# A simple ODE case

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 over the whole domain $[0, 1]$ for any $u(x)$.

**Linear case:** $g(s(x), u(x), x) = u(x)$

Then, the operator G would be: $G : u(x) \mapsto s(x) = \int_0^x u(\tau) d\tau.$

## DeepOnet

Learn the solution operators of parametric PDEs → We will try to approximate $G$ (the solution of our PDE operator) by two neural networks:

$$G_\theta(u)(y) = \sum_{k=1}^{q} \underbrace{b_k\left(u(x_1), u(x_2), \ldots, u(x_m)\right)}_{Branch} . \underbrace{t_k(\mathbf{y})}_{Trunk}$$

We want to obtain G, so our goal would be:

$$G_\theta(u)(y) \approx G(u)(y)$$

So we will enforce that condition into a loss function:

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

$$\mathcal{L}_{Operator}(\theta) = \frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} \left| \sum_{k=1}^{q} b_k\left(u(x_1), u(x_2), \ldots, u(x_m)\right) . t_k(y_j^{(i)}) - G(u^{(i)}) y_j^{(i)} \right|^2$$

where $N$ is the number of functions $u(x)$ in our training dataset, $P$ is the number of points inside the domain at which we will evaluate $G(u)$.

$m$ : Number of points at which we evaluated our input functions.

$N$ : Number of input functions.

$P$ : Number of points at which we evaluate the output function → output sensors.

## How to generate data?

We can choose uniformly m + 1 points $x_j = a + j(b - a)/m$, j = 0, 1, · · · , m from [a, b], and define the function $u_m(x)$ as follows:

$$u_m(x) = u(x_j) + \frac{u(x_{j+1}) - u(x_j)}{x_{j+1} - x_j}(x - x_j),\ x_j \le x \le x_{j+1},\ j = 0, 1, \cdots, m - 1.$$

Regression using Gaussian:

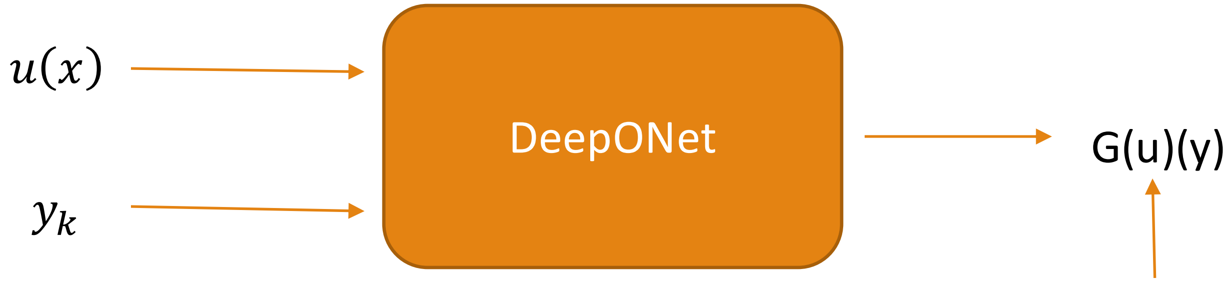
$$u \sim \mathcal{G}(0, k_l(x_1, x_2)),$$

$$\boldsymbol{\sum(x_1, x_2) = k(x_1, x_2) + I\sigma_y^2}$$

$$\boldsymbol{k(x_1, x_2) = \sigma^2 e^{\frac{1}{2l^2}(x_1 - x_2)}}$$

$$G : u(x) \mapsto s(x) = \int_0^x u(\tau) d\tau.$$

- We generate discrete $u_m(x)$ with discrete x values and different set of output locations $y_j, j = 1,2, \ldots P$.
- For these output locations, we generate discrete ground truth values $\cos(y_j)$ to train our model.




Then we can define the loss function as follows:

**Predictions** **Ground truth**

$$L(\theta) = \frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} \left| \sum_{k=1}^{q} b_k(\boldsymbol{u}^{(i)}(\boldsymbol{x}_1), \ldots, \boldsymbol{u}^{(i)}(\boldsymbol{x}_m)) t_k(\boldsymbol{y}_j^{(i)}) - s(\boldsymbol{y}_j^{(i)}) \right|^2,$$

- P: Total no. of output locations
- q: no. of branch nets or no. of neurons in output layer of trunk net