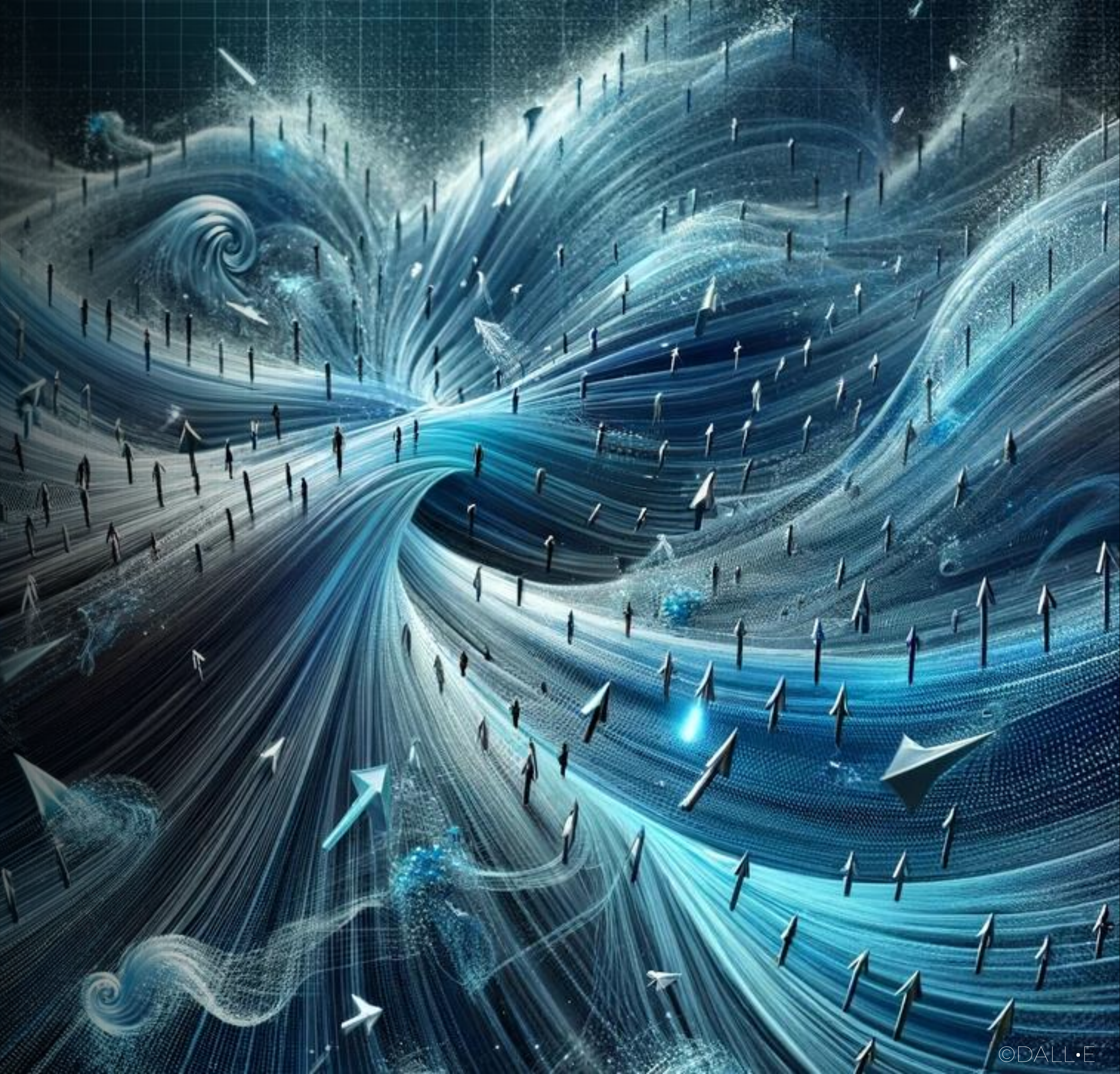
Physics-informed neural networks (PINNs)

A novel approach for forward and inverse modelling

Adil Thami

23/05/2024 | ELI-E seminar



©DALL·E

Neural Networks quésako?

Artificial Neural Networks (ANNs)

1. A computational model inspired by the structure and function of the human brain, *e.g.*, the perceptron – *Historically*
2. A function mapping the input(s) vector(s) $\boldsymbol{x}$ to predicted output(s) vector(s) $\widehat{\boldsymbol{y}}$ – *Mathematically*

$$\widehat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$$

with $\boldsymbol{\theta}$ *being weights* $\boldsymbol{W}$ *and bias* $\boldsymbol{b}$

Deep Learning

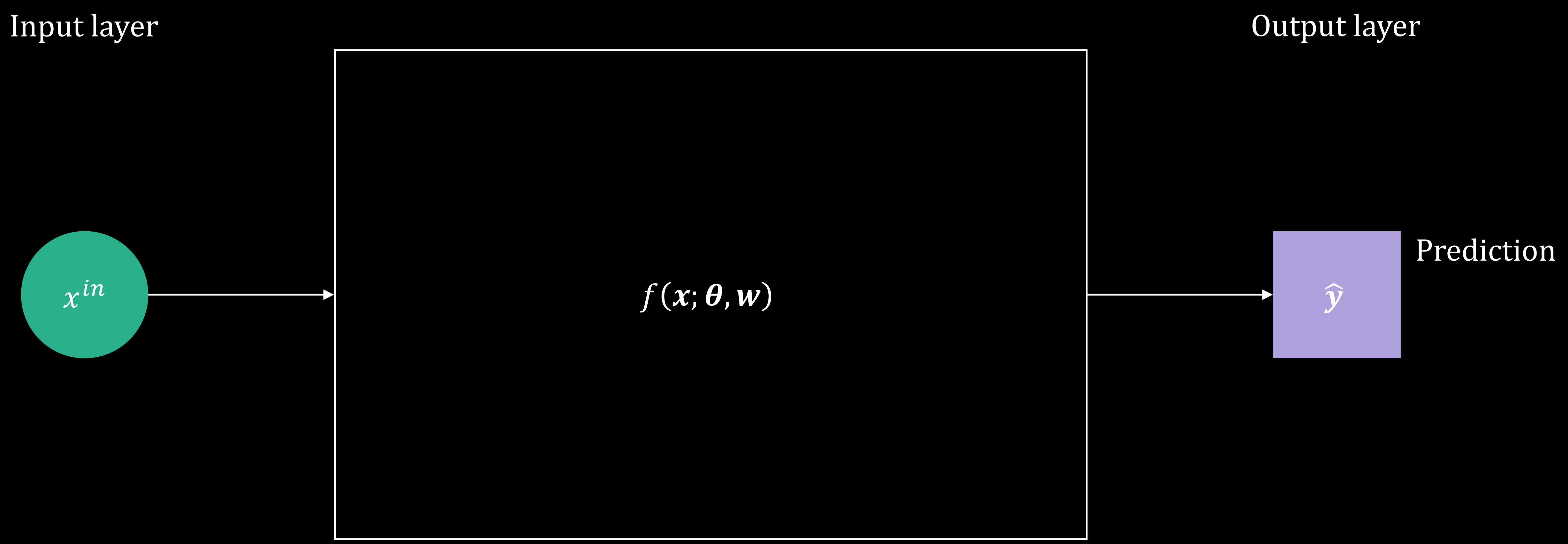
1. Learning multiple levels of composition, *i.e.*, constructing complex patterns or functions by combining simpler ones
   - *e.g.*, Deep Neural Networks (DNNs)

Neural Networks quésako?

NNs ~ function mapping the input(s) vector(s) $\boldsymbol{x}$ to predicted output(s) vector(s) $\hat{\boldsymbol{y}}$

$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$; $\hat{\boldsymbol{y}}$ is a linear combination of the input features

$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{w}$; $\hat{\boldsymbol{y}}$ is a non-linear combination, with $\boldsymbol{\phi}$ a nonlinear transformation, and $\boldsymbol{w}$ being mapping parameters (weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$)



Input layer
Output layer
x^{in}
$f(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{w})$
$\hat{\boldsymbol{y}}$
Prediction

Neural Networks quésako?

NNs ~ function mapping the input(s) vector(s) $\boldsymbol{x}$ to predicted output(s) vector(s) $\hat{\boldsymbol{y}}$

$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$; $\hat{\boldsymbol{y}}$ is a linear combination of the input features

$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{w}$; $\hat{\boldsymbol{y}}$ is a non-linear combination, with $\boldsymbol{\phi}$ a nonlinear transformation, and $\boldsymbol{w}$ being mapping parameters (weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$)

Layers

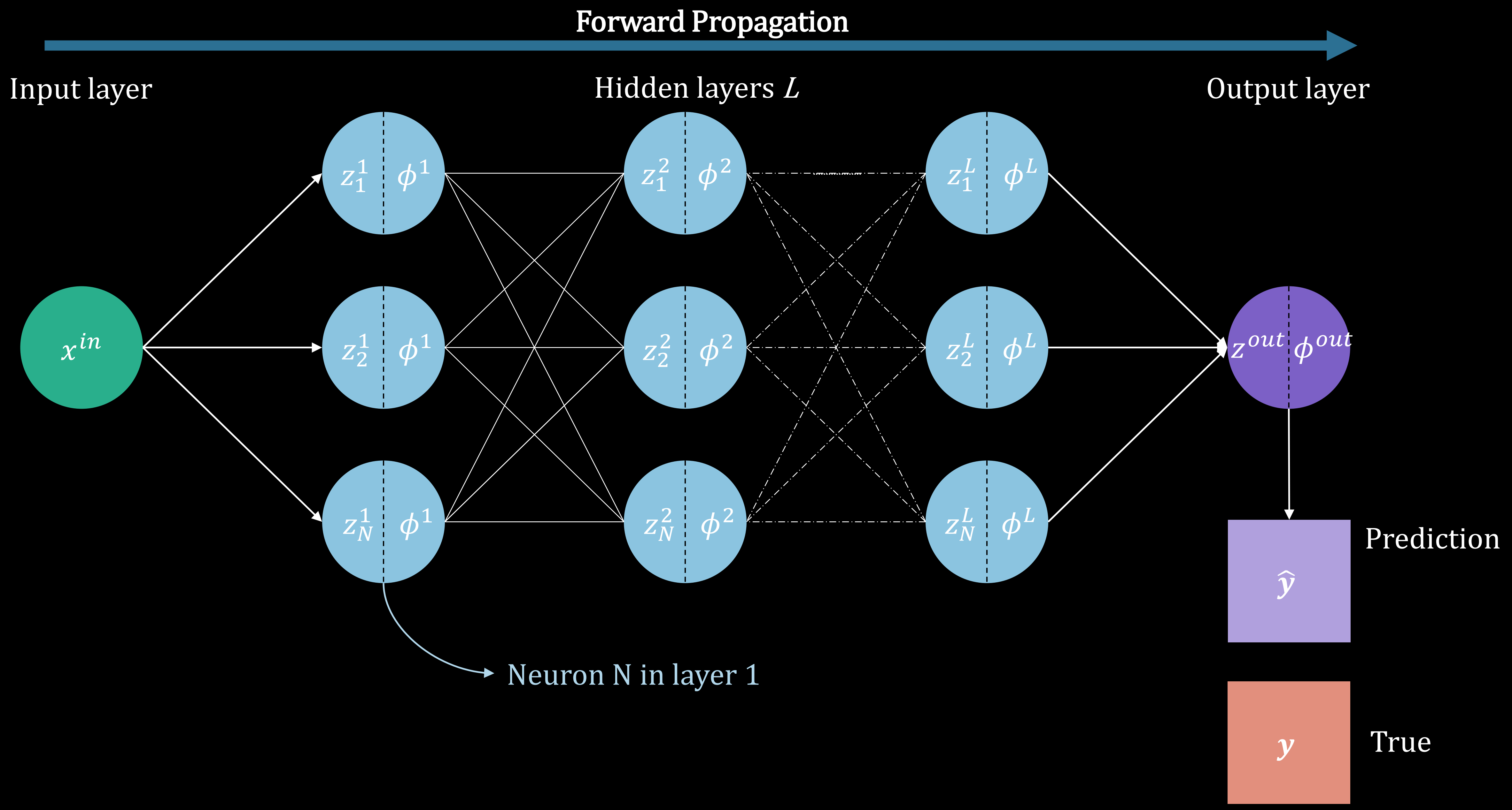
$\boldsymbol{x}$; Input layer

$\boldsymbol{a}^l = \phi(\boldsymbol{z}^l) = \phi(\boldsymbol{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l)$; Hidden layers $l = 1 \dots L-1$

$\hat{\boldsymbol{y}}$; Output layer L

Forward

Propagation


Forward Propagation
Input layer
Hidden layers L
Output layer
x^{in}
z_1^1 ϕ^1
z_2^1 ϕ^1
z_N^1 ϕ^1
z_1^2 ϕ^2
z_2^2 ϕ^2
z_N^2 ϕ^2
z_1^L ϕ^L
z_2^L ϕ^L
z_N^L ϕ^L
z^{out} ϕ^{out}
Neuron N in layer 1
Prediction
$\hat{y}$
y
True

Neural Networks quésako?

NNs ~ function mapping the input(s) vector(s) $\boldsymbol{x}$ to predicted output(s) vector(s) $\hat{\boldsymbol{y}}$

$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$; $\hat{\boldsymbol{y}}$ is a linear combination of the input features

$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{w}$; $\hat{\boldsymbol{y}}$ is a non-linear combination, with $\boldsymbol{\phi}$ a nonlinear transformation, and $\boldsymbol{w}$ being mapping parameters (weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$) to the final output

Forward propagation

$\boldsymbol{x}$; Input layer

$\boldsymbol{a}^l = \phi(\boldsymbol{z}^l) = \phi(\boldsymbol{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l)$; Hidden layers $l = 1 \ldots L-1$

$\hat{\boldsymbol{y}}$; Output layer L

Forward

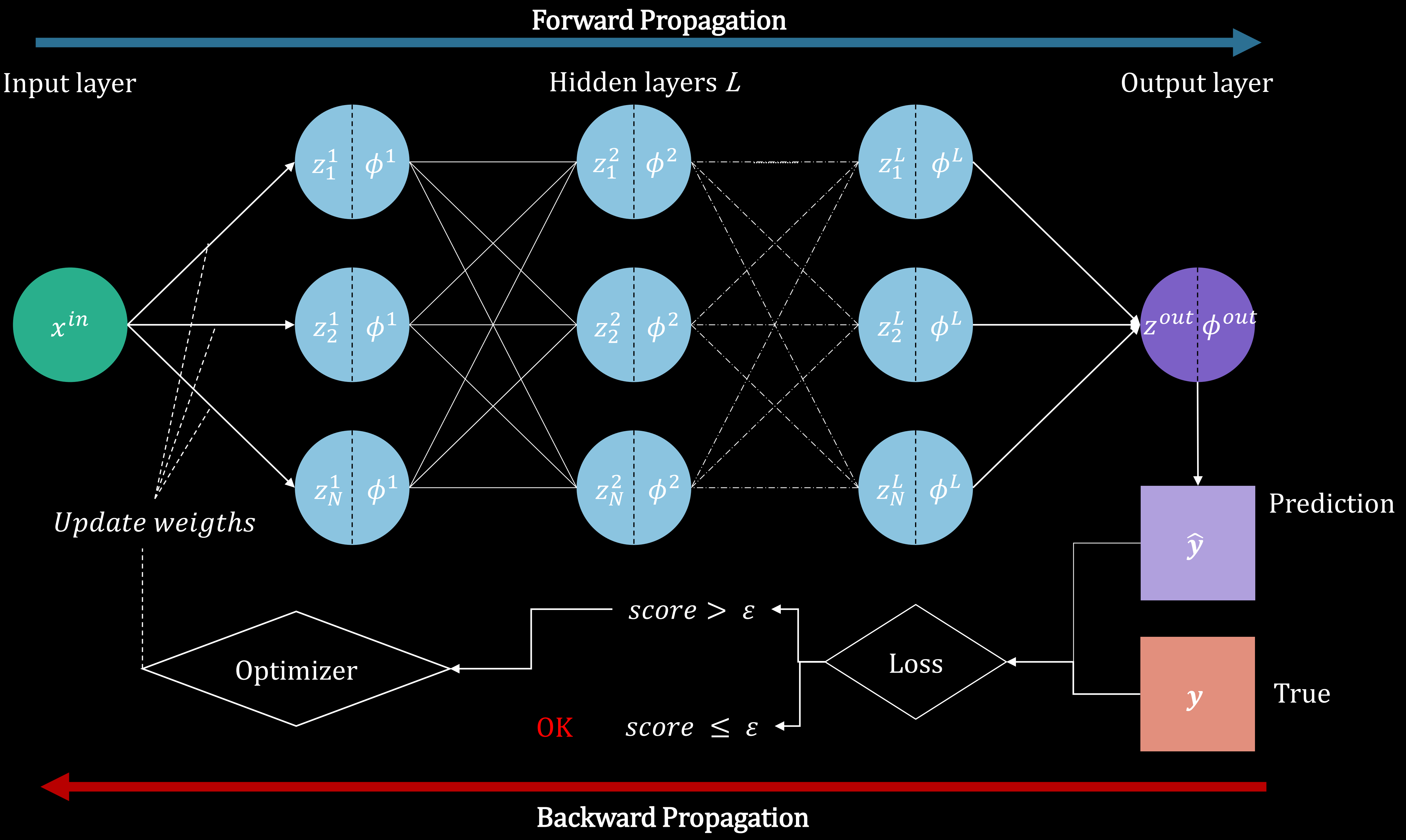
Back

Propagation

Backpropagation

Update $\boldsymbol{w}$ to minimize a loss function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ (*e.g.*, MSE, MAE, etc.)

1. At the output layer $\delta^L = \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}^L} \cdot \boldsymbol{a}^L$
2. For Hidden layers $\delta^l = \boldsymbol{W}^{(l+1)\,T} \delta^{(l+1)} \cdot \boldsymbol{a}^l$
3. Weight update: $\boldsymbol{W}^l \coloneqq \boldsymbol{W}^l - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^l}$
4. For Hidden layers $\boldsymbol{b}^l \coloneqq \boldsymbol{b}^l - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^l}$



Forward Propagation
Input layer
Hidden layers L
Output layer
x^{in}
z_1^1 ϕ^1
z_2^1 ϕ^1
z_N^1 ϕ^1
z_1^2 ϕ^2
z_2^2 ϕ^2
z_N^2 ϕ^2
z_1^L ϕ^L
z_2^L ϕ^L
z_N^L ϕ^L
z^{out} ϕ^{out}
Prediction
$\hat{y}$
True
y
Update weigths
Optimizer
$score > \varepsilon$
OK
$score \leq \varepsilon$
Loss
Backward Propagation

Physics Informed Neural Networks?

PINNs ~ class of NNs that incorporate physical laws directly in the loss function to guide the learning process

- Most used case: PDE, ODE solutions

- Some applications:
  - Fluid dynamics: NS fluid flow simulations, Heat fluxes, soil hydrodynamics, …
  - Solid mechanic: material stress and strains, …
  - Quantum mechanic: approx. Shrödinger's equation, …
  - Electromagnetics: solving Maxwell's equation, …

Advantages ✅

1. Integration of physics law (constraints)
2. Requires less data (+incomplete datasets)
3. Generalization (interpolation, extrapolation)

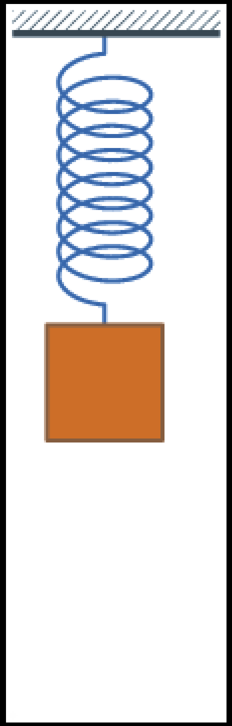
Disadvantages ❌

1. Computer intensive (GPU)
2. Requires knowledge of physics
3. Hyperparameter sensitivity

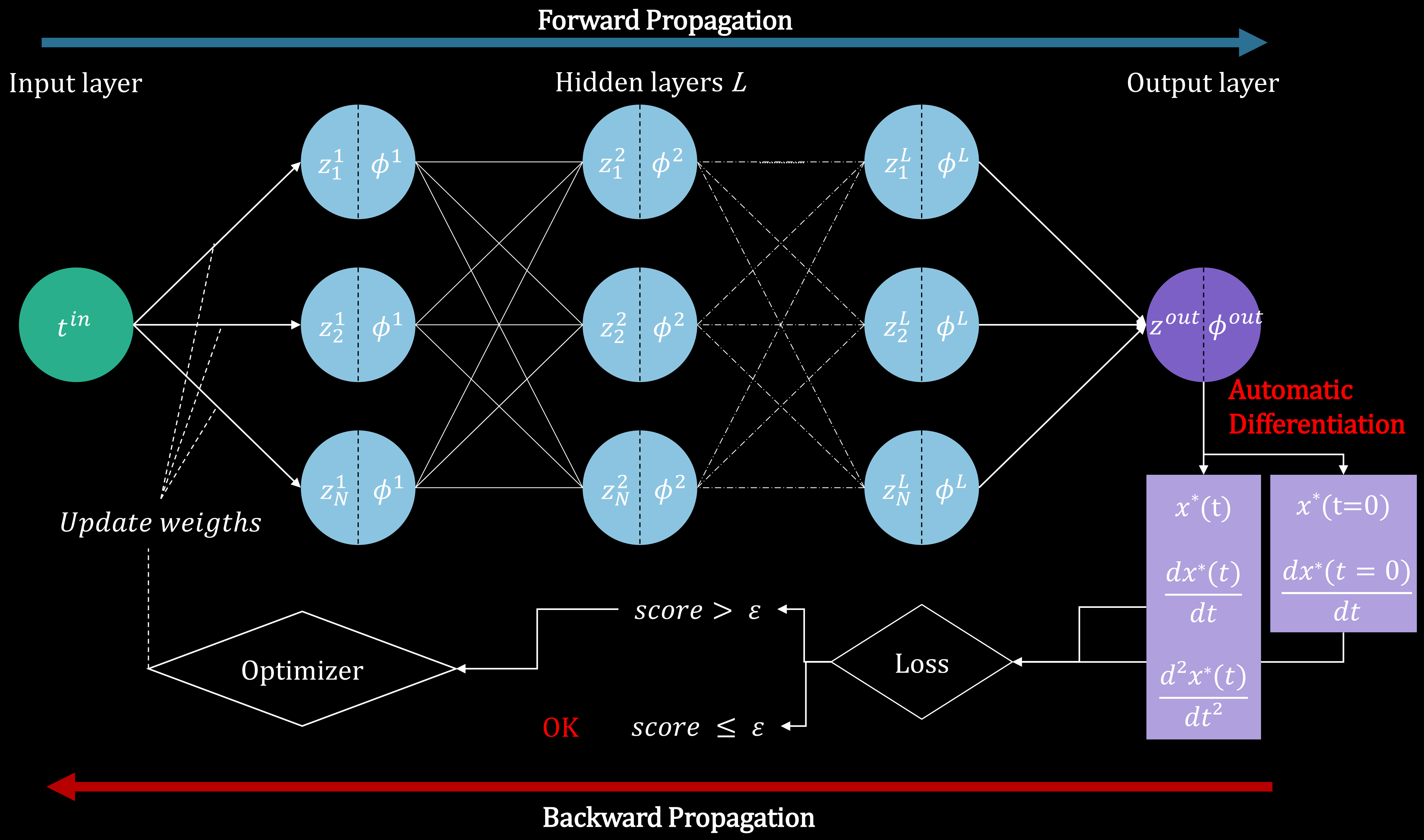
Physics Informed Neural Networks?

PINNs ~ class of NNs that incorporate physical laws directly in the loss function to guide the learning process

But let's try to build a PINN to solve the harmonic oscillator case



https://github.com/adil-thami/PINNs



Forward Propagation
Input layer
Hidden layers L
Output layer
t^{in}
z_1^1 ϕ^1
z_2^1 ϕ^1
z_N^1 ϕ^1
z_1^2 ϕ^2
z_2^2 ϕ^2
z_N^2 ϕ^2
z_1^L ϕ^L
z_2^L ϕ^L
z_N^L ϕ^L
z^{out} ϕ^{out}
Automatic Differentiation
$x^*(\mathrm{t})$
$\frac{dx^*(t)}{dt}$
$\frac{d^2x^*(t)}{dt^2}$
$x^*(\mathrm{t{=}0})$
$\frac{dx^*(t=0)}{dt}$
Update weigths
$score > \varepsilon$
Optimizer
Loss
OK
$score \leq \varepsilon$
Backward Propagation