Take a look at a basic ODE:

$$\frac{ds(t)}{dt} = u(t) \qquad t \in [0,1]$$

$$s(0) = 0$$

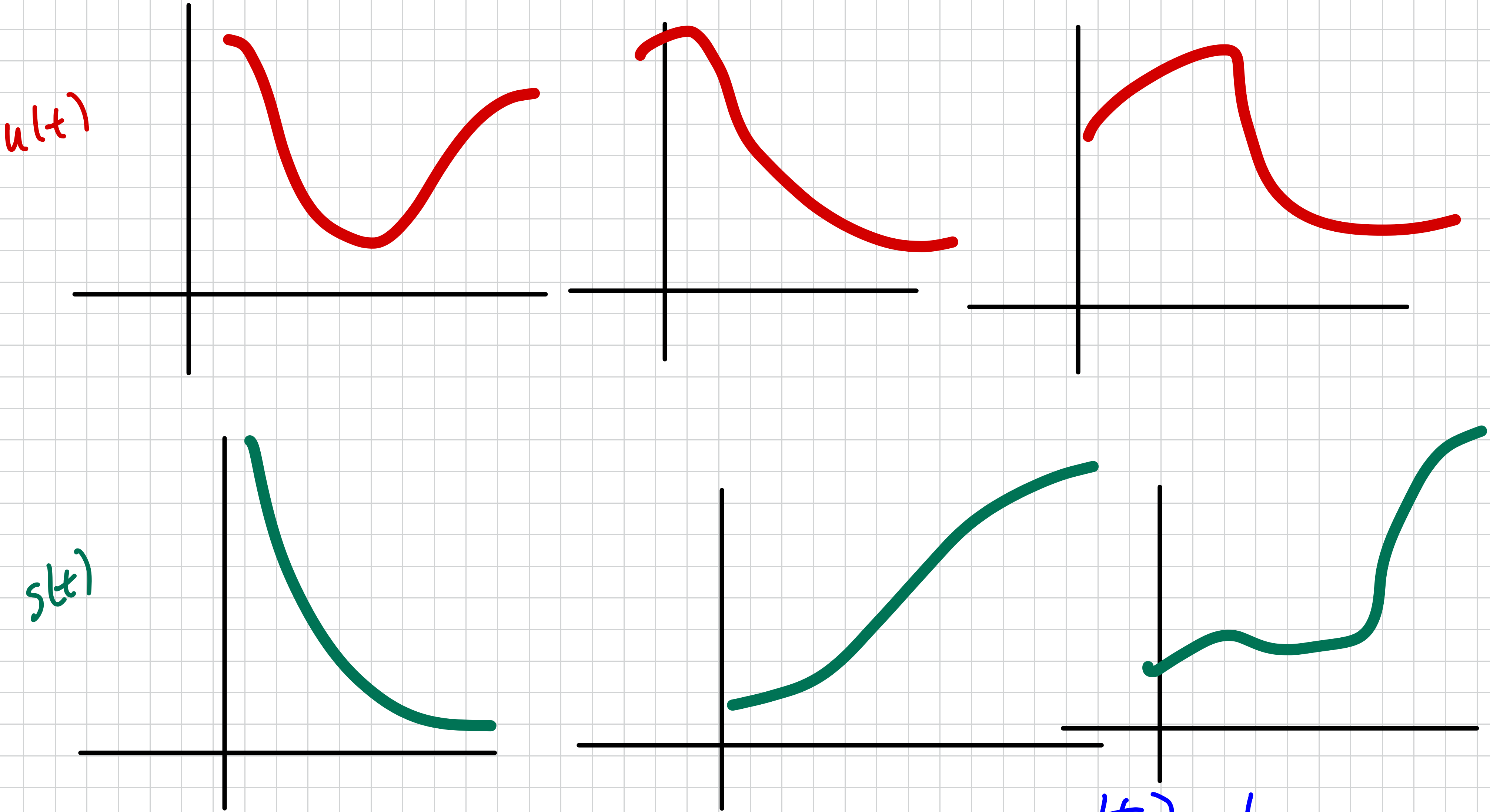
$u(t)$ = known function

Want: mapping $G : u \longmapsto s$

$G(u) = s$

↳ note: $G(u)$ is evaluated at a point "y": $G(u)(y) = s(y)$

$$\frac{ds(t)}{dt} = u(t)$$



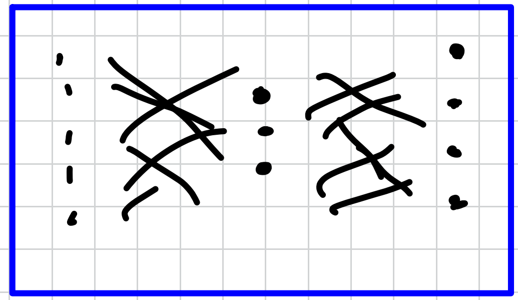
when u(t) changes, s(t) changes

let t be the only input for now

want to build a NN so that

$NN \approx G$

give u $\rightarrow$



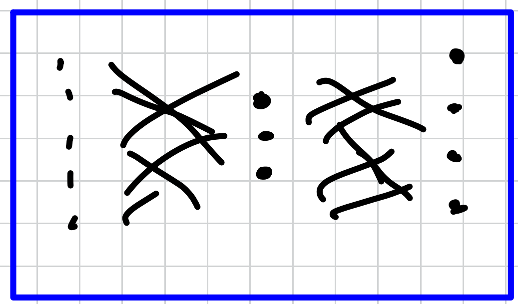
$\rightarrow$ s

Specifically want

N

give $u(t)$ on $[0,T]$ $\rightarrow$

t $\rightarrow$

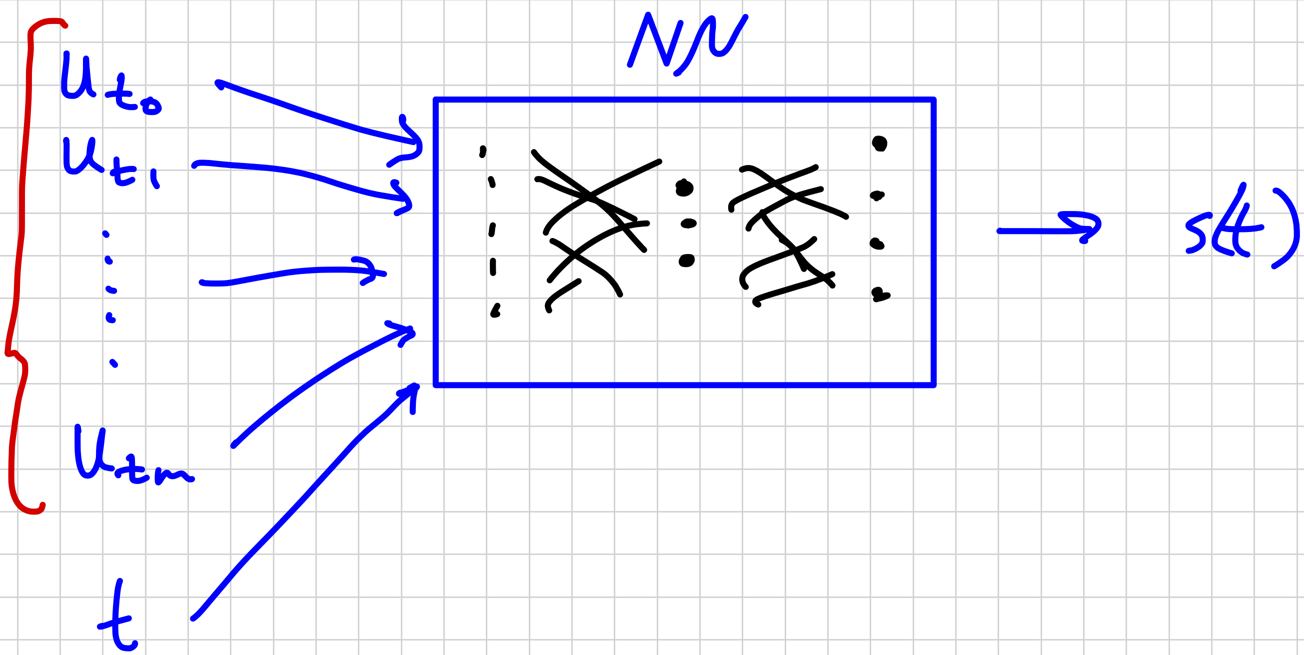


$\rightarrow$ $s(t)$

How will we input <u>any</u> function $u(t)$?

$\rightarrow$ sample $u(t)$ at $t_0, \ldots, t_m$

"sensor" in DeepONets



Approach #1 : use FF NN on all inputs

$\rightarrow$ limited convergence

Approach #2

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*

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

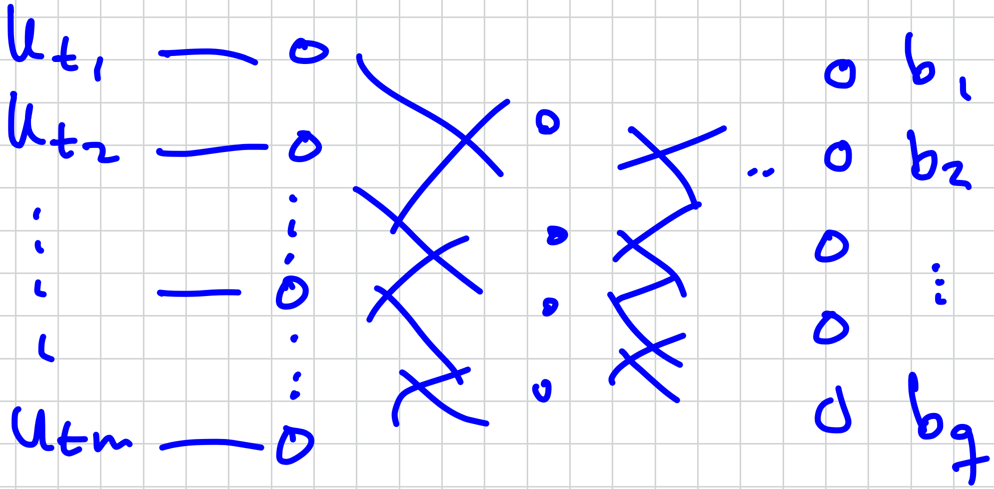
holds for all $u \in V$ and $y \in K_2$.

use a NN for
$u_{t_1} \cdots u_{t_m}$

use another for
t

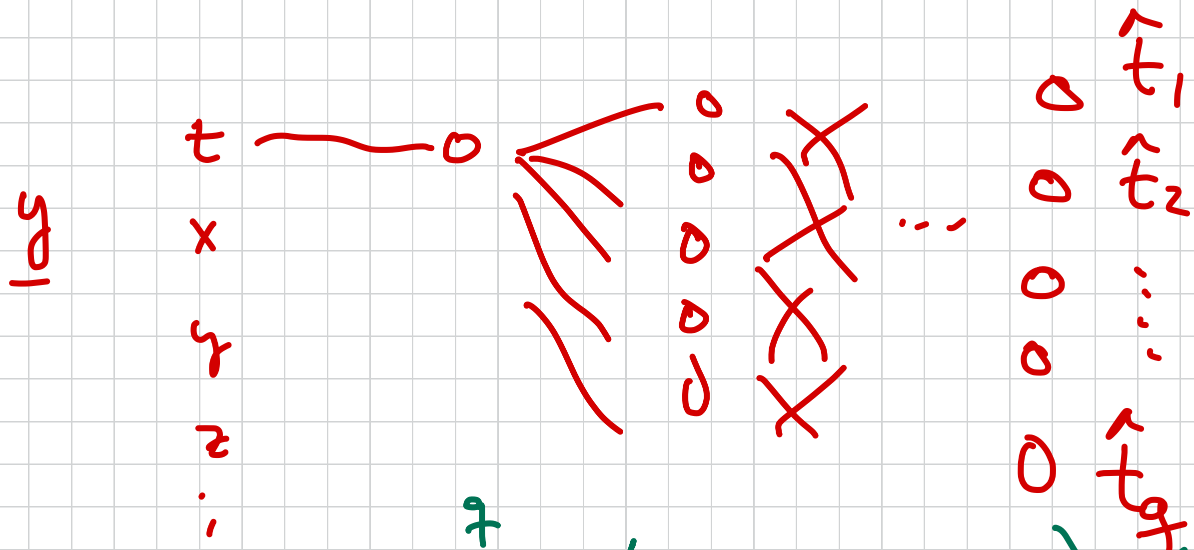
take u respresented by $u(t_1) \; ---- \; u(t_m)$

$t_1 \; ---- \; t_m$



features

"branch network"



features

$$G_\theta(u)(y) = \sum_{k=1}^{q} b_k\big(u(t_1) \; ..... \; u(t_m)\big) \; \hat{t}_k(y)$$

weights + biases

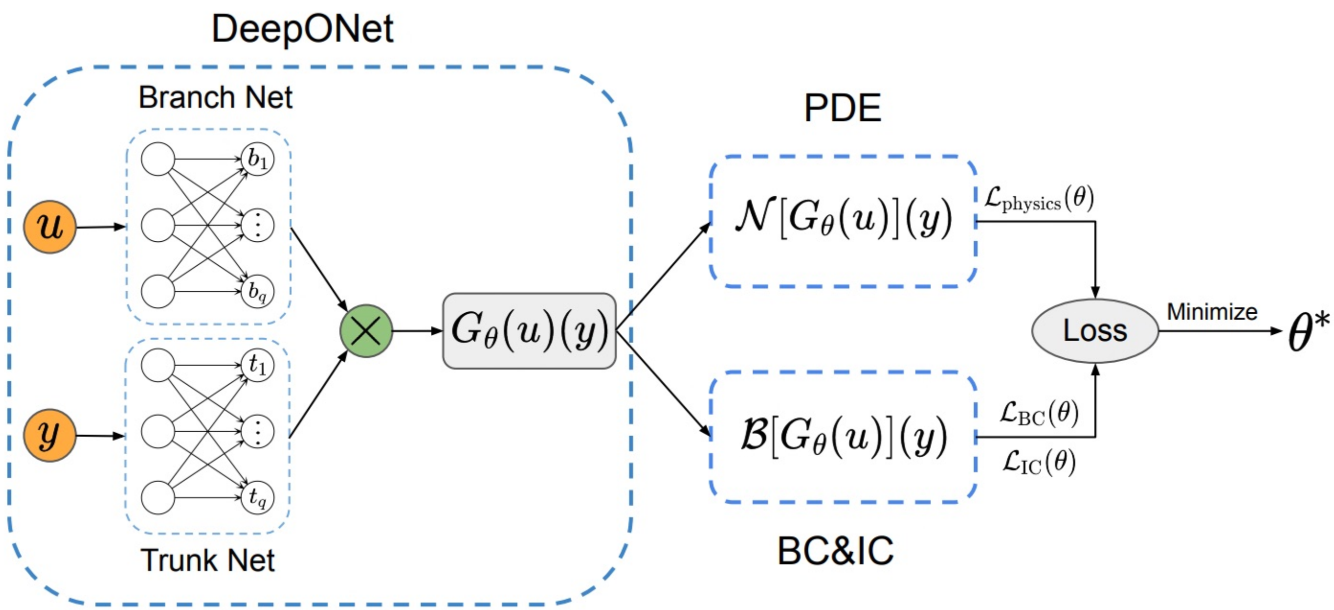
Take N samples of functions

$$u^{(i)} \quad i = 1 - \cdots - N$$

$$\to u^{(i)} \to \left(u^{(i)}(t_1) - \cdots - u^{(i)}(t_m) \right)$$

Take P output evaluations (for "t")

$$\mathcal{L}(\Theta) = \frac{1}{N \cdot P} \sum_{i=1}^{N} \sum_{j=1}^{P} \left| G_\theta(u^{(i)})(y_j^{(i)}) - G(u^{(i)})(y_j^{(i)}) \right|^2$$



Training set:

$u, y, G(u)(y)$

$u^{(1)}(t_1) \cdots u^{(1)}(t_m)$ $\quad y_1^{(1)}$ $\quad G(u^{(1)})(y_1^{(1)})$

$u^{(1)}(t_1) \cdots u^{(1)}(t_m)$ $\quad y_2^{(1)}$

$\vdots$ $\quad \vdots$ $\quad \vdots$

$u^{(1)}(t_1) \cdots u^{(1)}(t_m)$ $\quad y_P^{(1)}$

$N\cdot P\times m$

$\vdots$

$u^{(i)}(t_1) \cdots u^{(i)}(t_m)$ $\quad y_1^{(i)}$

$u^{(i)}(t_1) \cdots u^{(i)}(t_m)$ $\quad y_P^{(i)}$ $\quad G(u^{(i)})(y_P^{(i)})$

$N\cdot P\times d$ $\quad N\cdot P\times 1$

$\vdots$

Back to example:

$$\frac{ds(t)}{dt} = u(t) \qquad t \in [0,1]$$

$$s(0) = 0$$

train

Generate 10,000 $u(t)$ from a zero mean Gaussian Process (with a quadratic and length scale 0.2.

Generate corresponding $s(t)$ with RK45

Choose $m=100$ sensors $t_1 \cdots t_m$ uniformly

Choose $P=1$ observation for $s(\cdot)$ in $[0,1]$

testing with $m=100$, $P=100$

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left| G_\theta(u^{(i)})(y^{(i)}) - s^{(i)}(y^{(i)}) \right|^2$$

$$u^{(i)} = [u^{(i)}(t_1) \;----\; u^{(i)}(t_m)]$$

ODE45 result.

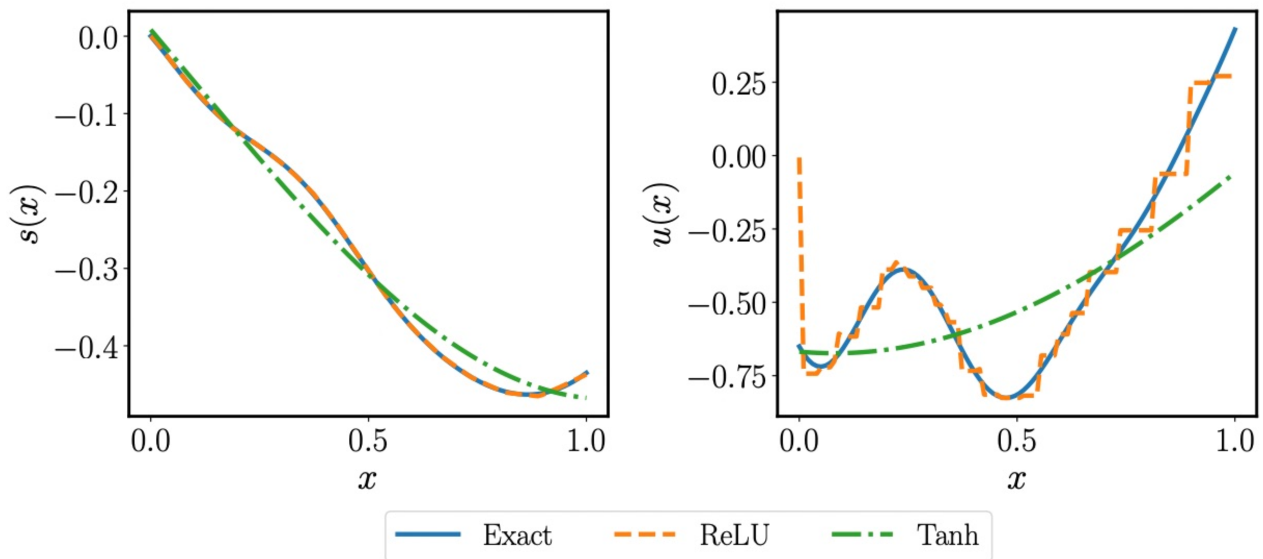


Figure 2: *Learning the anti-derivative operator*: Predicted solution $s(x)$ and residual $u(x)$ versus the ground truth for a representative input function. The results are obtained by training a conventional DeepONet model [33] equipped with different activation functions after 40,000 iterations of gradient descent using the Adam optimizer.