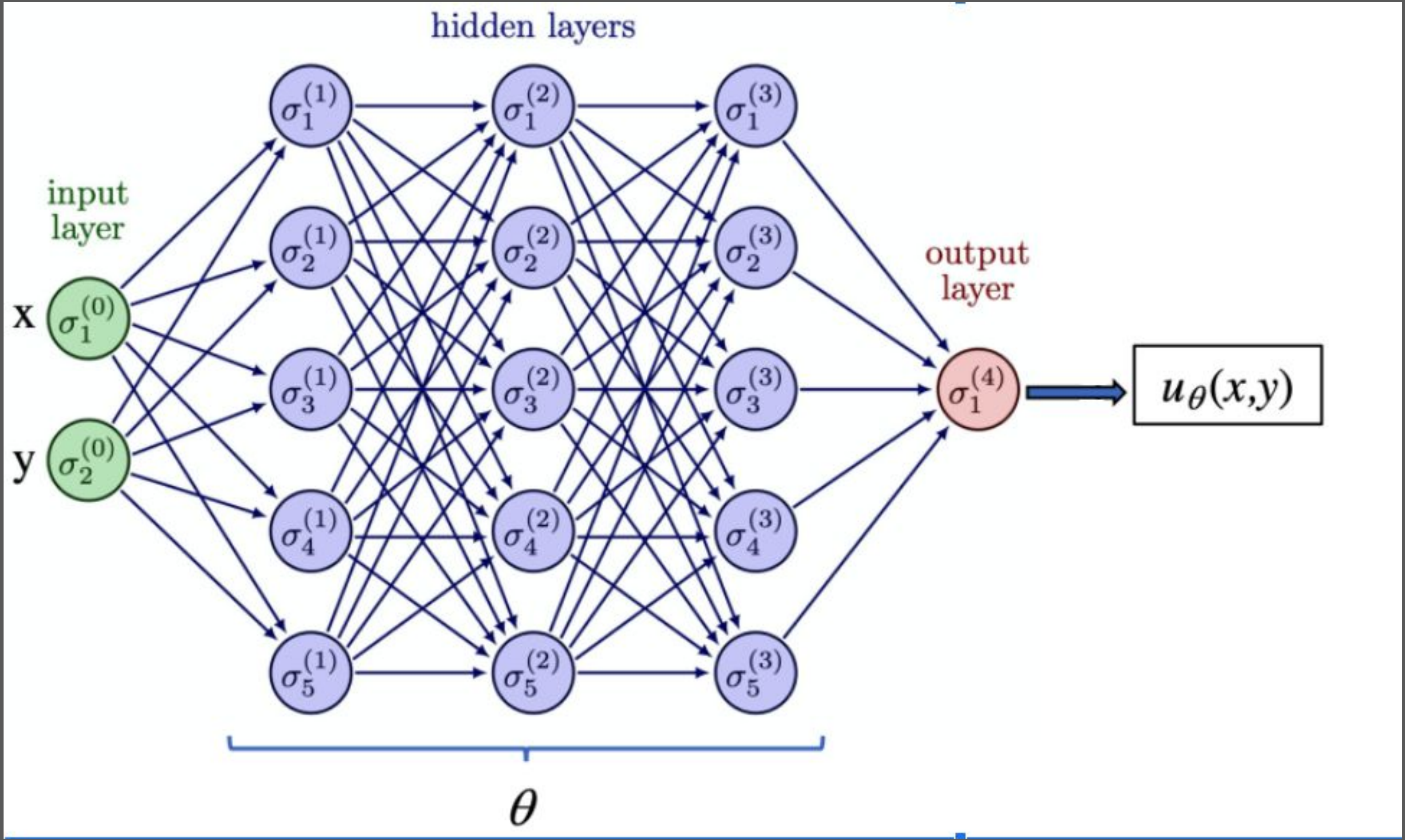
I. Dimitropoulos
(PhD student at university of Patras)

I. Contopoulos
(Supervisor, Researcher at academy of Athens)

Solution of a Simple Physics Problem with Physics Informed Neural Networks (PINNs)

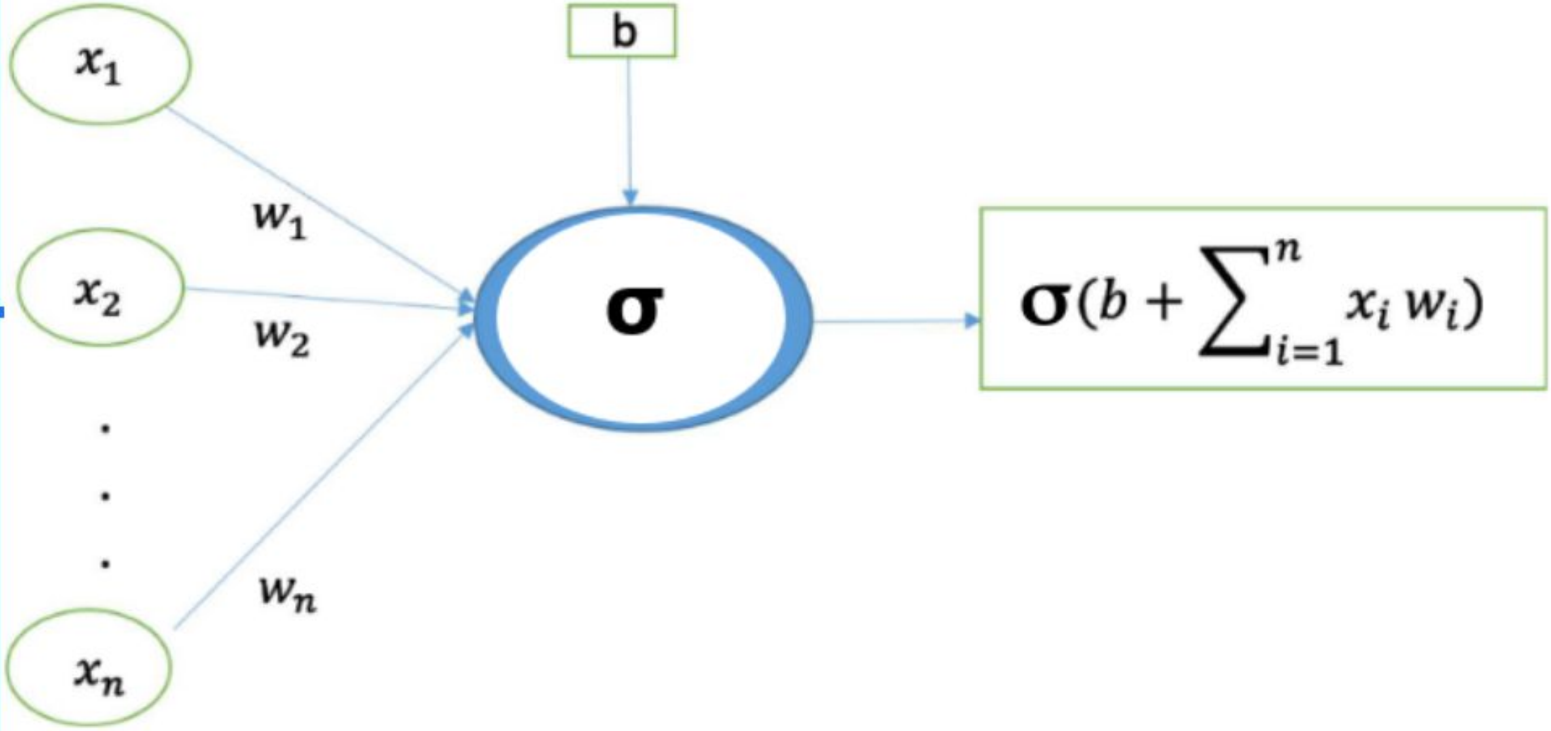
ΕΛ.Ι.Δ.Ε.Κ.

Ελληνικό Ίδρυμα Έρευνας & Καινοτομίας



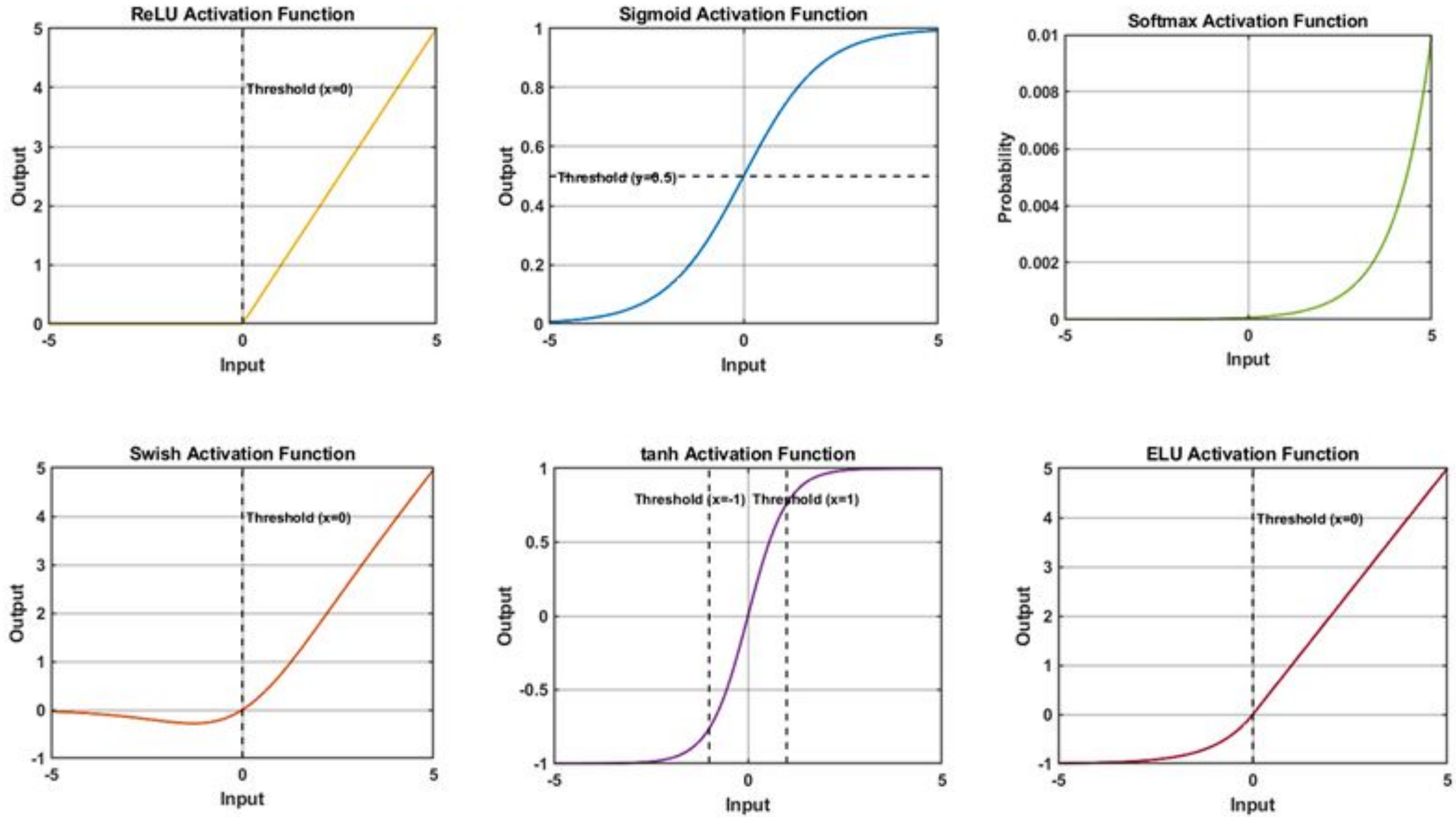
Hubert Baty, A hands-on introduction to PINNs, 2024

structure of a neuron



A diagram to show the work of a neuron: input *x*, weights *w*, bias b, activation function σ

Activation function



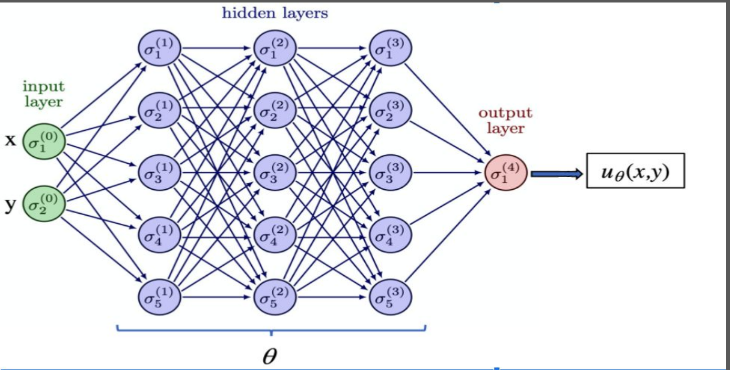
we introduce what is called a multi-layer perceptron, which is one of the most common kind of neural networks. Note that any other statistical model could alternatively be used. The goal is to calibrate its parameters θ such that u_θ approximates the target solution $u(\boldsymbol{x})$. u_θ is a non-linear approximation function, organized into a sequence of $L+1$ layers. The first layer $\mathcal{N}^0$ is called the input layer and is simply:

$$\mathcal{N}^0(\boldsymbol{x}) = \boldsymbol{x}. \tag{3}$$

Each subsequent layer ℓ is parameterized by its weight matrix $\boldsymbol{W}^\ell \in \mathbb{R}^{d_{\ell-1}\times d_\ell}$ and a bias vector $\boldsymbol{b}^\ell \in \mathbb{R}^{d_\ell}$, with d_ℓ defined as the output size of layer ℓ. Layers ℓ with $\ell \in [\![1, L-1]\!]$ are called hidden layers, and their output value can be defined recursively:

$$\mathcal{N}^\ell(\boldsymbol{x}) = \sigma(\boldsymbol{W}^\ell \mathcal{N}^{\ell-1}(\boldsymbol{x}) + \boldsymbol{b}^\ell), \tag{4}$$

σ is a non-linear function, generally called activation function.



Hubert Baty, A hands-on introduction to PINNs, 2024

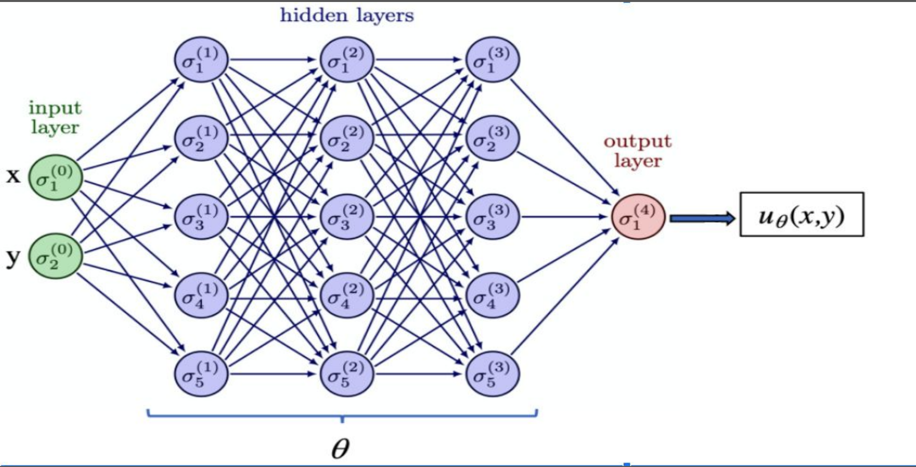
The final layer is the output layer, defined as follows:

$$\mathcal{N}^L(\boldsymbol{x}) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\boldsymbol{x}) + \boldsymbol{b}^L, \tag{5}$$

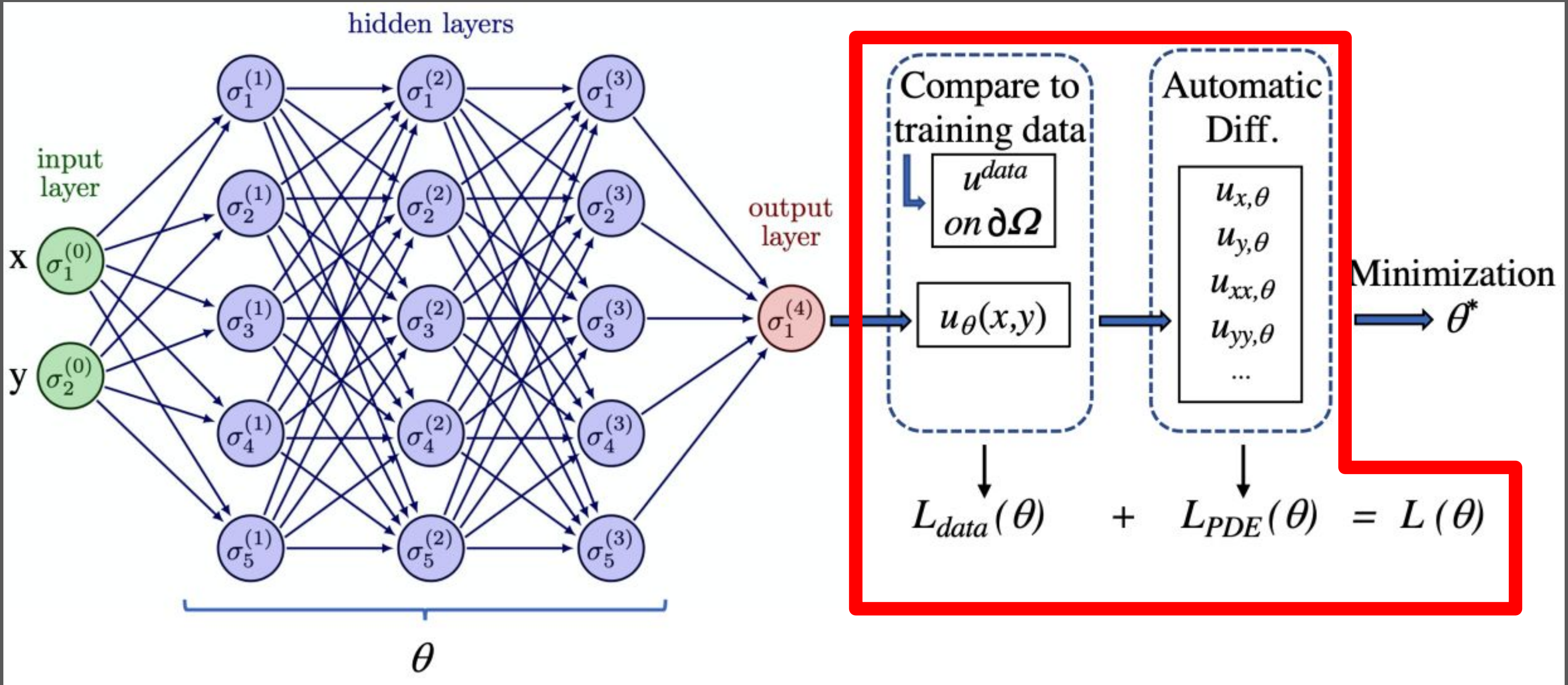
Finally, the full neural network u_θ is defined as $u_\theta(\boldsymbol{x}) = \mathcal{N}^L(\boldsymbol{x})$. It can be also written as a sequence of non-linear functions

$$u_\theta(\boldsymbol{x}) = (\mathcal{N}^L \circ \mathcal{N}^{L-1} \ldots \mathcal{N}^0)(\boldsymbol{x}), \tag{6}$$

where $\circ$ denotes the function composition and $\theta = \{\boldsymbol{W}^l, \boldsymbol{b}^l\}_{l=1,L}$ represents the parameters of the network.



Hubert Baty, A hands-on introduction to PINNs, 2024



Hubert Baty, A hands-on introduction to PINNs, 2024

$$L_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left|(u_\theta(\boldsymbol{x_i}) - u_i^{data}\right|^2 . \quad (8)$$

L_{data} is called the loss function, and equation (7) the learning problem.

In PINNs approach, a specific loss function L_{PDE} can be defined as,

$$L_{PDE}(\theta) = \frac{1}{N_c} \sum_{i=1}^{N_c} |\mathcal{F}(u_\theta(\boldsymbol{x_i}))|^2 , \quad (10)$$

where the evaluation of the residual equation is performed on a set of N_c points denoted as $\boldsymbol{x}_i$. These points are commonly referred to as collocation points. A composite total loss function is typically formulated as follows

$$L(\theta) = \omega_{data} L_{data}(\theta) + \omega_{PDE} L_{PDE}(\theta), \quad (11)$$

where ω_{data} and ω_{PDE} are weights to be assigned to ameliorate potential imbalances between the two partial losses. These weights can be user-specified or automatically tuned.

Hubert Baty, A hands-on introduction to PINNs, 2024

$$L_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \Big|(u_\theta(\boldsymbol{x_i}) -$$

L_{data} is called the loss function, and equation (7) the learning

In PINNs approach, a specific loss function L_{PDE} can be defi

Example for the heat equation

$$\frac{\partial u}{\partial t} = a\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

The NN gives us the function u_θ and we need:

$$\frac{\partial u_\theta}{\partial t} - a\left(\frac{\partial^2 u_\theta}{\partial x^2} + \frac{\partial^2 u_\theta}{\partial y^2} + \frac{\partial^2 u_\theta}{\partial z^2}\right) \to 0$$

so $\mathcal{F}(u_\theta) = \frac{\partial u_\theta}{\partial t} - a\left(\frac{\partial^2 u_\theta}{\partial x^2} + \frac{\partial^2 u_\theta}{\partial y^2} + \frac{\partial^2 u_\theta}{\partial z^2}\right)$

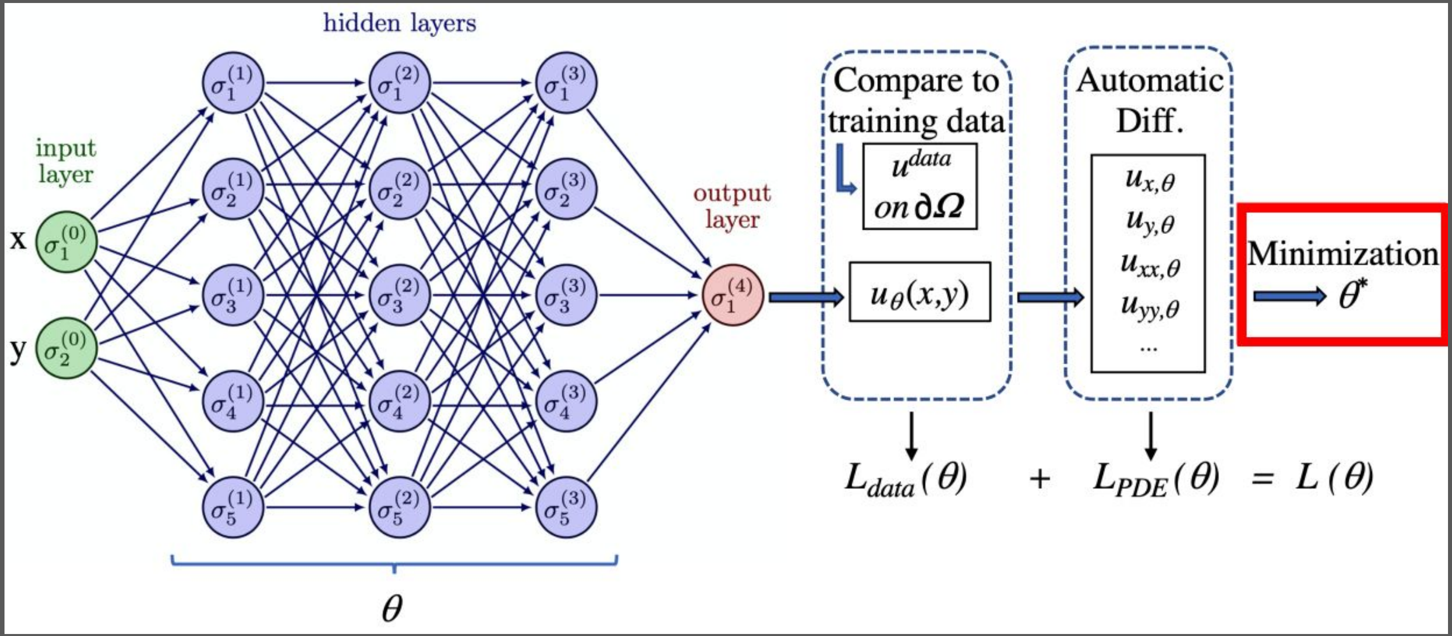
$$L_{PDE}(\theta) = \frac{1}{N_c} \sum_{i=1}^{N_c} |\mathcal{F}(u_\theta(\boldsymbol{x_i}))|^2, \quad (10)$$

where the evaluation of the residual equation is performed on a set of N_c points denoted as $\boldsymbol{x}_i$. These points are commonly referred to as collocation points. A composite total loss function is typically formulated as follows

$$L(\theta) = \omega_{data} L_{data}(\theta) + \omega_{PDE} L_{PDE}(\theta), \quad (11)$$

where ω_{data} and ω_{PDE} are weights to be assigned to ameliorate potential imbalances between the two partial losses. These weights can be user-specified or automatically tuned.

Hubert Baty, A hands-on introduction to PINNs, 2024



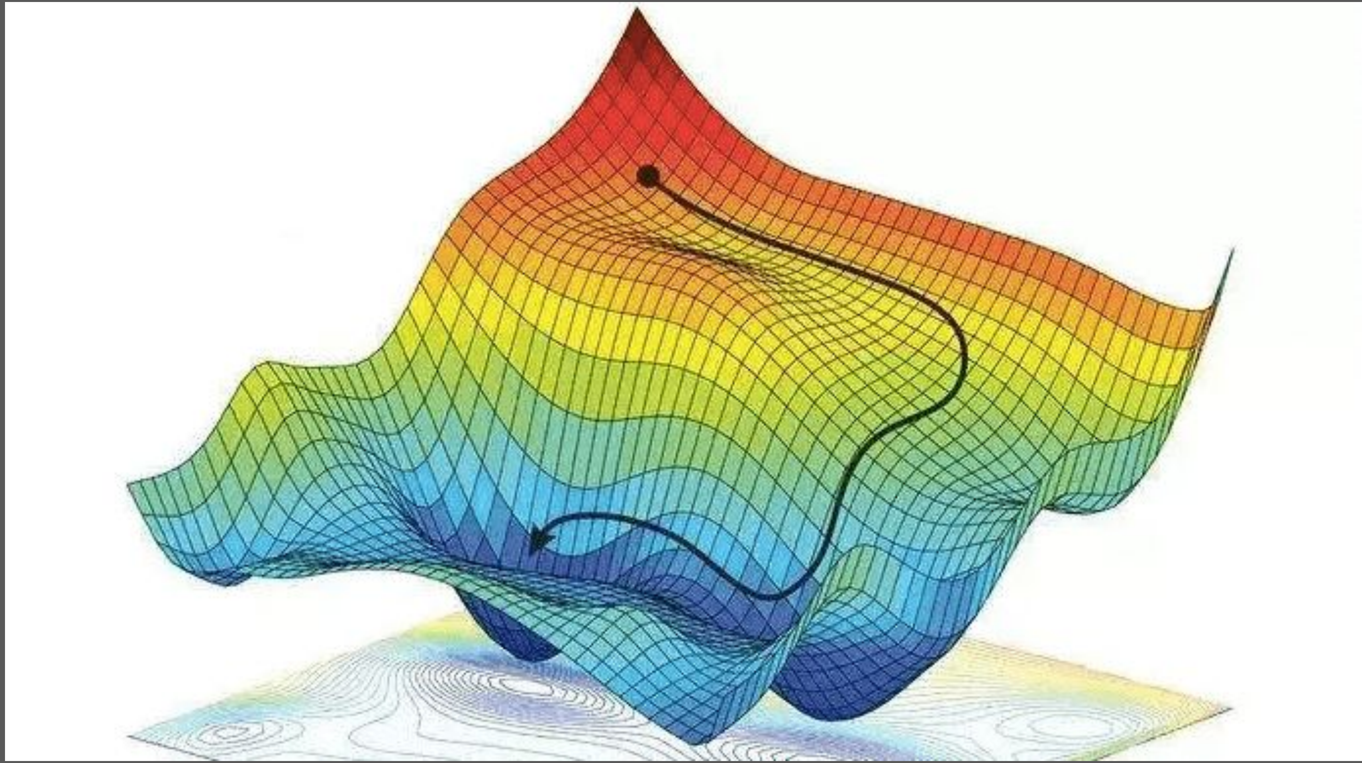
u_θ is considered to be a good approximation of u if predictions $u_\theta(\boldsymbol{x_i})$ are close to target outputs u_i^{data} for every data samples i. We want to minimize the prediction error on the dataset, hence it's natural to search for a value $\theta^\star$ solution of the following optimization problem:

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} L_{data}(\theta) \quad * \tag{7}$$

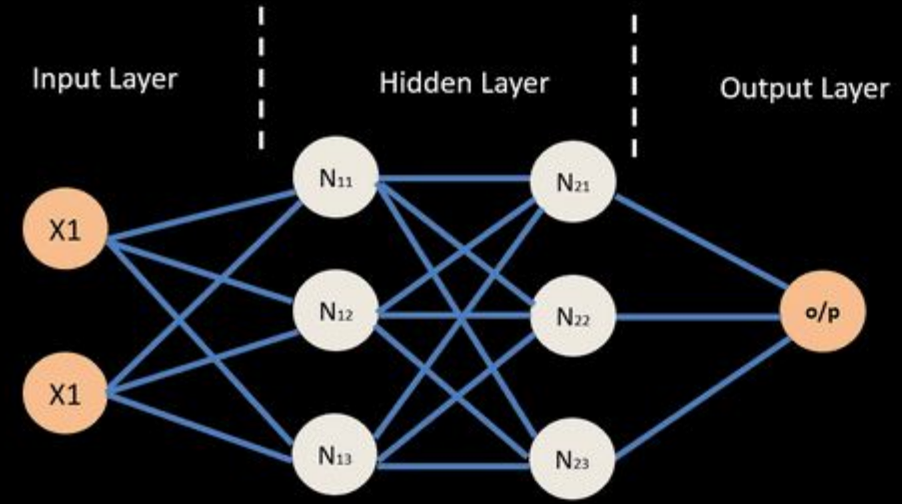
* $\underset{x \in S}{\arg\min} f(x) := \{x \in S \,:\, f(s) \geq f(x) \text{ for all } s \in S\}$

Hubert Baty, A hands-on introduction to PINNs, 2024

Gradient Descent Algorithm



Neural Network – Backpropagation



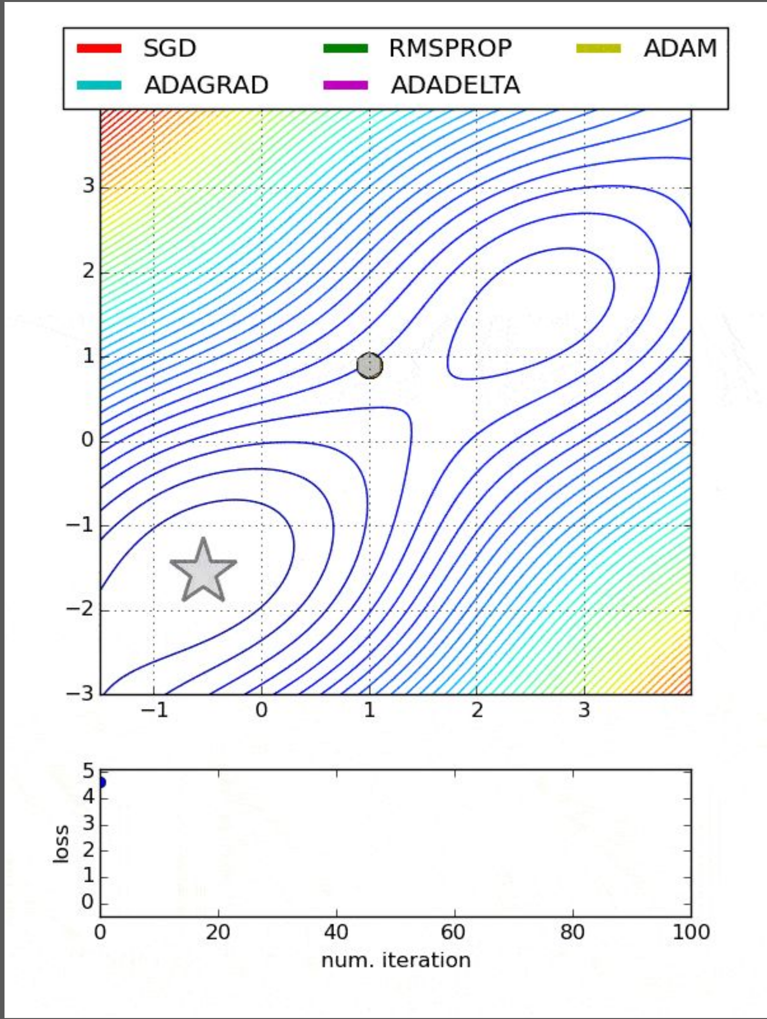
Solving equation (7) is typically accomplished through a (stochastic) gradient descent algorithm. This algorithm depends on automatic differentiation techniques to compute the gradient of the loss L_{data} with respect to the network parameters θ. The algorithm is iteratively applied until convergence towards the minimum is achieved, either based on a predefined accuracy criterion or a specified maximum iteration number as,

$$\theta^{j+1} = \theta^j - l_r \nabla_\theta L(\theta^j), \tag{9}$$

with $L = L_{data}$, for the j-th iteration also called epoch in the literature, where l_r is called the learning rate parameter.

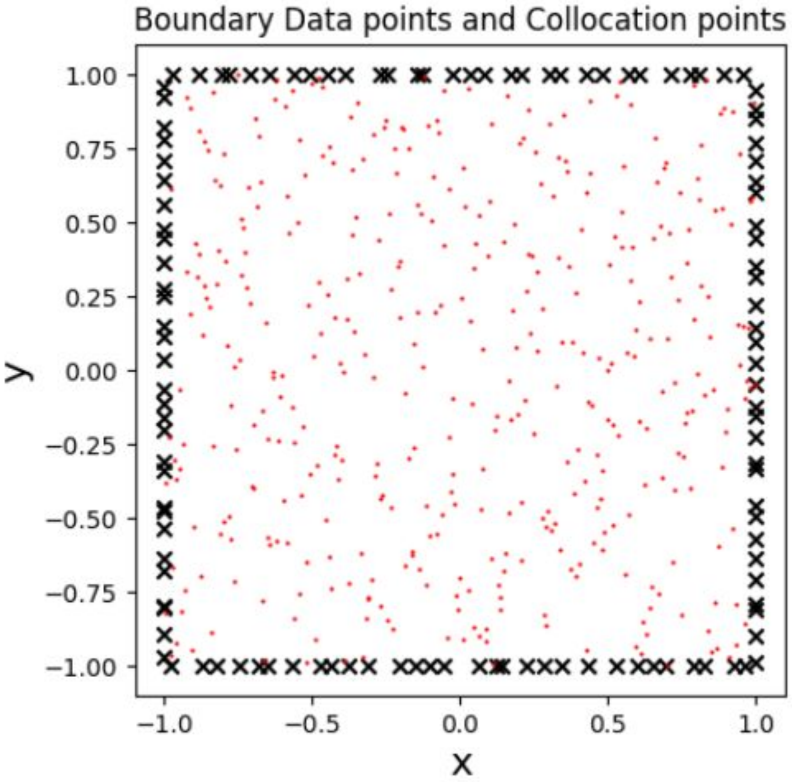
Hubert Baty, A hands-on introduction to PINNs, 2024

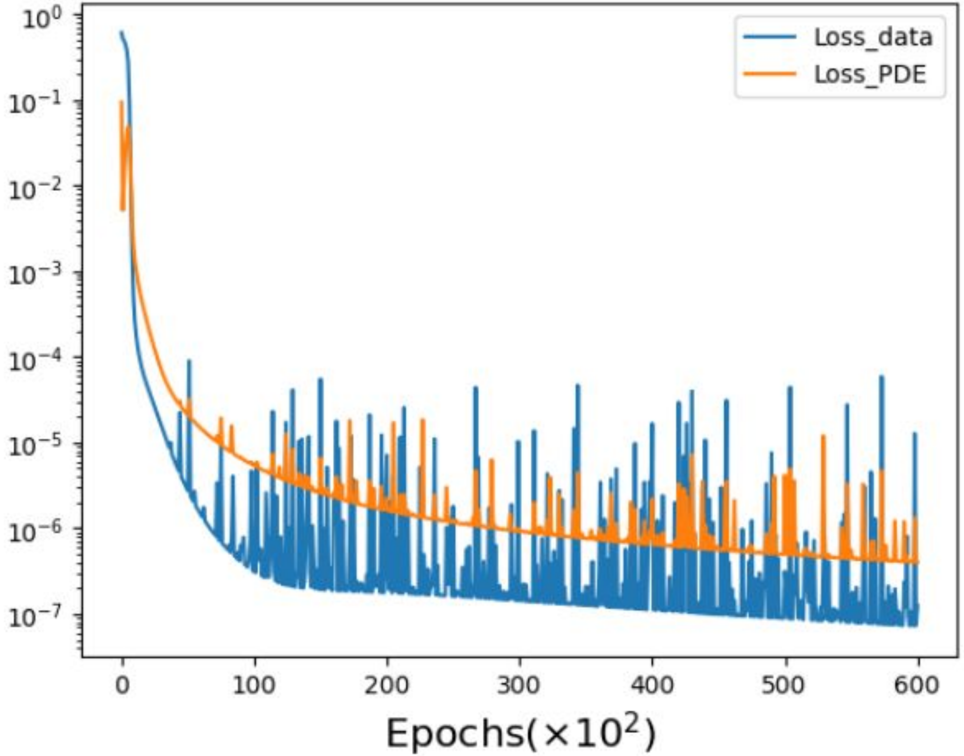
Optimizer



miro.medium.com

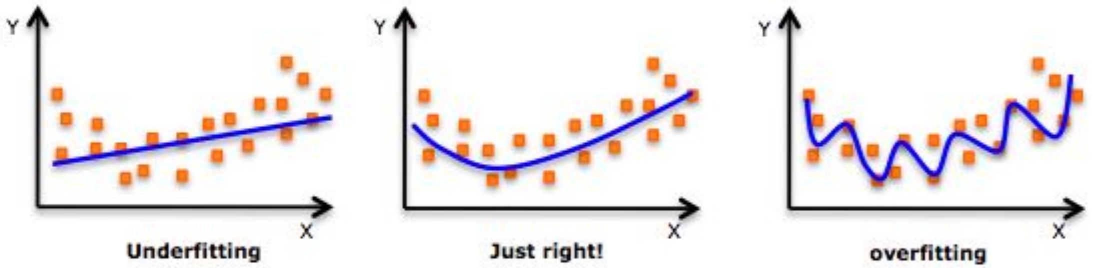
PINNs are mesh-free method!





Hubert Baty, A hands-on introduction to PINNs, 2024

But you have to be careful with the overfitting…



An example of overfitting, underfitting and a model that's "just right!"

Heat equation

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

- $u = u(x, y, z, t)$ is temperature as a function of space and time;
- $\frac{\partial u}{\partial t}$ is the rate of change of temperature at a point over time;
- u_{xx}, u_{yy}, and u_{zz} are the second spatial derivatives (*thermal conductions*) of temperature in the x, y, and z directions, respectively;
- $\alpha \equiv \frac{k}{c_p \rho}$ is the thermal diffusivity, a material-specific quantity depending on the *thermal conductivity* k, the *specific heat capacity* c_p, and the *mass density* ρ.

Wikipedia: Heat equation

We are going to solve 1D heat equation

$$\frac{\partial T}{\partial t} = \frac{k}{\rho \hat{C}_p}\left(\frac{\partial^2 T}{\partial x^2}\right) = \alpha\left(\frac{\partial^2 T}{\partial x^2}\right)$$

Initial condition: $t = 0, T = 50 \ \forall \ x$

Boundary conditions:

$$\left.\begin{array}{ll} x = 0, & T = 60 \\ x = 2H, & T = 60 \end{array}\right\} \ \forall \ t > 0$$

Exercise for 2D heat equation