FNO AND ITS PROVENANCE

Constraining Predictive Models

Nis Meinert, Institute of Communications and Navigation, 06/28/2023

Motivation

Goal: understand FNO as a (physics informed) Neural Operator and its relation to PINNs, GNOs, NKNs, IFNOs, Geo-FNO, . . .

. . . in 15min!

(see my talk *FNO and Beyond* (FDL 2022) for details about FNO)

Motivation

Goal: understand FNO as a (physics informed) Neural Operator and its relation to PINNs, GNOs, NKNs, IFNOs, Geo-FNO, . . .

. . . in 15min!

Why constraining physics?

- We never have enough data
- Q: How/Where to encode prior knowledge to become more data efficient?

Physics-Informed NNs ({c,v,p,s,. . .}PINNs)

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \left(\mathcal{L}_{\ell_2}[\mathrm{NN}](\theta) + \lambda\, \mathcal{L}_{\mathrm{physics}}[\mathrm{NN}](\theta)\right)$$

$$= \min_\theta \left(\sum_i |\mathrm{NN}_\theta(x_i) - y_i|^2 + \lambda \sum_j |\nabla \cdot (\kappa\, \nabla\, \mathrm{NN}_\theta)(x_j) + f_j|^2 \right)$$

- Don’t change NN
- Minimization problem becomes a *multicriterion optimization*
- (λ is **not** a Lagrange multiplier but the scalarization to find Pareto optimal points!)

Physics-Informed NNs ({c,v,p,s,. . .}PINNs)

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \left(\mathcal{L}_{\ell_2}[\mathrm{NN}](\theta) + \lambda\, \mathcal{L}_{\mathrm{physics}}[\mathrm{NN}](\theta)\right)$$

$$= \min_\theta \left(\sum_i |\mathrm{NN}_\theta(x_i) - y_i|^2 + \lambda \sum_j |\nabla \cdot (\kappa\, \nabla\, \mathrm{NN}_\theta)(x_j) + f_j|^2 \right)$$

- Don’t constrain architecture of NN
- Minimization problem becomes a *multicriterion optimization*
- (λ is **not** a Lagrange multiplier but the scalarization to find Pareto optimal points!)

Physics-Informed NNs ({c,v,p,s,. . .}PINNs)

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \left(\mathcal{L}_{\ell_2}[\mathrm{NN}](\theta) + \lambda\, \mathcal{L}_{\mathrm{physics}}[\mathrm{NN}](\theta)\right)$$

$$= \min_\theta \left(\sum_i |\mathrm{NN}_\theta(x_i) - y_i|^2 + \lambda \sum_j |\nabla \cdot (\kappa\, \nabla\, \mathrm{NN}_\theta)(x_j) + f_j|^2 \right)$$

- Don't constrain architecture of NN
  - NN is trained for **specific** b.c., i.c., loading/source terms, etc.
  - **No generalization!**
- Minimization problem becomes a *multicriterion optimization*
  - More data efficient
  - **(Very) challenging to train**

What do we want?

$$\mathrm{NN}_\theta(x_i) = y_i \quad \longrightarrow \quad \mathrm{NN}_\theta[\kappa, f](x_i) = y_i$$

- Be explicit about b.c., i.c., etc.
  - More **data efficient** to constrain solution on manifold but still ...
  - **Generalize** (efficiently) to different (κ, f)
- This makes $\mathrm{NN}(\bullet)$ an operator $\mathrm{NN}[\bullet](\bullet)$
  - Universal *function* approximation $\mapsto$ universal *operator* approximation
- Ideally, no multicriterion optimization necessary / learn physics from data

Example: DeepONet (arXiv:1910.03193)

$$\mathcal{G}_\theta[\kappa, f](x) = \mathrm{NN}_\theta^{\mathrm{branch}}(\kappa, f) \otimes \mathrm{NN}_\theta^{\mathrm{trunk}}(x)$$

- Universal operator approximation: theorem by Chen & Chen (1995)
- Factorization adds inductive bias that is more efficient than vanilla FNN
  - $\mathrm{NN}_\theta^{\mathrm{branch}}$ learns the basis function
  - $\mathrm{NN}_\theta^{\mathrm{trunk}}$ learns the corresponding weights
- Sample
  - (κ, f) at points ξ_i: $(\kappa(\xi_1), f(\xi_1), \kappa(\xi_2), f(\xi_2), \ldots)$
  - x at points x_i: $(x_1, x_2, \ldots)$
- Train with ℓ_2 (and refine with additional ℓ_{physics}?) . . . **still, very generic**

How to scale (efficiently)?

$$h_{j+1}(x) - h_j(x) \sim \int \mathrm{d}y \, (h_j(y) - h_j(x)) \underbrace{k_\theta(x, y, \kappa(x), \kappa(y))}_{\text{Green's function: } G[\kappa](x,y)}$$

- **Vertically (aka *height* $\mapsto$ long-range interactions)**
  - Neurons per layer $\mapsto \infty$
  - $\sum \mapsto \int$
- **Horizontally (aka *depth* $\mapsto$ accuracy)**
  - ResNet / Neural ODE
  - *Shallow-to-deep approach*

(see Nonlocal Kernel Networks (NKNs, `arXiv:2201.02217`))

How to scale (efficiently)?

$$\frac{h(x,(j+1)\,\Delta t) - h(x,j\,\Delta t)}{\Delta t} \sim \int \mathrm{d}y\,(h(y,j\,\Delta t) - h(x,j\,\Delta t))\,k_\theta(x,y,\kappa(x),\kappa(y))$$

- **Vertically (aka *height* $\mapsto$ long-range interactions)**
  - Neurons per layer $\mapsto \infty$
  - $\sum \mapsto \int$
- **Horizontally (aka *depth* $\mapsto$ accuracy)**
  - ResNet / Neural ODE
  - *Shallow-to-deep approach*

(see Nonlocal Kernel Networks (NKNs, `arXiv:2201.02217`))

How to scale (efficiently)?

$$\frac{h(x, (j+1)\,\Delta t) - h(x, j\,\Delta t)}{\Delta t} \sim \int \mathrm{d}y\, (h(y, j\,\Delta t) - h(x, j\,\Delta t))\, k_\theta(x, y, \kappa(x), \kappa(y))$$

- $\int \mathrm{d}y\, k_\theta(x, y)$ is very expensive
- what if **underlying physics** induces $k_\theta(x, y) = k_\theta(|x - y|)$?
  - kernel integral $\mapsto$ convolution
  - evaluation of convolution is very cheap w/ FFT

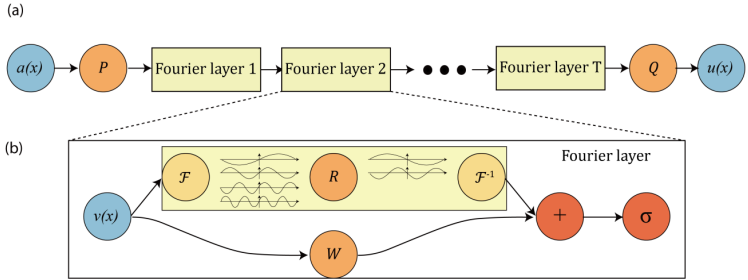
How to scale (efficiently)?

$$\frac{h(x, (j+1)\,\Delta t) - h(x, j\,\Delta t)}{\Delta t} \sim \int \mathrm{d}y\, (h(y, j\,\Delta t) - h(x, j\,\Delta t))\, k_\theta(x, y, \kappa(x), \kappa(y))$$

- $\int \mathrm{d}y\, k_\theta(x, y)$ is **very expensive**
- what if underlying physics induces $k_\theta(x, y) = k_\theta(|x - y|)$?
  - kernel integral $\mapsto$ convolution
  - evaluation of convolution is **very cheap** w/ FFT

Fourier Neural Operators (FNOs, arXiv:2010.08895)

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \underbrace{\mathcal{F}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big]}_{\text{FN-Operator}}(x)$$



Fourier Neural Operators (FNOs, arXiv:2010.08895)

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

Good

- Fast
- Resolution independent*

Bad

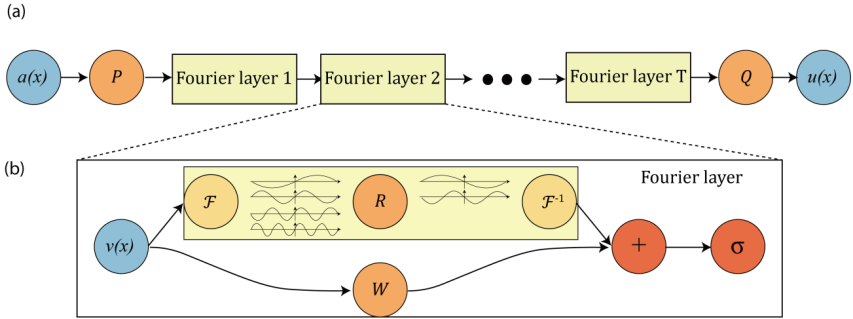
- Unstable
- Only works for uniform meshes
- Implicitly assumes periodicity

*Example: train on 32×32 uniform grid and evaluate on 64×64 uniform grid

Fourier Neural Operators (FNOs, arXiv:2010.08895)

Resolution independence*

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$



Fourier Neural Operators (FNOs, arXiv:2010.08895)

Resolution independence*

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

by integrating out x:

$$\mathcal{F}[f](x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathrm{d}x \underbrace{\mathrm{e}^{-\mathrm{i}wx}}_{\text{orth. basis}} f(x)$$

Resolution independence*

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

via blurring on inverse transformation:

$$\mathcal{F}^{-1}[\tilde{f}](x) = \int_{-\infty}^{\infty} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w) \quad \longrightarrow \quad \mathcal{F}_{\mathrm{tr}}^{-1}[\tilde{f}](x) = \int_{-w_{\min}}^{w_{\max}} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w)$$

... works as long as $|\mathcal{X}| > w_{\max} - w_{\min}$, $x \in \mathcal{X}$

Fourier Neural Operators (FNOs, arXiv:2010.08895)

Resolution independence*

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

via blurring on inverse transformation:

$$\mathcal{F}^{-1}[\tilde{f}](x) = \int_{-\infty}^{\infty} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w) \quad \longrightarrow \quad \mathcal{F}_{\mathrm{tr}}^{-1}[\tilde{f}](x) = \int_{-w_{\min}}^{w_{\max}} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w)$$

$\mathcal{F}_{\mathrm{tr}}^{-1} \sim$ ***physics informed*** **autoencoder** learns LF dynamics

Fourier Neural Operators (FNOs, arXiv:2010.08895)

Resolution independence*

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

via blurring on inverse transformation:

$$\mathcal{F}^{-1}[\tilde{f}](x) = \int_{-\infty}^{\infty} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w) \quad \longrightarrow \quad \mathcal{F}_{\mathrm{tr}}^{-1}[\tilde{f}](x) = \int_{-w_{\min}}^{w_{\max}} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w)$$

$\mathcal{F}_{\mathrm{tr}}^{-1} \sim$ ***physics informed*** **autoencoder** learns low energy regime

Fourier Neural Operators (FNOs, arXiv:2010.08895)

Resolution independence*

$$h_{j+1}(x) \sim \int \mathrm{d}y\, h_j(y)\, k_\theta(|x-y|) = \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h_j\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

via blurring on inverse transformation:

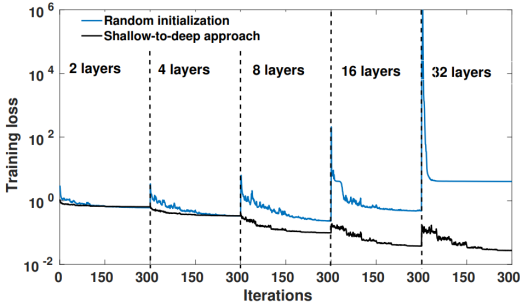
$$\mathcal{F}^{-1}[\tilde{f}](x) = \int_{-\infty}^{\infty} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w) \quad \longrightarrow \quad \mathcal{F}_{\mathrm{tr}}^{-1}[\tilde{f}](x) = \int_{-w_{\min}}^{w_{\max}} \mathrm{d}w\, \mathrm{e}^{\mathrm{i}wx}\, \tilde{f}(w)$$

$\mathcal{F}_{\mathrm{tr}}^{-1} \sim$ ***physics informed*** **autoencoder** learns an effective theory

Implicit/Incremental Fourier Neural Operators (IFNOs)

$$\frac{h(x,(j+1)\,\Delta t) - h(x,j\,\Delta t)}{\Delta t} \sim \mathcal{F}_{\mathrm{tr}}^{-1}\Big[\mathcal{F}\big[h(\bullet, j\,\Delta t)\big] \cdot \mathcal{F}\big[k_\theta\big]\Big](x)$$

- Stabilize training by weight sharing across layers h
- *Shallow-to-deep* training via ResNet structure



Geometry-Aware FNO (Geo-FNO, arXiv:2207.05209)

FNO $\rightarrow$ Geo-FNO:

- Uniform meshes $\rightarrow$ unstructured meshes*
- by replacing first DFT with NDFT

***Caveats:** (IMO! Talk to me if you are interested)

- NDFT is no longer an orthogonal projection
- Memory efficient implementation of NDFT necessary (*fast NDFT*)
- NDFT approximates a DFT which deteriorates if mesh get *less* uniform

Invariant Neural Operators (INOs, arXiv:2212.14365)

- Constrain kernel beyond $k_\theta(x, y) = k_\theta(|x - y|)$
- Translational **invariance**:

$$\mathcal{G}[\tilde{f}](x + \Delta x) = \mathcal{G}[f](x)$$

  where $\tilde{f}(x + \Delta x) = f(x)$
- Translational **equivariance**:

$$\mathcal{G}[\tilde{f}](x + \Delta x) = \mathcal{G}[f](x) + \Delta x$$

  where $\tilde{f}(x + \Delta x) = f(x)$
- and similar for rotations