buscamos aprender el mapa entre las funciones de entrada a las funciones soluciones.**

$$(a_i(x), u_i(x)) \to u = \mathcal{G}(a)$$

https://arxiv.org/pdf/2506.10973

# Operadores Neuronales

## Discretización




Buscamos una solución que no esté atada a ninguna discretización en particular, i.e. una formulación continua del problema.

# Operadores Neuronales

## DeepONets

- El teorema de aproximación universal establece que las redes neuronales pueden utilizarse para aproximar cualquier función continua con una precisión arbitraria si no se impone ninguna restricción sobre el ancho y la profundidad de las capas ocultas.

- Más aún: **Una red neuronal con una sola capa oculta puede aproximar con precisión cualquier funcional continuo no lineal.**

- Sea $G(u)$ la función de salida de aplicar el operador $G$ a la función de entrada $u$. Para cualquier puto $y$ en el dominio de $G(u)$, $G(u)(y)$ es un escalar. Si usamos una red neuronal para aprender esta transformación, necesitamos dos entradas.

https://arxiv.org/pdf/1910.03193

# Operadores Neuronales

## DeepONets

- Representamos la función de entrada de manera discreta en ciertos puntos: $\{x_i\}_{i=1,\dots,N}$ llamados *sensores*.

**Theorem 1 (Universal Approximation Theorem for Operator).** *Suppose that $\sigma$ is a continuous non-polynomial function, $X$ is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in $X$ and $\mathbb{R}^d$, respectively, $V$ is a compact set in $C(K_1)$, $G$ is a nonlinear continuous operator, which maps $V$ into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers $n$, $p$, $m$, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \dots, n$, $k = 1, \dots, p$, $j = 1, \dots, m$, such that*

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma \left( \sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} \right| < \epsilon \tag{1}$$

*holds for all $u \in V$ and $y \in K_2$.*

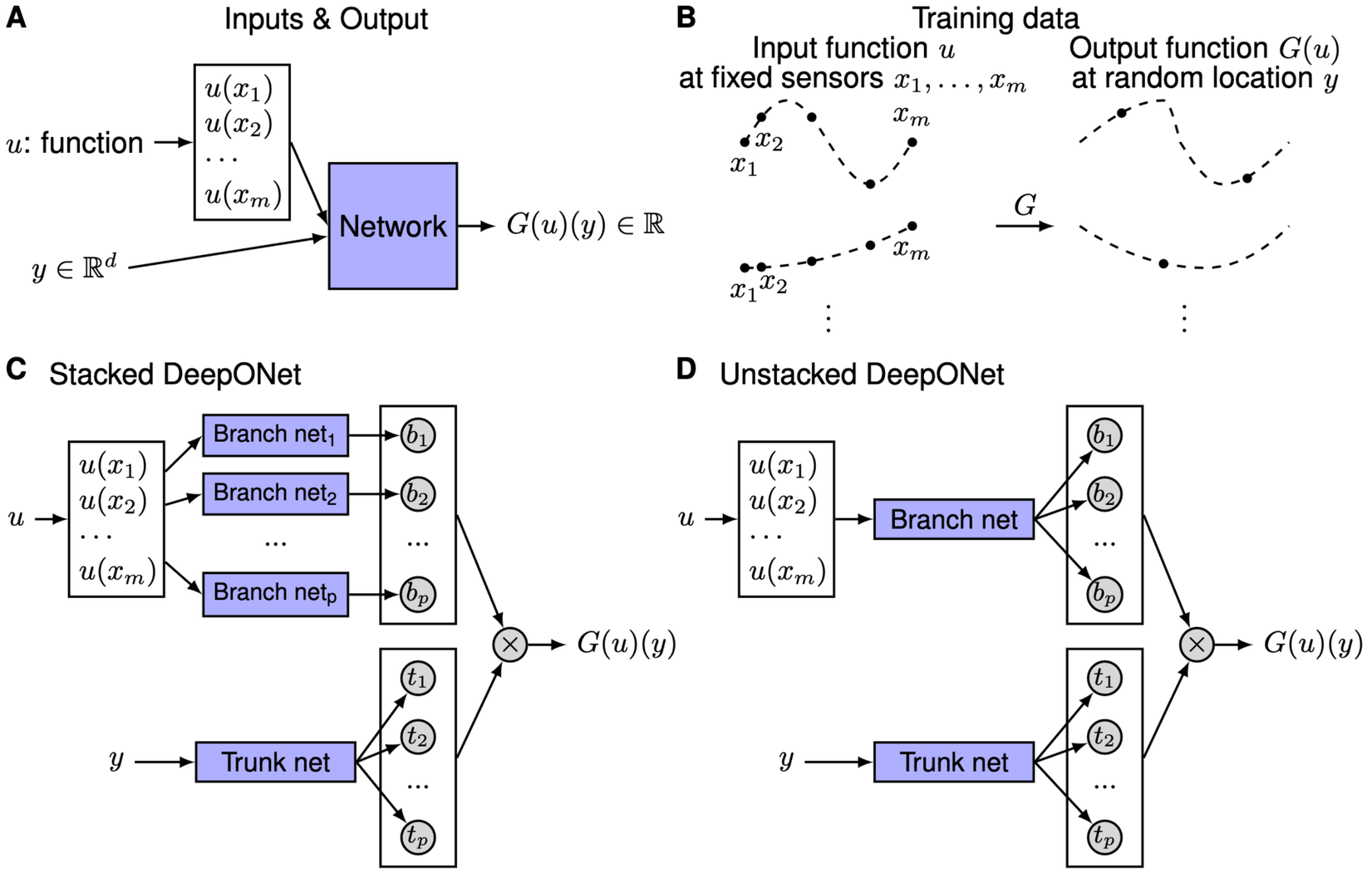
*No nos informa cómo aprender efectivamente los operadores*

https://arxiv.org/pdf/1910.03193

# Operadores Neuronales

## DeepONets

$$G(u)(y) \approx \sum_{k=1}^{p} b_k t_k$$




**Notebook Operadores Neuronales**

https://arxiv.org/pdf/1910.03193

# Operadores Neuronales

## Fourier Neural Operators (FNOs)

Si definimos $\hat{u}(k) = \mathscr{F}(u)(k)$, entonces la diferenciación se transforma en multiplicación

$$\mathscr{F}\left(\frac{d}{dx}u\right) = ik\hat{u}(k).$$

Por lo tanto la ecuación diferencial cambia

$$-\Delta u = f \implies |k|^2 \hat{u}(k) = \hat{f}(k) \implies \hat{u}(k) = \frac{1}{|k|^2}\hat{f}(k)$$

$\hat{u}(k) = R(k)\hat{f}(k)$ Cada modo espectral evoluciona independientemente

https://arxiv.org/pdf/2010.08895

# Fourier Neural Operators (FNOs)

## Formulación kernel

Para una PDE general de la forma $(\mathscr{L}_a u)(x) = f(x),\ \ x \in D$, con $u(x) = 0$ si $x \in \partial D$, podemos definir la función de Green $G : D \times D \to \mathbb{R}$ como la única solución al problema

$$\mathscr{L}_a G(x, \cdot\ ) = \delta_x$$

El operador verdadero $\mathscr{F}_{true}$ se puede escribir como un operador integral de la función de Green

$$u(x) = \int_D G_a(x, y) f(y) dy$$

Generalmente la función de Green es continua ($\forall x \neq y$), por lo que resulta natural modelarla con una red neuronal kernel $\kappa$, tal que

$$u(x) = \int_D \kappa_\phi(x, y, a(x), a(y)) f(y) dy$$

# Fourier Neural Operators (FNOs)

## Formulación kernel

$f$ es una función desconocida pero fija. Vamos a formularlo como un solver iterativo que aproxima $u$ mediante $u_t$, con $t = 0, \ldots, T$

Inicializamos con $u_0(x) = (x, a(x))$. En cada paso temporal

$$u_{t+1}(x) = \int_D \kappa_\phi(x, y, a(x), a(y)) u_t(y) dy$$

Más aún, llevaremos $u(x) \in \mathbb{R}^d$ a un espacio de mayor dimension $v(x) \in \mathbb{R}^n$ y escribimos

$$v_0(x) = NN_1(x, a(x))$$

$$v_{t+1}(x) = \sigma(W v_t(x) + \int_{B(x,r)} \kappa_\phi(x, y, a(x), a(y)) v_t(y) dy) = \sigma(W v_t(x) + (\mathcal{K}_{\phi,a} v_t)(x))$$

$$u(x) = NN_2(v_T(x))$$

$$\kappa_\phi(x, y, a(x), a(y)) \in \mathbb{R}^{n \times n}$$

# Fourier Neural Operators (FNOs)

## Formulación kernel

En FNO, remplazamos el operador kernel integral por un operador convolución definido en el espacio de Fourier

$$(\mathscr{F}f)_j(k) = \int_D f_j(x) e^{-2i\pi <x,k>} dx.$$

Si $\kappa_\phi(x, y, a(x), a(y)) = k_\phi(x - y)$ y aplicando el teorema de la convolución, obtenemos

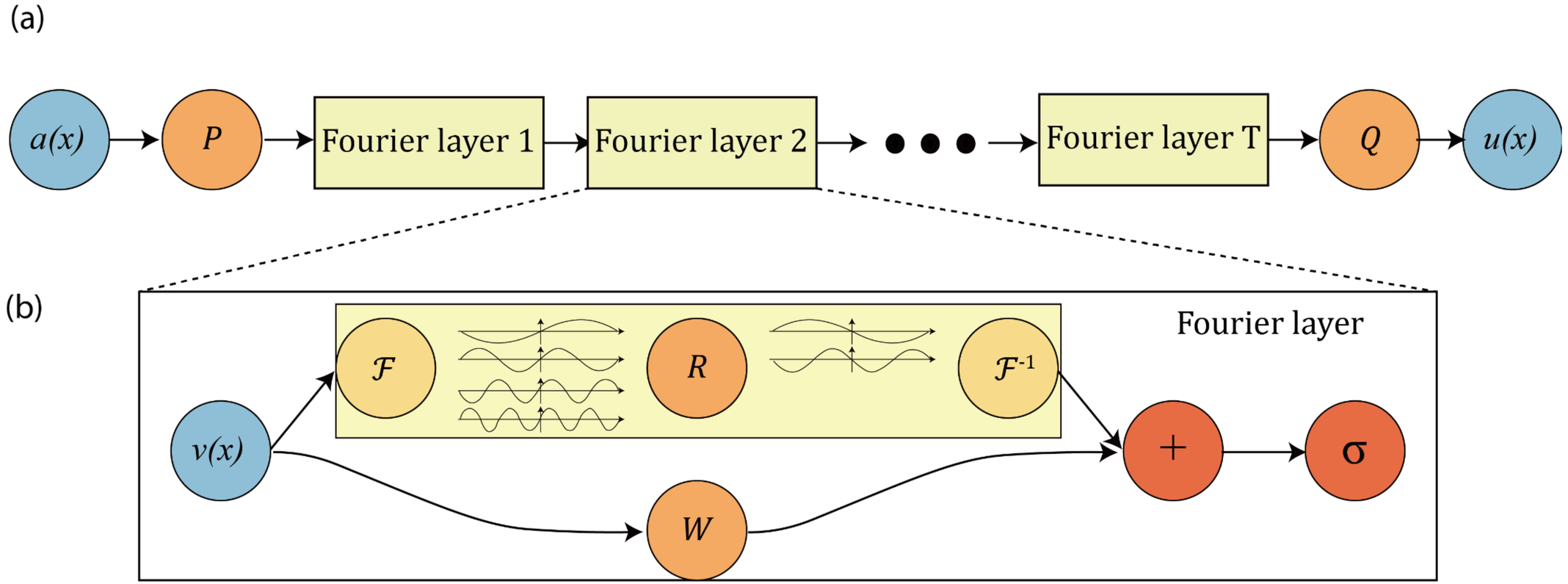
$$(\mathscr{K}_{\phi,a} v_t)(x) = \mathscr{F}^{-1}(\mathscr{F}(\kappa_\phi) \cdot \mathscr{F}(v_t))(x), \ \forall x \in D$$

Podemos parametrizar directamente $\kappa_\phi$ en el espacio de Fourier, i.e. $\mathscr{F}(\kappa_\phi) = R_\phi$

https://arxiv.org/pdf/2010.08895

# Operadores Neuronales

## Fourier Neural Operators (FNOs)




**Notebook Operadores Neuronales**

https://arxiv.org/pdf/2010.08895

# Operadores Neuronales

## Algunas referencias

- Physics-Informed Neural Operator for Learning Partial Differential Equations https://dl.acm.org/doi/full/10.1145/3648506
- Physics-Guided Deep Learning for Dynamical Systems https://qiniu.pattern.swarma.org/pdf/arxiv/2107.01272.pdf
- Mechanical Characterization and Inverse Design of Stochastic Architected Metamaterials Using Neural Operators https://arxiv.org/pdf/2311.13812