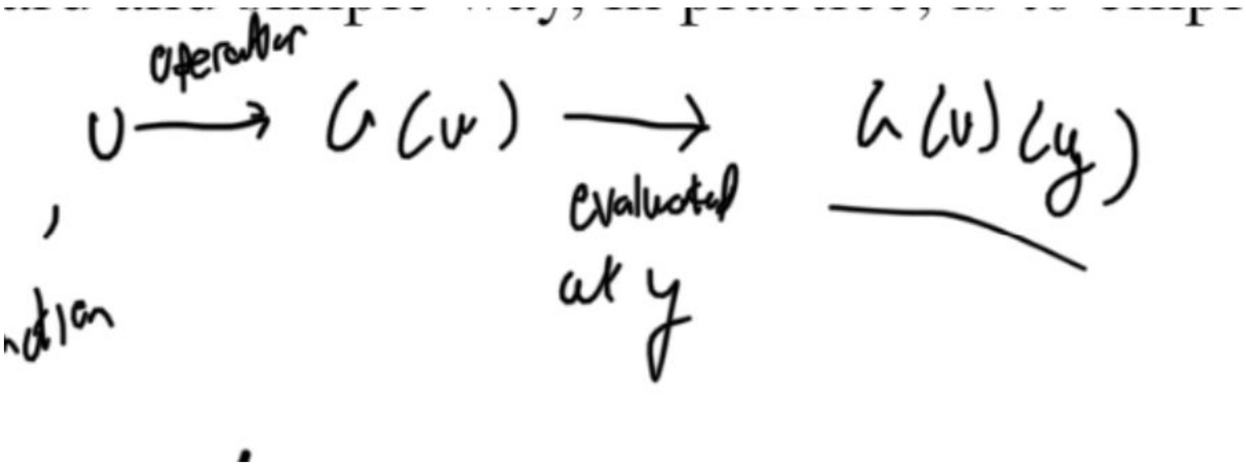
DeepONets

Universal Approximation Theorem for Neural networks

- Neural Networks can approximate any continuous function, nonlinear continuous functional, or nonlinear operator
  - Functional - function to reals mapping
  - Operator - function to function mapping



What is the UAT Saying?

- Does not say how operators can be learned effectively - only proves that it can be done
- I.E. CNNs vs. FNNs for image classification, RNNs vs. FNNs for time series data, etc.
    - Different architectures can be more efficient than simple FNNs
- UAT only concerned with approximation error
    - Optimization error? How easy is it to train?
    - Generalization error? How well does it generalize?
- Thus, though UAT says that a Feedforward Neural Network <u>CAN</u> learn an operator, it doesn't guarantee efficient learning or good generalization

An Approach - DeepONet

- Keep the problem very general
  - Weakest possible constraints on training data - consistent sensor locations for input functions in the training dataset
- An approach for more accurate, efficient learning of operators
  - Goal: achieve smaller error than baseline FNNs
- Working for a wide class of ODEs and PDEs
- Learn an operator *G* mapping u -> G(u)
  - Input: u, y

Training Data - u spaces

- Sample u from different function spaces
  - Gauss Random Field (GRF)
  - Orthogonal (Chebyshev) Polynomials
- Solve sampled functions via numerical methods (operation)
  - RK4, RK5 for ODE
  - 2nd order finite difference for PDE
- Generate data triplet
  - (u, y, G(u)(y))

Representing Functions

- Discrete representation
  - Sufficient points at which function is evaluated
  - “Sensor” locations
- Sensors: $x_1, x_2, \ldots, x_m$
- Function Representation: $[u(x_1), u(x_2), \ldots, u(x_m)]$

DeepONet Architecture

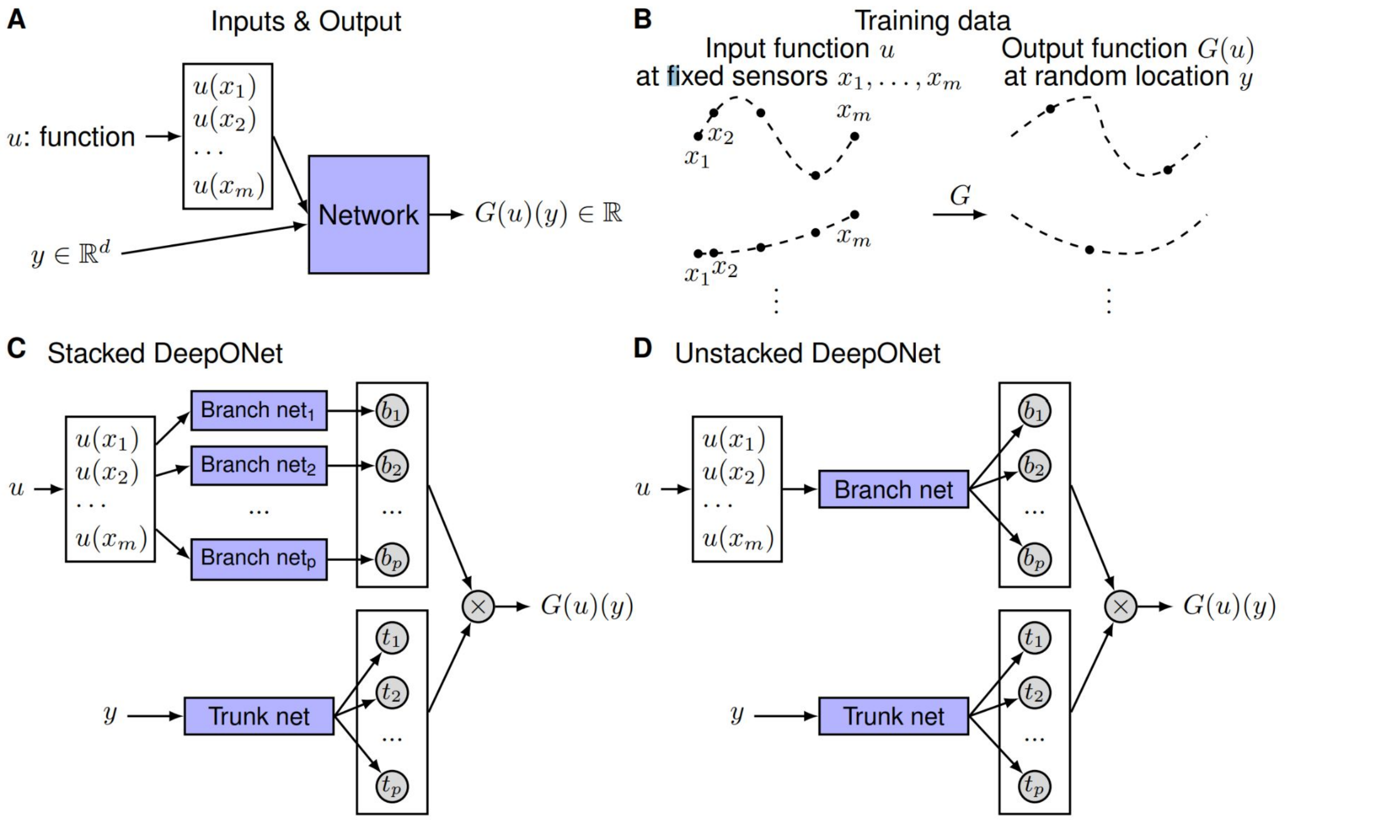
- Employ an architecture based on UAT

Theorem 1 (Universal Approximation Theorem for Operator). *Suppose that σ 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 $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \ldots, n$, $k = 1, \ldots, p$, $j = 1, \ldots, m$, 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 $y \in K_2$.

Architecture Visualization



"Branch" network

$$\left[\sum_{k=1}^{p}\sum_{i=1}^{n} c_i^k\, \sigma\left(\sum \xi_{ij}^k\, u(x_j) + \theta_i^k\right)\right] \boxed{\sigma(w_k \cdot y + \zeta_k)}$$

"trunk" network

learning association for output location y and labelled data

One layer within branch

all layers within one branch network

several branch networks, multiple $\underline{u}$

$$\begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_p \end{bmatrix}$$

(unstacked)

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_p \end{bmatrix}$$

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

Architecture Discussion

- The architecture of trunk and branch networks is malleable
  - Paper demonstrates FNNs
  - Other subnetworks may improve accuracy
- Embedding prior knowledge = better generalization
  - Inductive bias - alter network architecture to embed physics knowledge
  - Independent u and y - establish understanding of physical laws expressed by functions
  - Output G(u)(y) is a function of y that is “conditioned” on u

Experiments - Linear ODE

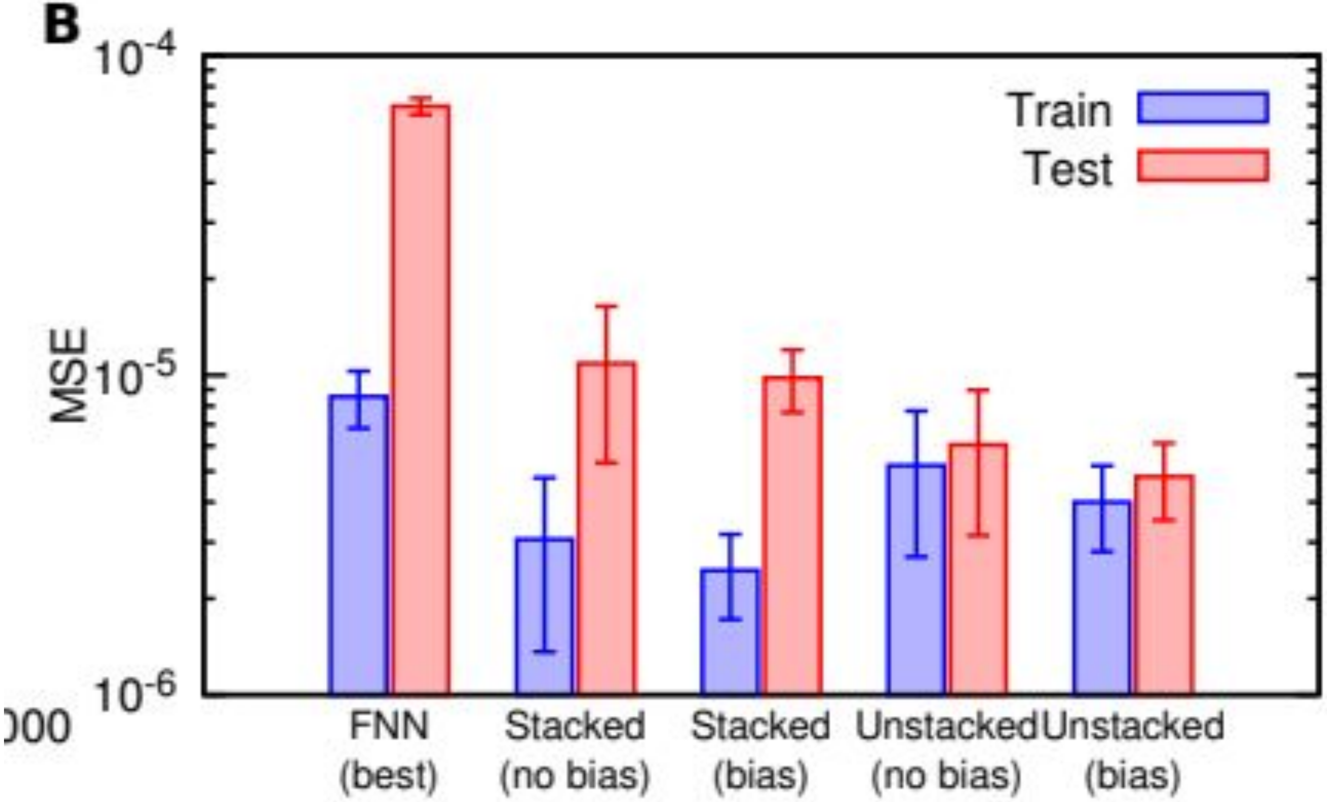
A 1D dynamic system is described by

$$\frac{ds(x)}{dx} = g(s(x), u(x), x), \quad x \in [0, 1],$$

with an initial condition $s(0) = 0$. Our goal is to predict $s(x)$ over the whole domain $[0, 1]$ for any $u(x)$.

- Generate Training Data - Sufficient Sensor locations*
- Sufficient Training for convergence
- Learn the antiderivative operator
- Baseline FNNs - grid search optimal hyperparameters and experiment for best depth
- Train DeepONets on the same data

Experiments - Linear ODE Results



SNN Proposal

- Use SNNs as branch and trunk networks within the DeepONet
- Assess performance of SNN-DeepONet against FNN-DeepONet and Baseline FNNs
    - Reproducibility enabled by detail in paper and DeepXDE library

Additional

Case	u space	# Sensors m	# Training	# Test	# Iterations	Other parameters
4.1.1	GRF ($l = 0.2$)	100	10000	100000	50000	
4.1.2	GRF ($l = 0.2$)	100	10000	100000	100000	
4.2	GRF ($l = 0.2$)	100	10000	100000	100000	$k = 1, T = 1$
4.3	GRF ($l = 0.2$)	100		1000000	500000	

Sources

- "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators" (Lu et al.) https://arxiv.org/abs/1910.03193
- "DeepXDE: A deep learning library for solving differential equations" (Lu et al.) https://arxiv.org/abs/1907.04502, https://deepxde.readthedocs.io/en/latest/