Physical Symmetries Embedded in Neural Networks

Pavlos Protopapas

Marios Mattheakis, David Sondak, D. Randle, M. Di Giovanni, E. Kaxiras

School of Engineering and Applied Science

Outline

1. Motivation
    1. Training Set
    2. Stellar Formation
    3. Fluid dynamics
2. NN to Solve Differential Equations
    1. Supervised
    2. Unsupervised
3. Physical Symmetries Embedded in Neural Networks
    1. Constraint …
    2. Sympletic Approach

Motivation

Training Data

Anima:

What does a training set look like?

CAT

DOG

DOG

CAT

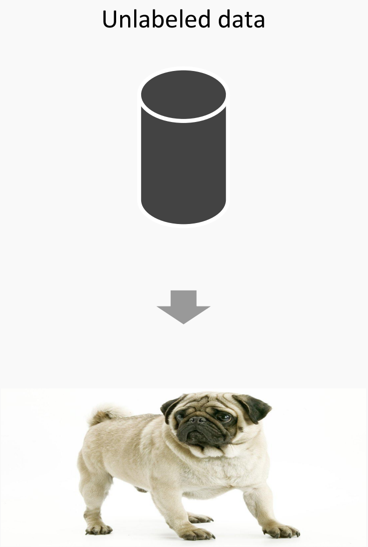
To classify we need .....

Training Sets

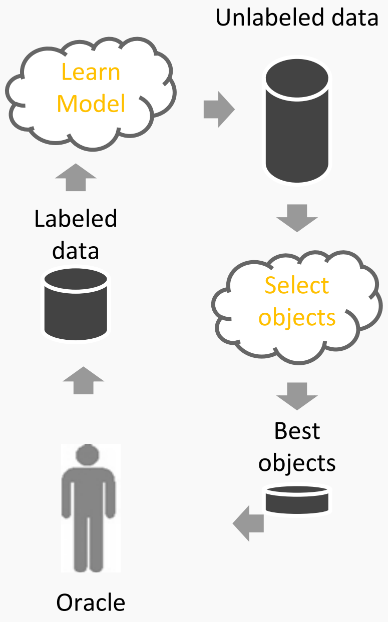
Labeling objects can be extremely hard

How can we create training sets?

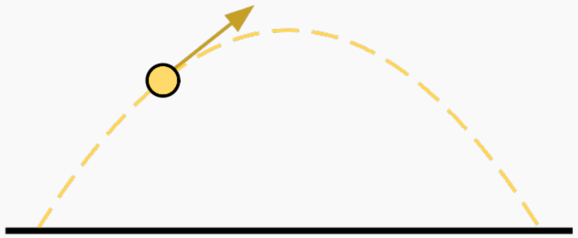
Manually Visual Inspection



Active Learning

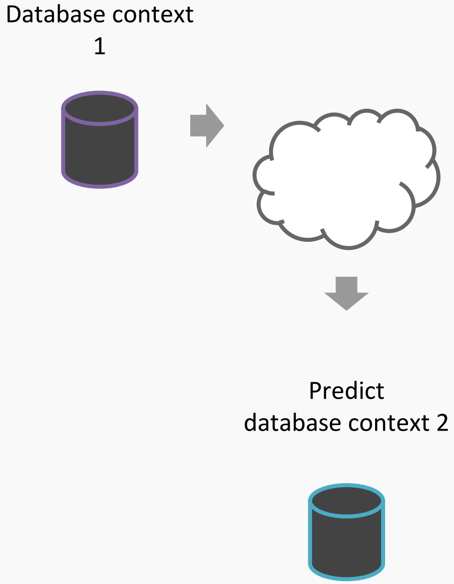


Synthetic Data Augmentation Generation



$$v = \sqrt{v_x^2 + v_y^2}$$

Transfer Learning



Chapter 2: Stellar Formation

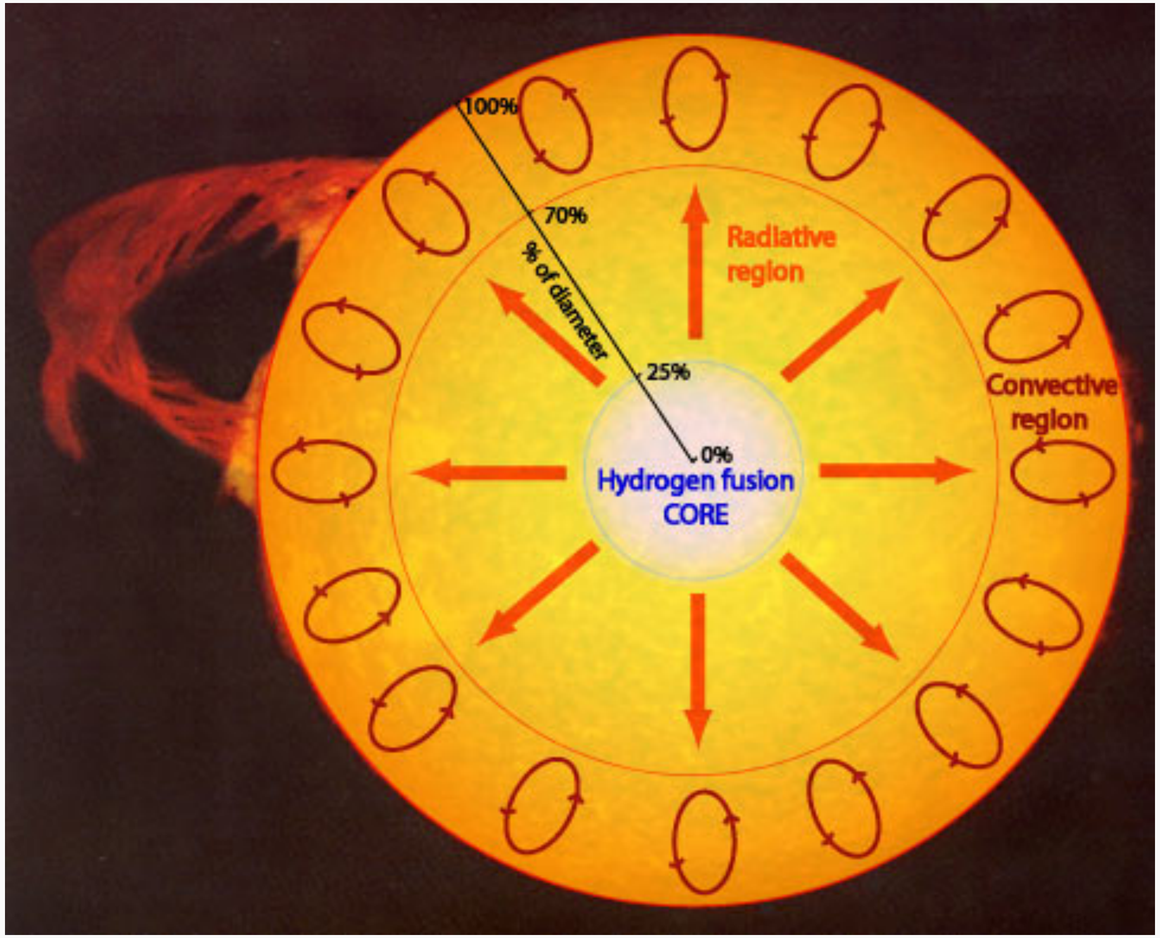
Stellar Formation

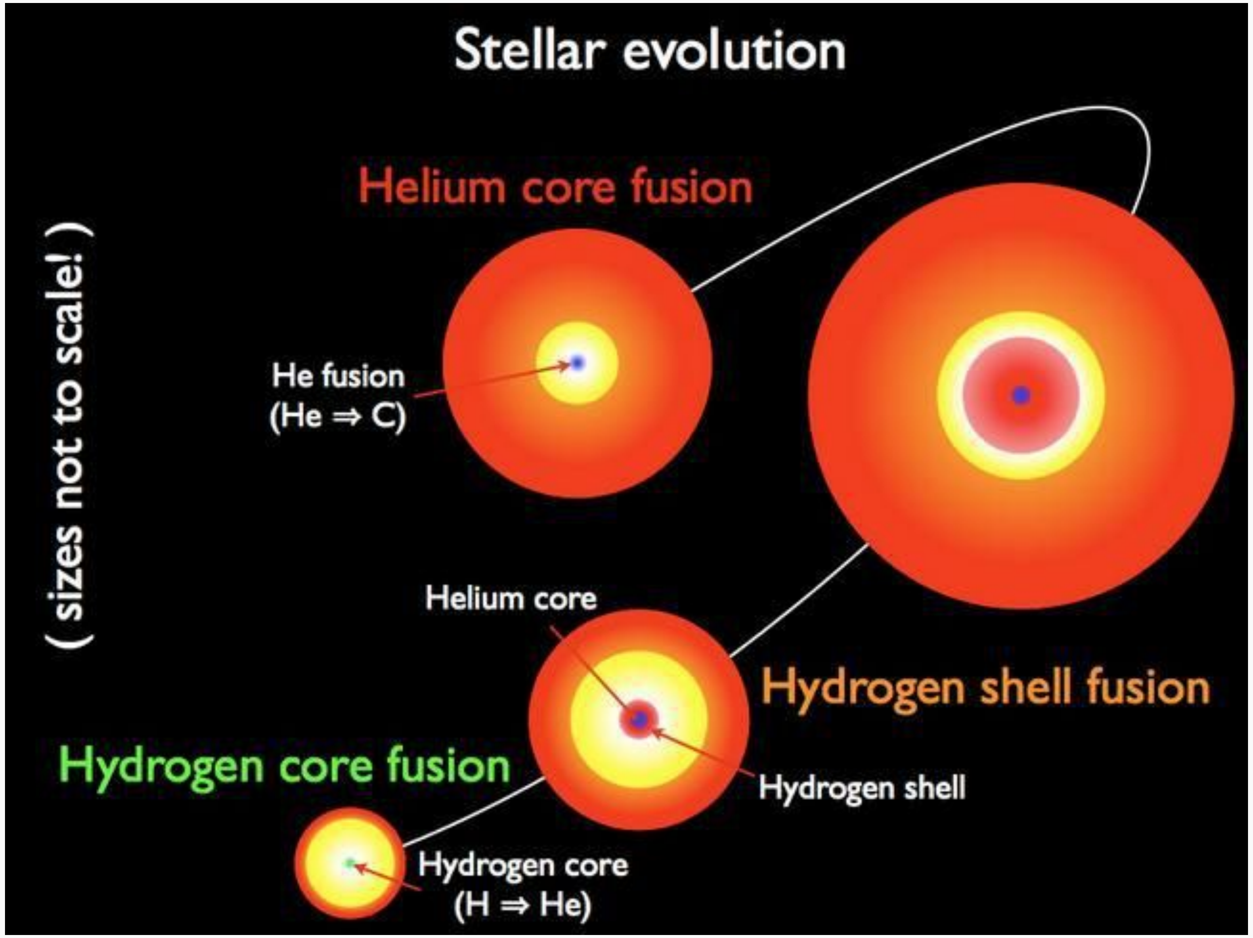
Pillars of Creation

High resolution optical (left) and infrared (right) images, HST 2014.

Stellar Evolution

From collapse to nuclear burning

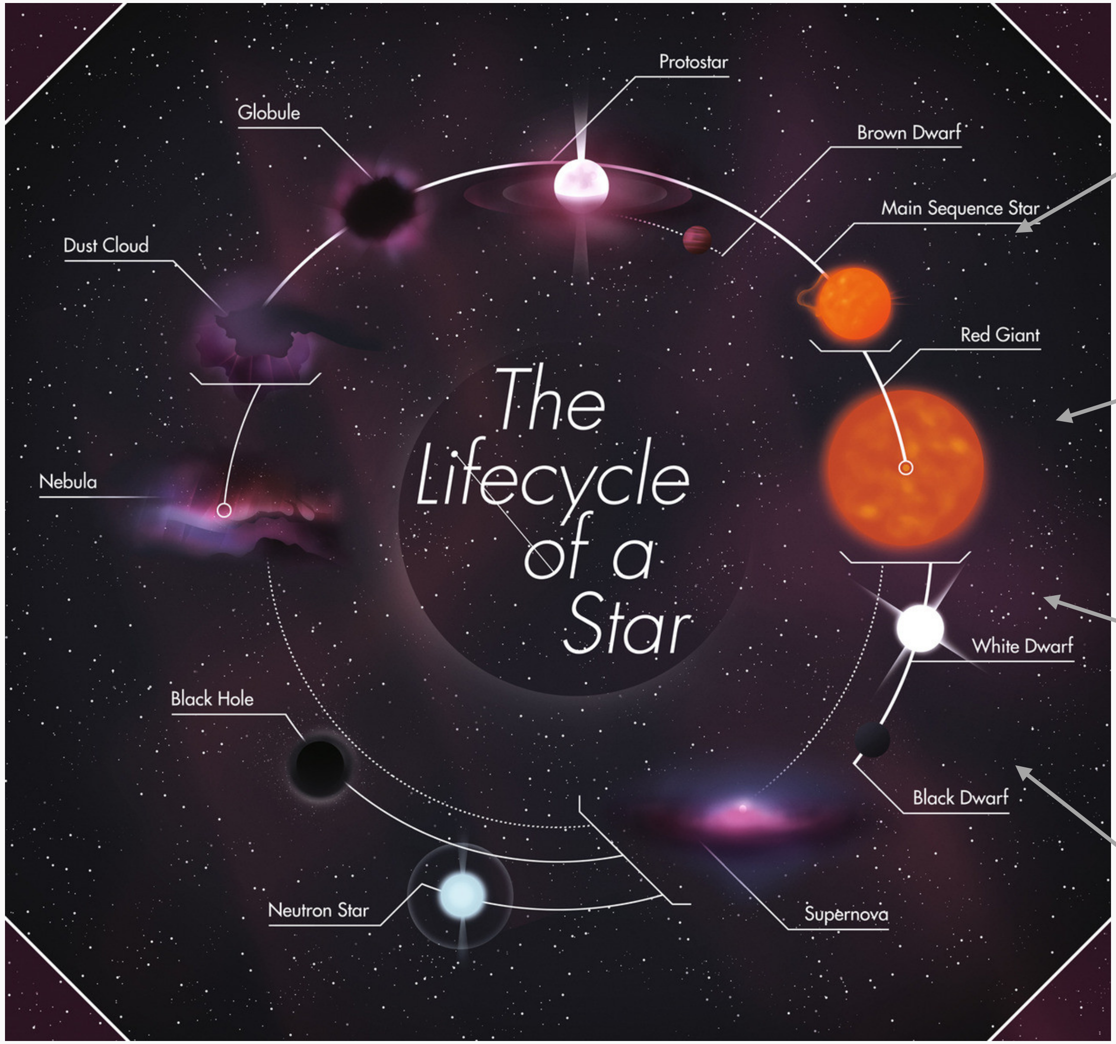




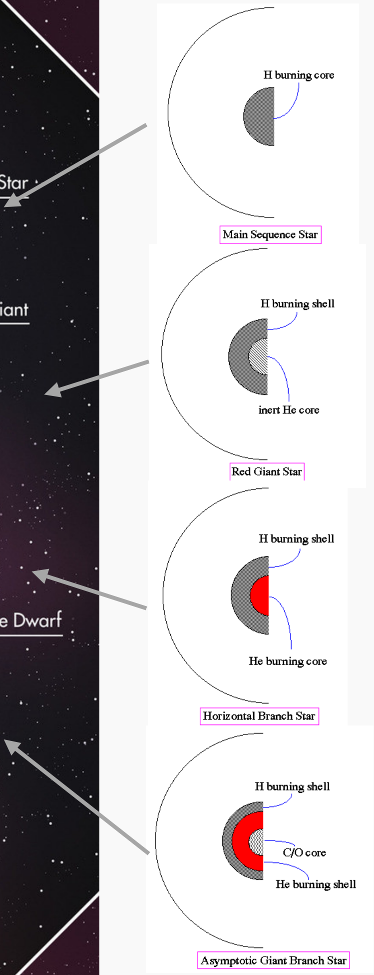
Naval Research Laboratory

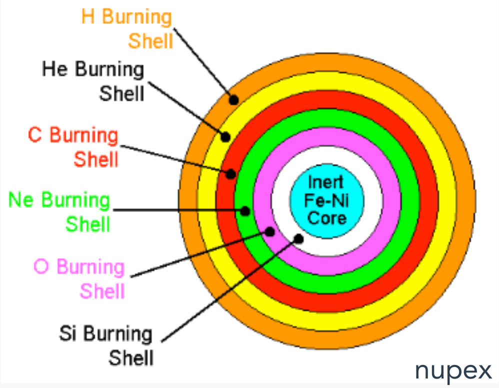
Stellar Evolution

Cycle from inside out



futurism.com

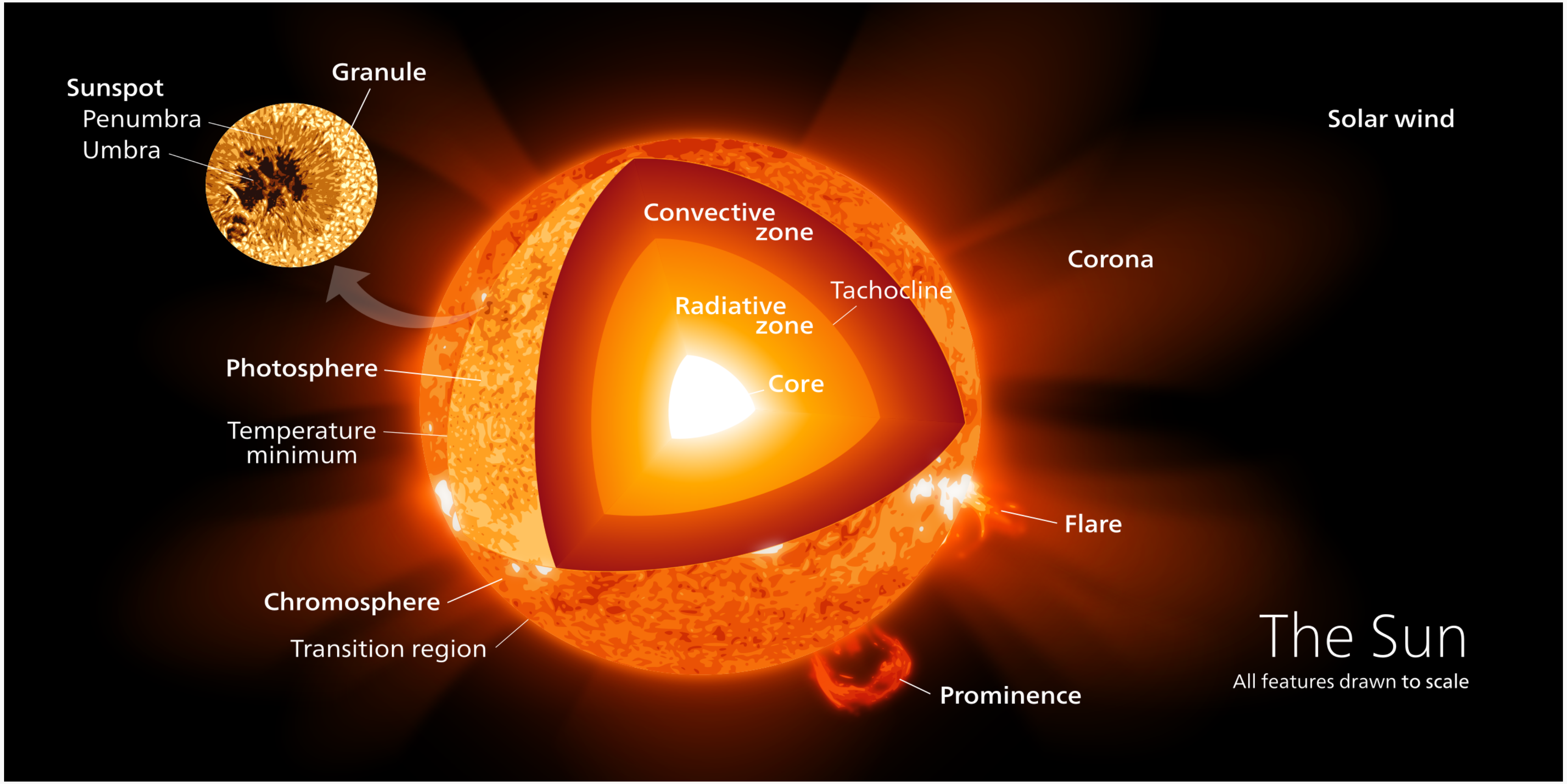




nupex

Stellar Interior

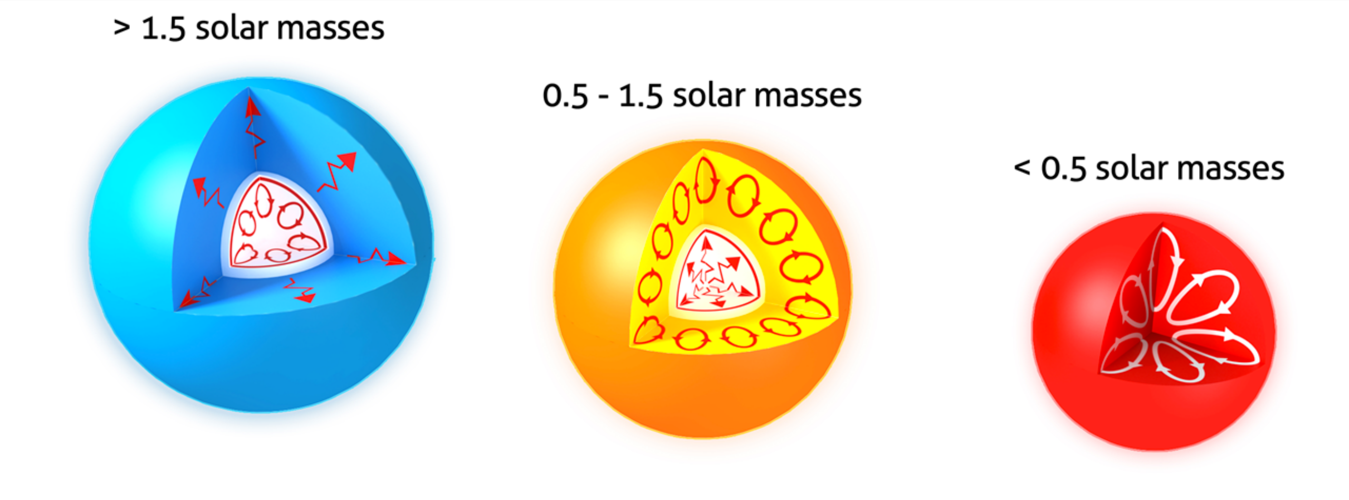
Sun-like Star

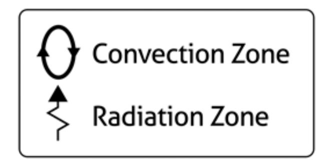


Stellar Interior

Across stellar mass

Transport of Energy is the key





Stellar Interior

Solving convection

The equations reflect basic physics laws: conservation of mass, momentum, and energy:

$$\frac{\partial}{\partial t}\rho = -\vec{\nabla}\cdot(\rho\vec{u}),$$
$$\frac{\partial}{\partial t}\rho\vec{u} = -\vec{\nabla}\cdot(\rho\vec{u}\otimes\vec{u}) - \vec{\nabla}p + \rho\vec{g},$$
$$\frac{\partial}{\partial t}\rho\epsilon_t = -\vec{\nabla}\cdot(\rho\epsilon_t\vec{u} + p\vec{u}) + \rho\vec{u}\cdot\vec{g} + \vec{\nabla}\cdot(\chi\vec{\nabla}T) - q$$

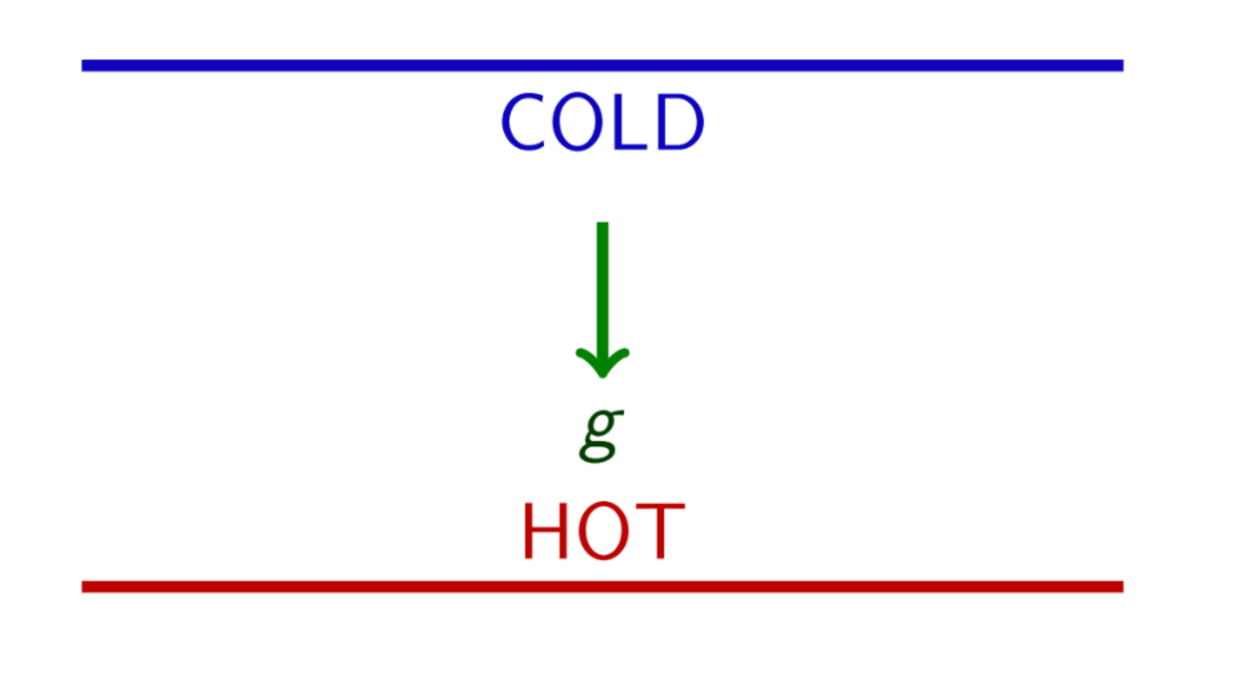
But they are expensive to solve, computationally: they need to be solved in three dimensions over a huge range of length scales and time scales, and of pressures, densities and temperatures

Outline

1. Motivation
   1. Training Set
   2. Stellar Formation
   3. Fluid dynamics
2. NN to Solve Differential Equations
   1. Supervised
   2. Unsupervised
3. Physical Symmetries Embedded in Neural Networks
   1. Constraint …
   2. Sympletic Approach

Fluids

τα πάντα ρει



Cold fluid falls, hot fluid rises

$$\frac{\partial T}{\partial t} = \underbrace{-\nabla \cdot (\boldsymbol{u}T)}_{\text{Convection}} + \underbrace{k\nabla^2 T}_{\text{Conduction}}$$

The full equation

Navier-Stokes equation for the velocity field $\boldsymbol{u}$

Conservation of momentum:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\frac{1}{\rho} \nabla P + \nu \nabla^2 \boldsymbol{u} + f$$

Conservation of mass:

$$\nabla \cdot \boldsymbol{u} = 0$$

ν: viscosity
ρ: density

Turbulence

Disordered fluid flow

High dimensional chaos

Multiscale phenomenon

$$\frac{\partial}{\partial t}\rho = -\vec{\nabla}\cdot(\rho\vec{u}),$$
$$\frac{\partial}{\partial t}\rho\vec{u} = -\vec{\nabla}\cdot(\rho\vec{u}\otimes\vec{u}) - \vec{\nabla}p + \rho\vec{g},$$
$$\frac{\partial}{\partial t}\rho\epsilon_t = -\vec{\nabla}\cdot(\rho\epsilon_t\vec{u} + p\vec{u}) + \rho\vec{u}\cdot\vec{g} + \vec{\nabla}\cdot(\chi\vec{\nabla}T) - q$$

Can we "solve" it?

Numerically yes but way too expensive for stars

We do approximations! A lots and lots of approximations ....

Outline

1. Motivation
    1. Training Set
    2. Stellar Formation
    3. Fluid dynamics
2. **NN to Solve Differential Equations**
    1. Supervised
    2. Unsupervised
3. Physical Symmetries Embedded in Neural Networks
    1. Constraint …
    2. Sympletic Approach

NN to Solve Differential Equations

Supervised vs Unsupervised

Supervised:

- Solve Reynolds-averaged Navier Stokes Equations (RANS)
- Solve the full NS equation

Unsupervised:

- Simulate the whole thing using Deep Neural Networks

Solving differential equations using NN: setup

Express the differential equation as:

$$f\left(x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, \dots, \lambda\right) = 0$$

and initial and/or boundary conditions.

Find:

$$x = g(t)$$

that *satisfies* the differential equation and the initial or boundary conditions.

Harmonic Oscillator:

$$\frac{d^2x}{dt^2} + kx = 0$$

Initial value:

$$x(t=0) = x_0$$

$$x = g(t) = x_0 + \sin(kt)$$

Unless we know the exact solution, here we need to specify what we mean by *satisfies*.

Solving differential equations: **Supervised** learning

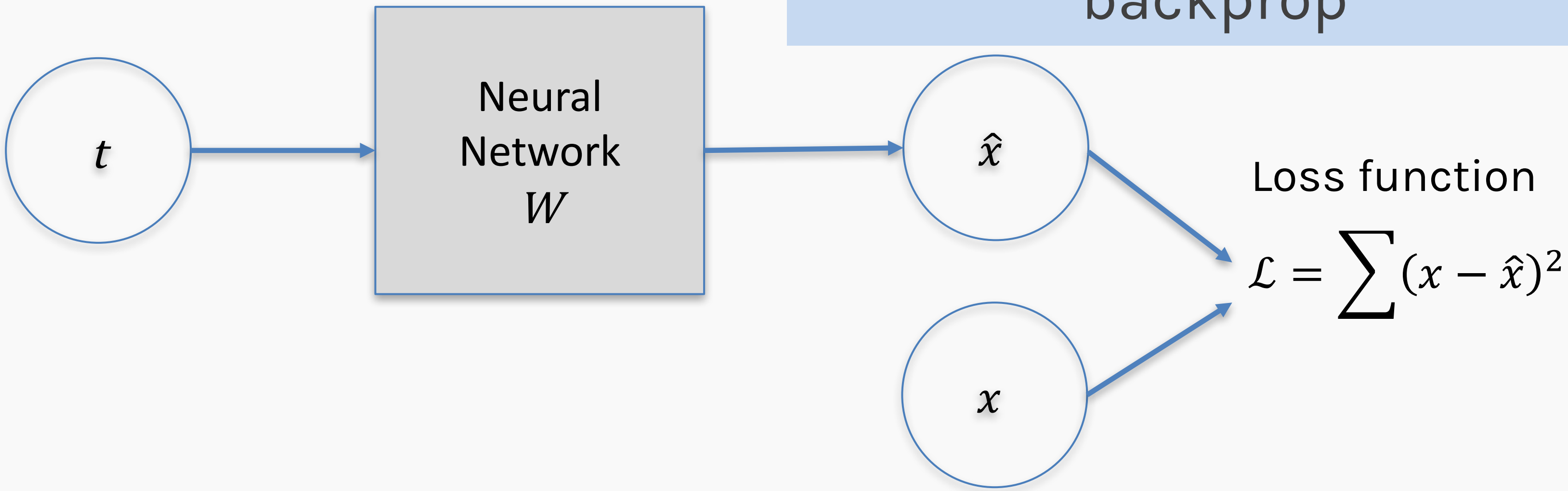
For some examples

$$\{(x_1, t_1), \dots, (x_n, t_n)\}$$

Find:

Estimate $g(t)$ with $\hat{g}(t)$ st $\hat{g}(t_i)$ is as close to x_i as possible.

Learn the weights using backprop



Solving differential equations: **Supervised** learning (cont)

Implementation:

Avoid overfitting by regularizing

Hyper-parameter (number of neurons, activation functions, optimization etc) with cross validation

Pros:

Relatively easy to set up

Cons:

We need training examples which can be very expensive to get.

Solving differential equation: **Unsupervised** Leaning

Remember:

We expressed the differential equation as:

$$f\left(x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, \dots, \lambda\right) = 0$$
$$x(t=0) = x_0$$

…

The goal is to find the mapping from t to x:

$$x = g(t)$$

such that:

$$f\left(x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, \dots, \lambda\right) = 0 \qquad x(0) = x_0$$

2

Unsupervised: Solving differential equation

Let's look at a simple example:

$$\frac{dx}{dt} = \lambda x, \qquad x(0) = x_0$$

We can express the differential equation simply as:

$$f\left(x, \frac{dx}{dt}, \lambda\right) = \frac{dx}{dt} - \lambda x = 0$$

The goal is to find the mapping from t to x:

$$x = g(t)$$

such that:

$$\left(\frac{dx}{dt} - \lambda x\right)^2 = 0 \qquad x(0) = x_0$$

In this case we know the exact solution: $x = x_0 e^{\lambda t}$. We can use it for evaluation only.

Unsupervised: Solving differential equation (cont.)

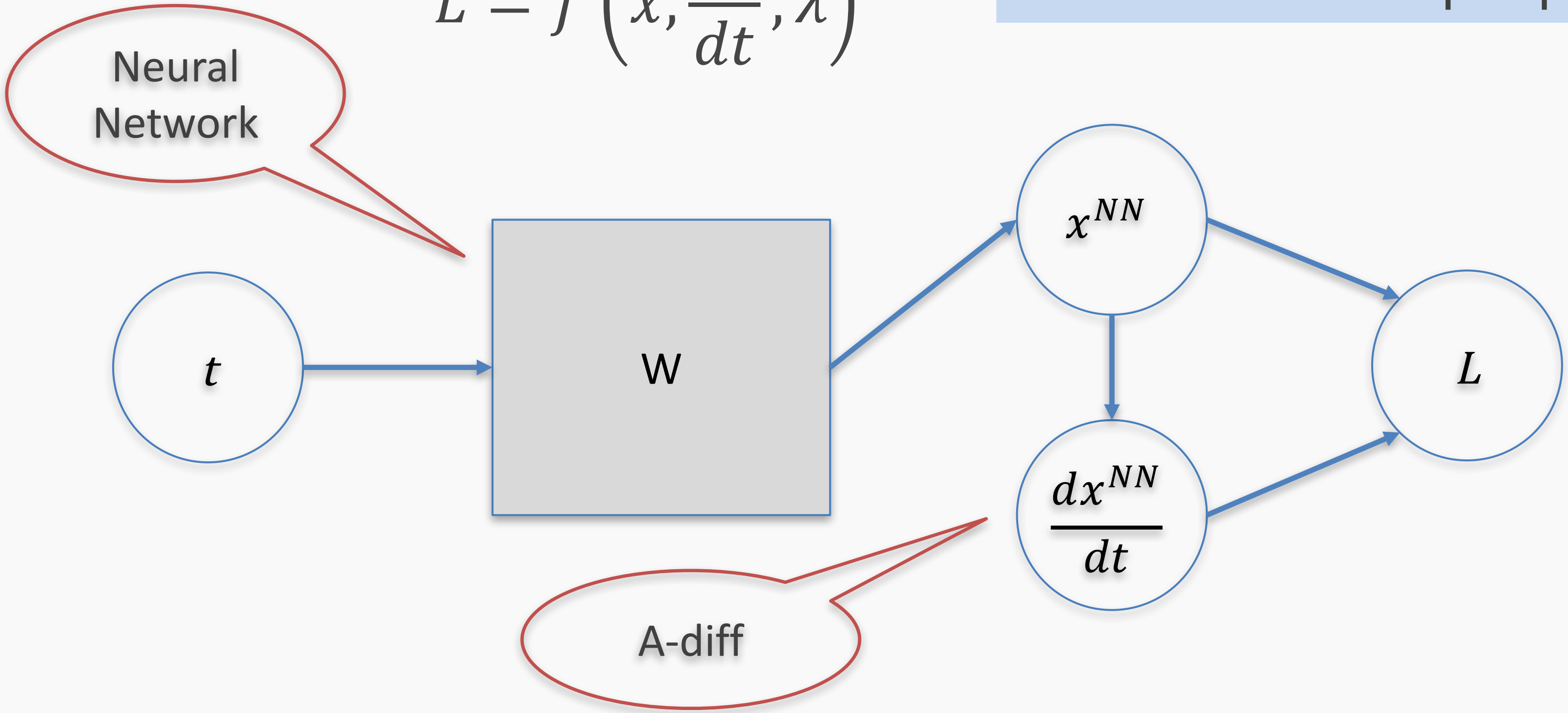
Find a function:

$$x = g(t)$$

Minimizing a loss function:

$$L = f\left(x, \frac{dx}{dt}, \lambda\right)^2$$

Learn the weights using backprop



Unsupervised: Solving differential equation (cont.)

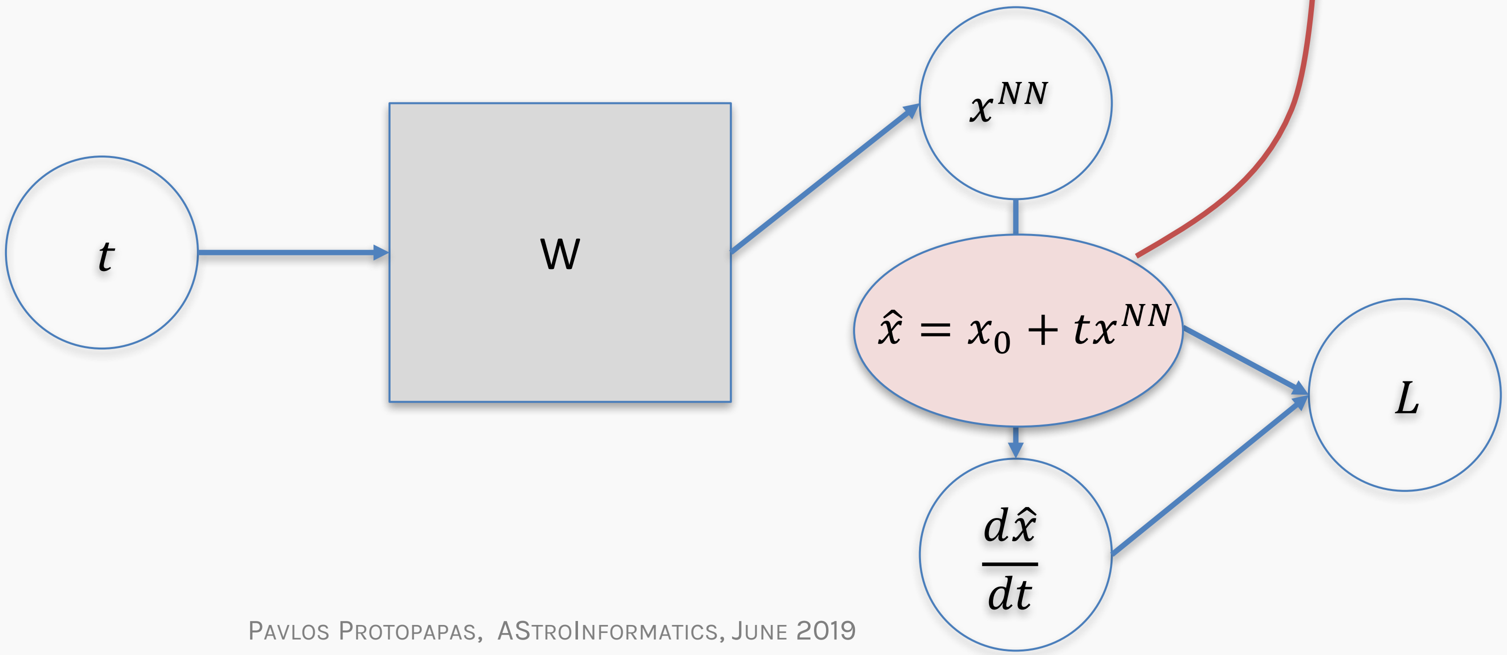
Find a function:

$$x = g(t)$$

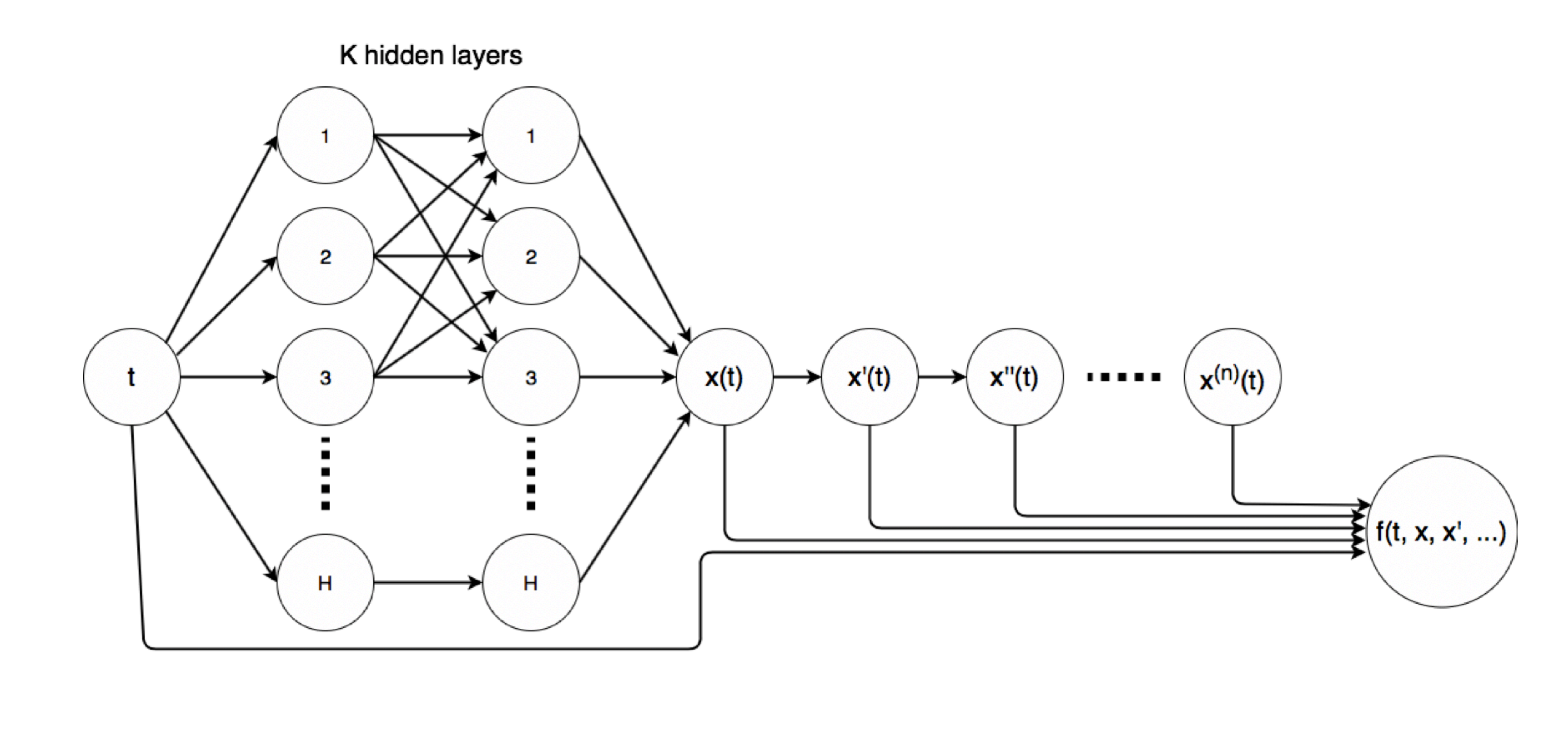
Minimizing a loss function:

$$\boxed{\hat{x} = x_0 + (1 - e^{-t})x^{NN}}$$

$$L = f\left(x, \frac{dx}{dt}, \lambda\right)^2$$



Unsupervised: Solving differential equation (cont.)



Some references

Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations, ***Yiping Lu, Aoxiao Zhong, Quanzheng Li, Bin Dong***

http://proceedings.mlr.press/v80/lu18d.html

Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations, ***Liu Yang, Dongkun Zhang, George Em Karniadakis***

https://arxiv.org/abs/1811.02033

A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, ***M.Raissi, P.Perdikaris, G.E.Karniadakis***

https://www.sciencedirect.com/science/article/pii/S0021999118307125

Physical Symmetries Embedded in Neural Networks, M***. Mattheakis , P. Protopapas , D. Sondak , M. Di Giovanni , E. Kaxiras***

https://arxiv.org/pdf/1904.08991.pdf

Outline

1. Motivation
   1. Training Set
   2. Stellar Formation
   3. Fluid dynamics
2. NN to Solve Differential Equations
   1. Supervised
   2. Unsupervised
3. Physical Symmetries Embedded in Neural Networks
   1. Constraint …
   2. Sympletic Approach

Physical Symmetries Embedded in Neural Networks

Motivation

- Neural Networks (NNs) are natively *physics-agnostic*:
  - Fit data without respecting the underlying physical laws
  - Approximate solutions that are not physically accepted

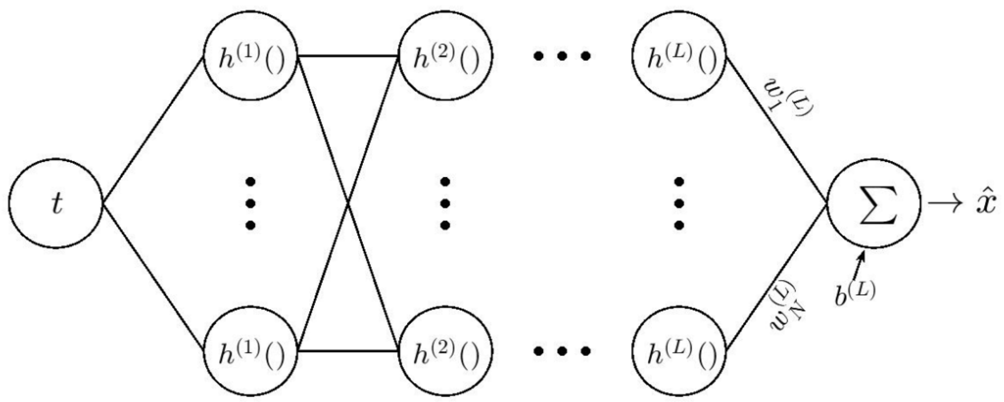
➢ Data augmentation
➢ Filtering the not physically accepted predictions
➢ Imposing physics through regularization

...not enough to respect the physics

- **Supervised** Neural Network:
  - Impose symmetries in the NN’s structure
  - Predictions with a certain symmetry
- **Unsupervised** Neural Network:
  - Energy conservation (new architecture)

- **Supervised** Neural Network:
  - Impose symmetries in the NN's structure
  - Predictions with a certain symmetry
- **Unsupervised** Neural Network:
  - Energy conservation (new architecture)

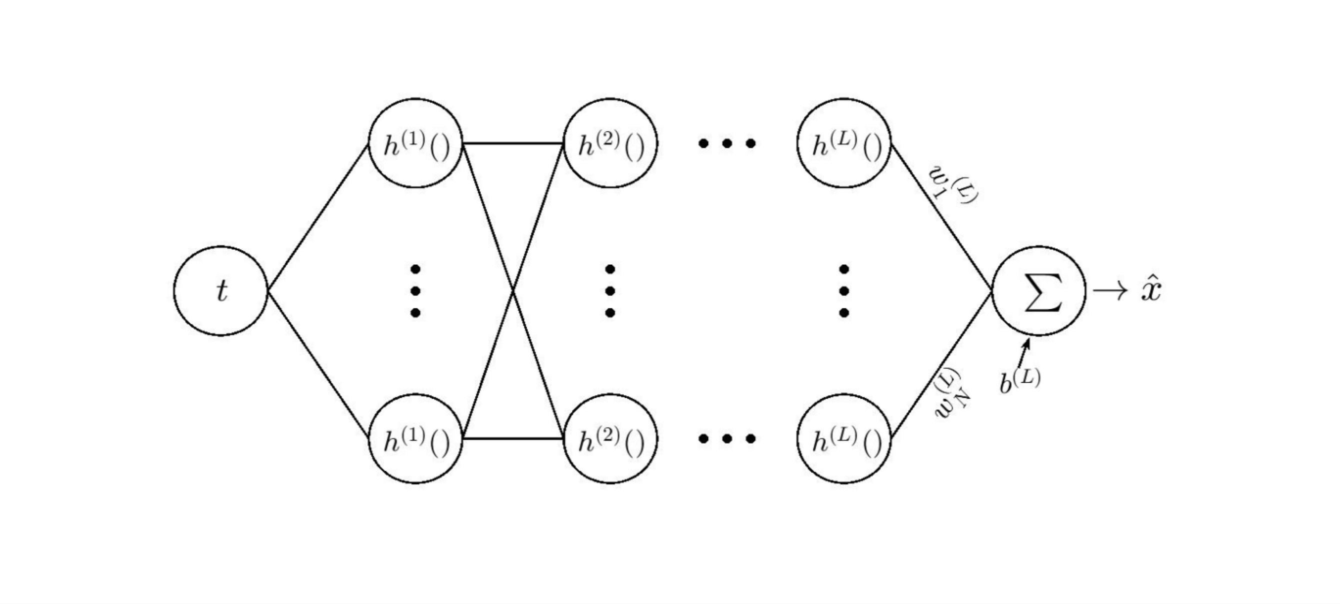
Impose odd/even symmetry in the NN structure



Forward propagation:

$$\hat{x}(t) = \sum_{i=1}^{N} w_i^{(L)} \; h_i^{(L)}(t) + b^{(L)}$$

Impose odd/even symmetry in the NN structure (cont)



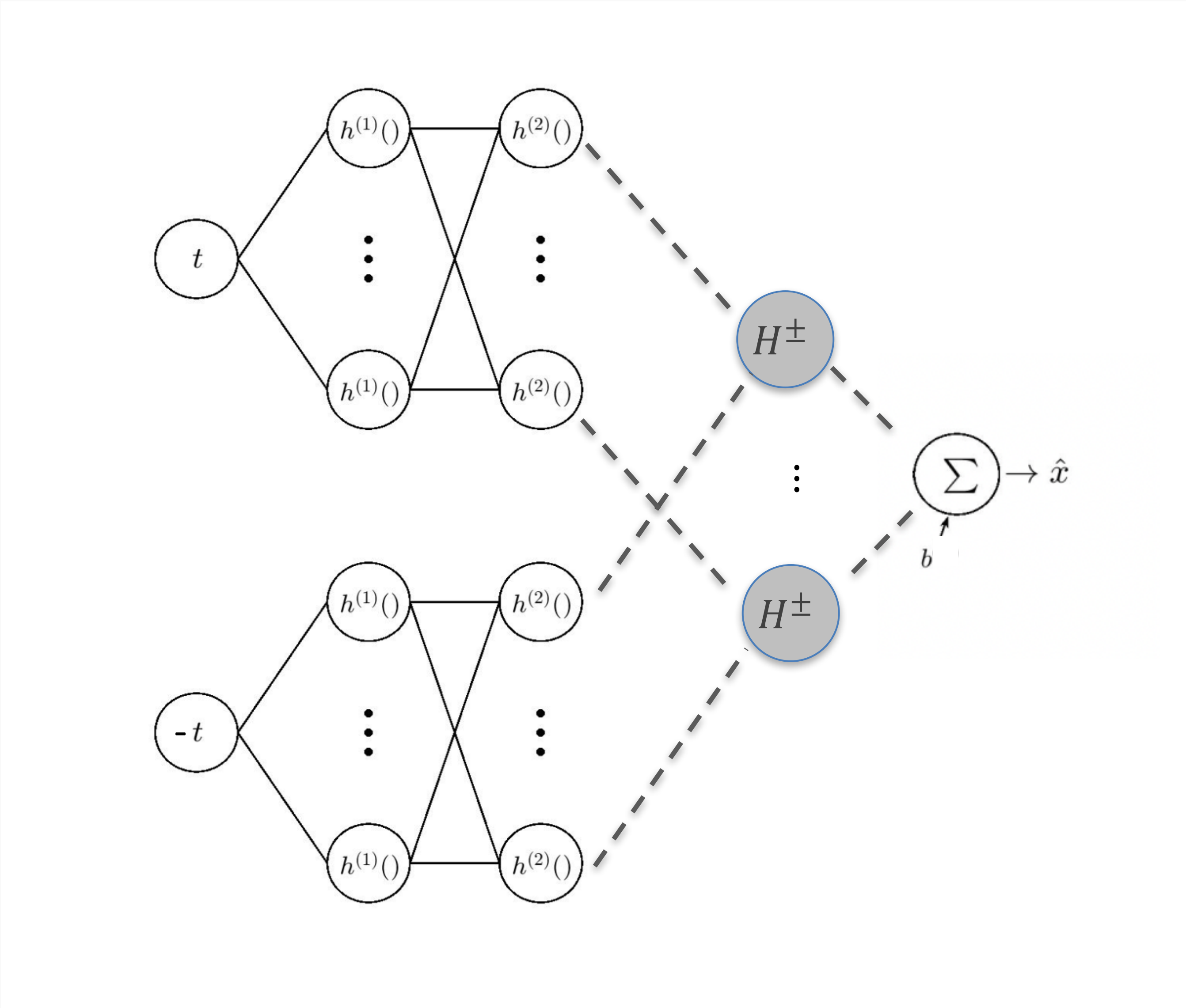
Decompose in even and odd parts

$$\hat{x}(t) = \frac{1}{2}\left(\sum_{i=1}^{N} w_i H_i^{+} + 2b\right) + \frac{1}{2}\left(\sum_{i=1}^{N} w_i H_i^{-}\right)$$

Hub Neurons

$$H_i^{\pm} = h_i(t) \pm h_i(-t)$$

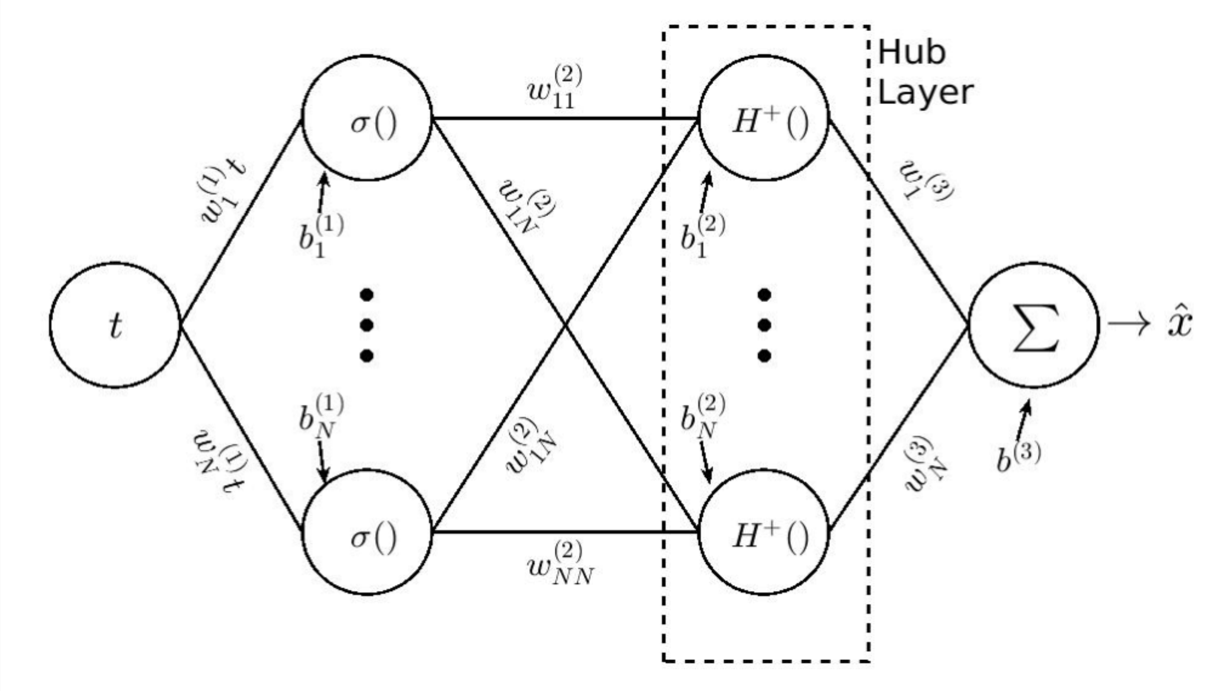
Impose odd/even symmetry in the NN structure (cont)



$$H_i^{\pm} = h_i(t) \pm h_i(-t)$$

Impose odd/even symmetry in the NN structure (cont)

Employ a two layer NN with N = 5 sigmoid and hub neurons in each layer



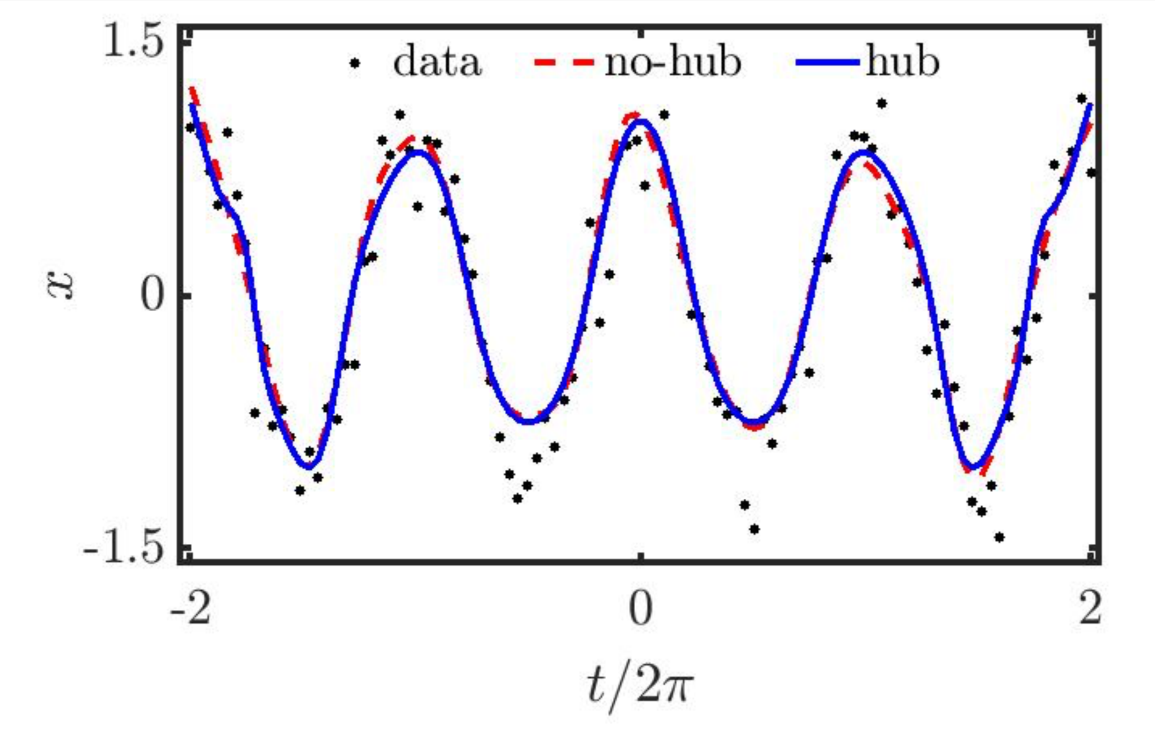
Even hub NN

$$\hat{x}(t) = \sum_{i=1}^{N} w_i^{(3)} H_i^+ + b^{(3)}$$

Regression: odd/even symmetry in the NN

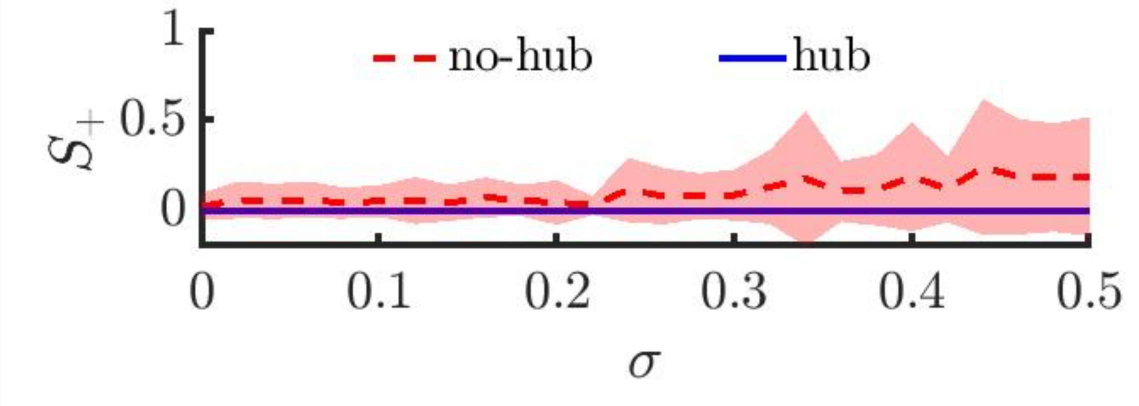
$$x(t) = \cos(t) + \epsilon$$

$$(\epsilon \sim \mathcal{N}(0, \sigma))$$



Measure the deviation from even symmetry with:

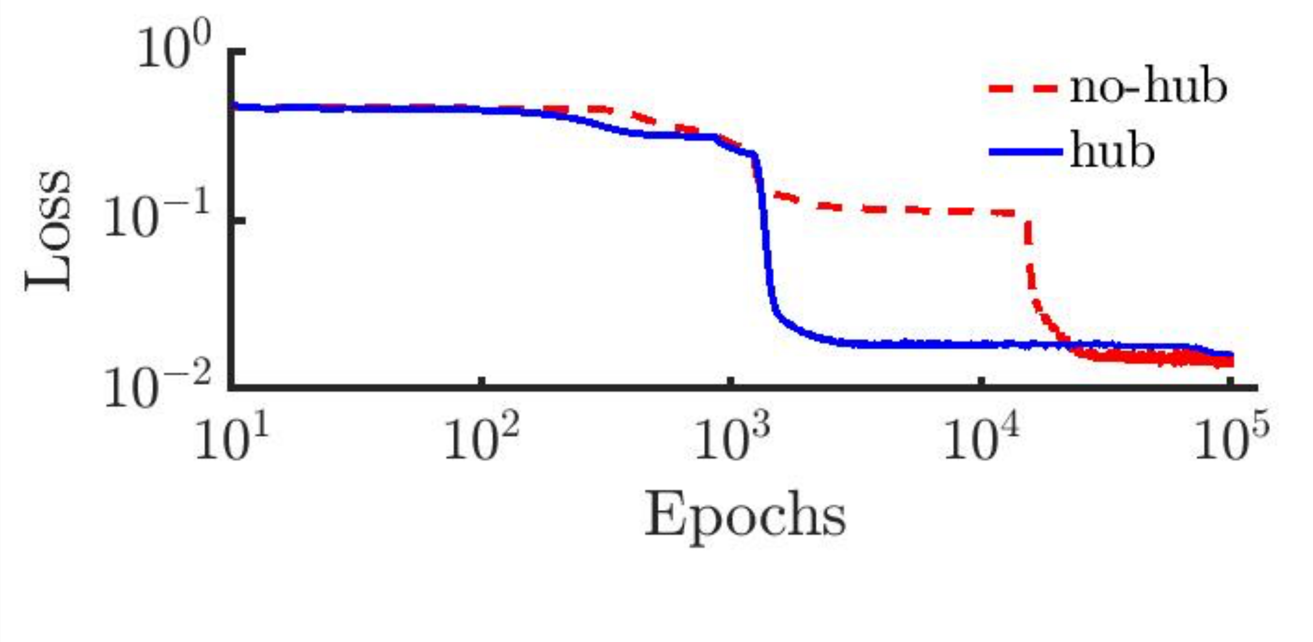
$$S_+ = \frac{1}{M}\sum_{i=1}^{M}\left(\hat{x}(t_i) - \hat{x}(-t_i)\right)^2$$



30 training sets for each σ

Regression: odd/even symmetry in the NN (cont)

MSE loss for training set.



- ✓ More efficient training due to reduced solutions
- ✓ Protect from over fitting

$$\epsilon \sim \mathcal{N}(0, \sigma = 0.2)$$

- **Supervised** Neural Network:
  - Impose symmetries in the NN’s structure
  - Predictions with a certain symmetry
- **Unsupervised** Neural Network:
  - Energy conservation (new architecture)

Energy conservation

Many physical systems are governed by differential equations derived from conservation of energy.

We use the hub layer idea to embed energy conservation into the NN

This NN guarantees that solution trajectories conserve energy

The starting point is Hamilton's equations

$$\dot{q}_k = p_k \qquad \dot{p}_k = -\frac{\partial}{\partial q_k} V(\boldsymbol{q})$$

Energy conservation

The starting point is Hamilton's equations

$$\dot{q}_k = p_k \qquad \dot{p}_k = -\frac{\partial}{\partial q_k} V(\boldsymbol{q}) \qquad k = 1, \dots, d$$

where

- $\boldsymbol{q}(t) = (\mathrm{q}_1(\mathrm{t}), \dots, q_d(t))$ and $\boldsymbol{p}(t) = (\mathrm{p}_1(\mathrm{t}), \dots, p_d(t))$ are the position and momentum, respectively
- $V(\boldsymbol{q})$ is the potential
- $(\dot{.}) = \frac{d}{dt}(.)$

Impose $\dot{q}_k = p_k$ and minimize $\dot{p}_k = -\frac{\partial}{\partial q_k} V(\boldsymbol{q})$ In the loss function

Choose the parametrization:

$$\hat{\boldsymbol{q}} = \boldsymbol{q_0} + t\,\boldsymbol{p_0} + e^{t-t_0}\boldsymbol{N}$$

$$\hat{\boldsymbol{p}} = \boldsymbol{p_0} + e^{t-t_0}\widetilde{\boldsymbol{N}}$$

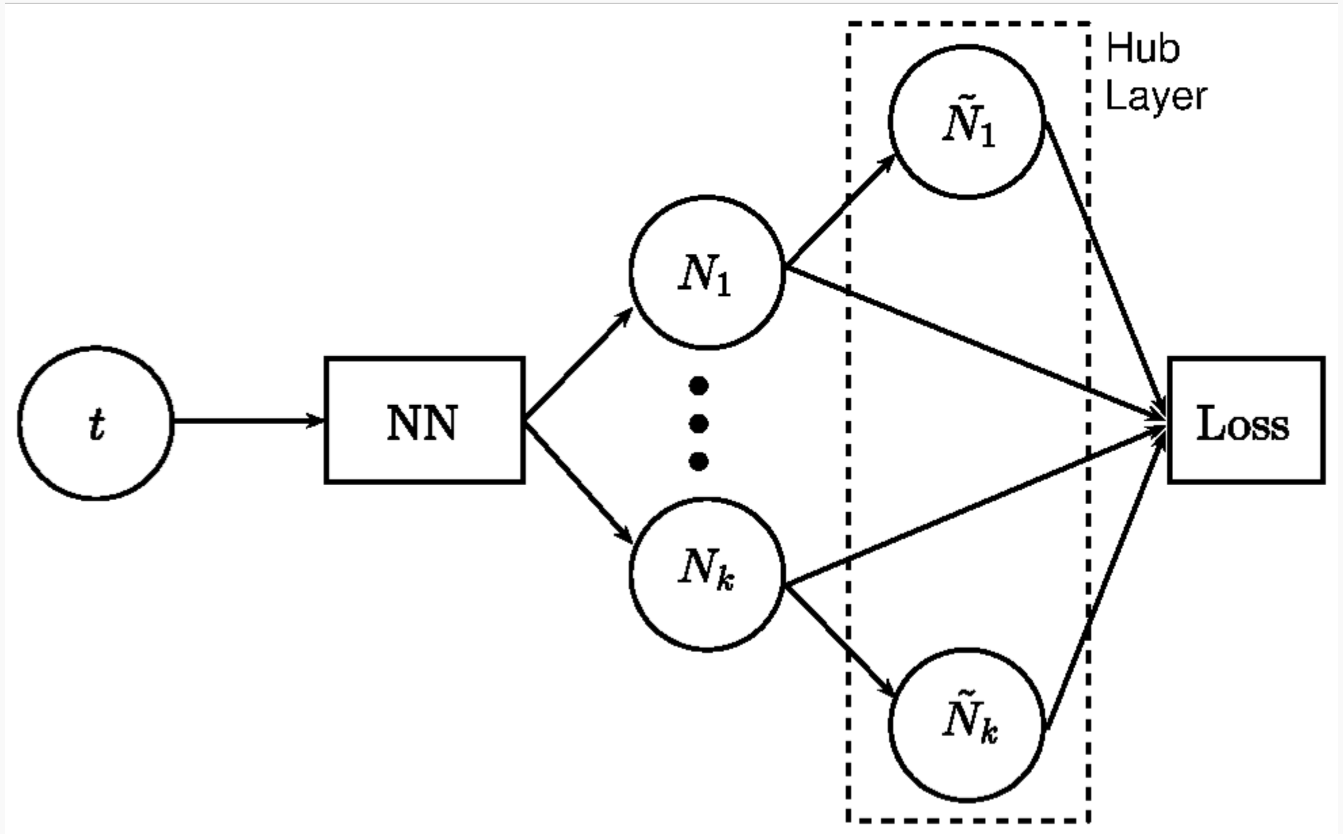
Impose $\dot{q}_k = p_k$ as a constrain, this yields an expression for the hub neuron,

$$\widetilde{\boldsymbol{N}} = [1 - e^{-t}]\dot{\boldsymbol{N}} + 2e^{[1-e^{-t}]}\boldsymbol{N}$$

Loss function is given by:

$$L = \sum_k \left(\dot{\hat{p}}_k + \frac{\partial V(\boldsymbol{q})}{\partial q_k} \right)^2$$

NN architecture:

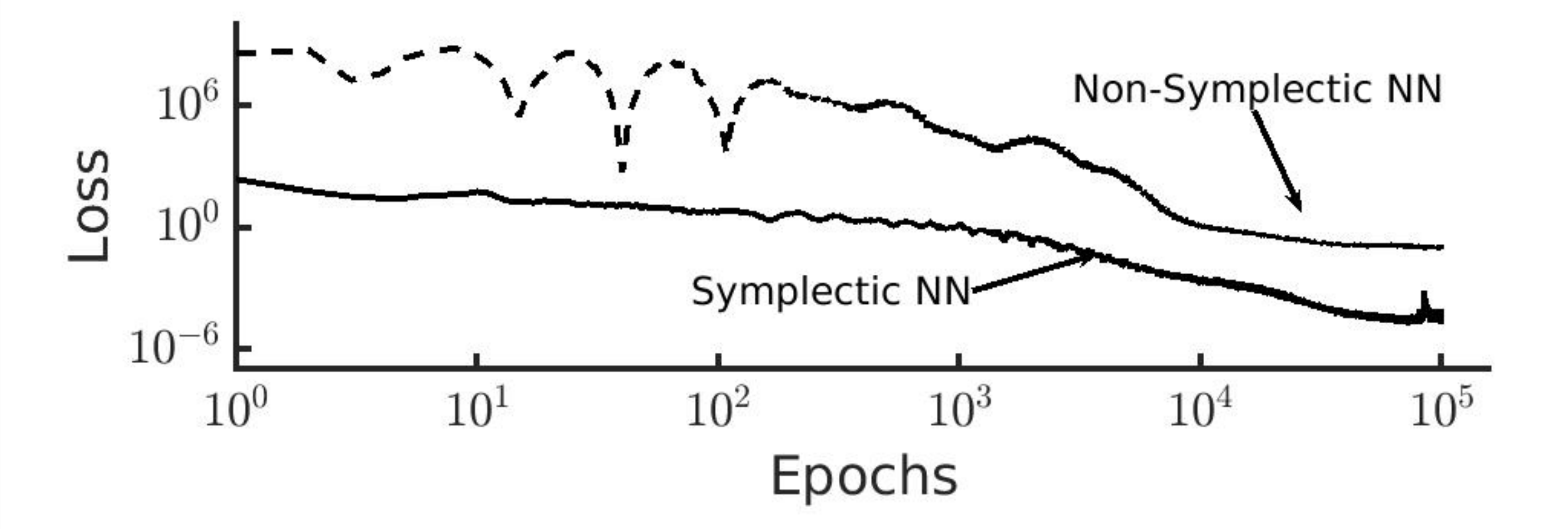


Loss function is given by:

$$L = \sum_k \left(\dot{\hat{p}}_k + \frac{\partial V(\boldsymbol{q})}{\partial q_k} \right)^2$$

Henon-Heiles Hamiltonian System

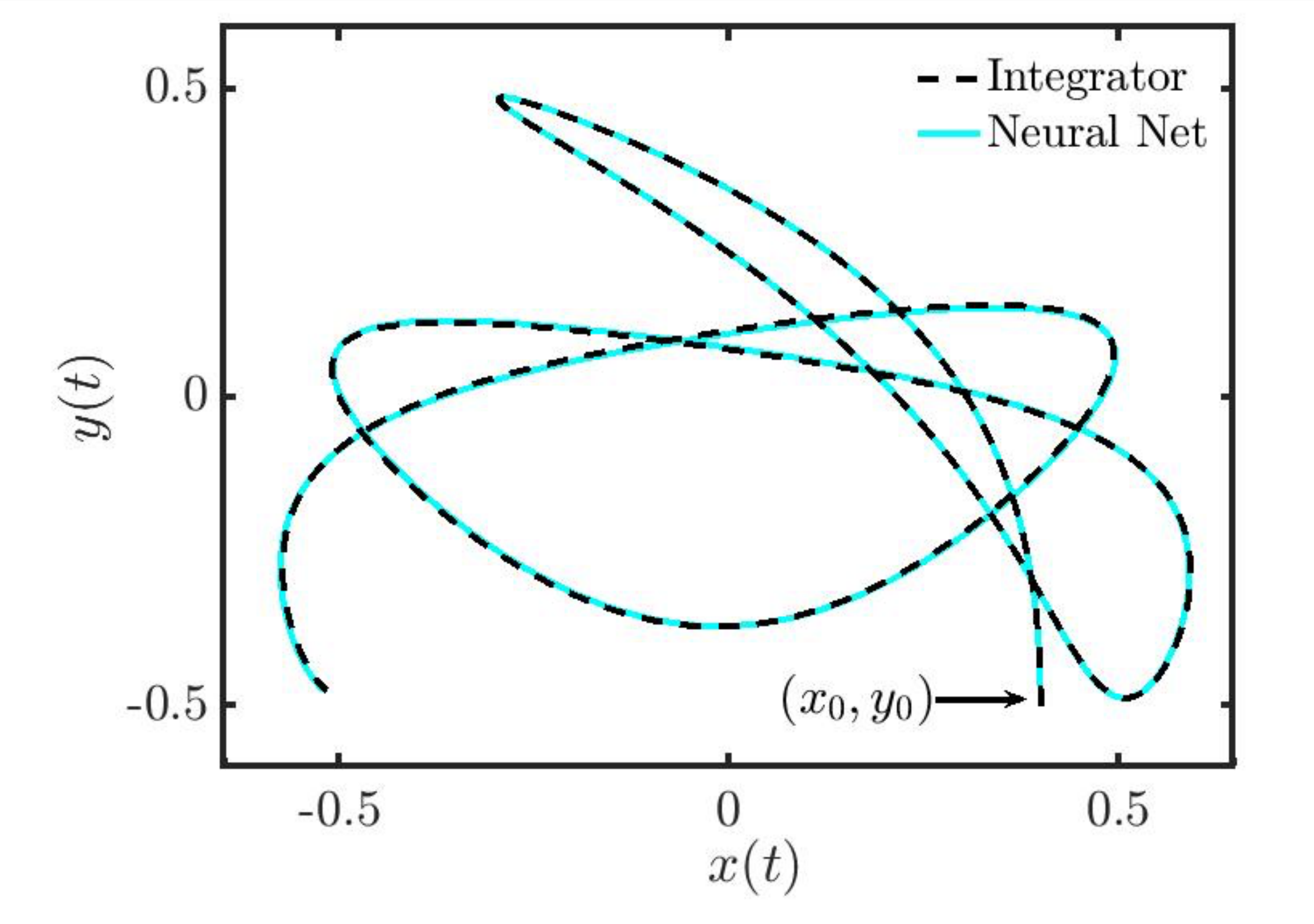
$$\mathcal{H} = \frac{1}{2}\left(p_x^2 + p_y^2\right) + \frac{1}{2}\left(x^2 + y^2\right) + \lambda\left(x^2 y - \frac{y^3}{3}\right)$$



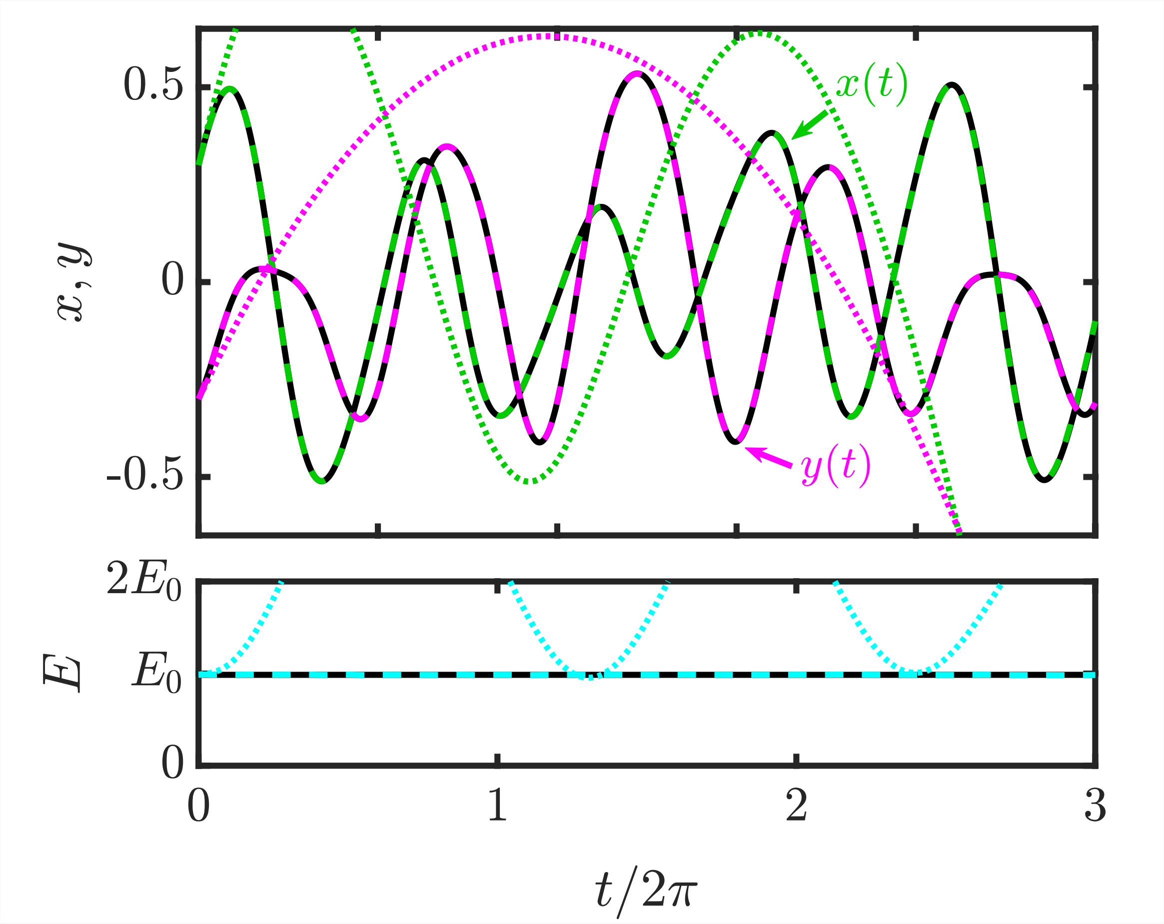
$$(\lambda = 1)$$

- 40 hidden units per layer
- Evaluate in 200 time points in the interval [0, 6π]

NN Results



Integrator
Neural Net
0.5
0
-0.5
$y(t)$
(x_0, y_0)
-0.5
0
0.5
$x(t)$



0.5
0
-0.5
x, y
$x(t)$
$y(t)$
$2E_0$
E_0
0
E
0
1
2
3
$t/2\pi$

Summary

1. Motivation: *Create training sets*
2. Stellar Formation: *Fluids*
3. Fluids: *Solving PDEs*
4. NN to Solve DE: *Supervised and Unsupervised*
5. Physical Symmetries: *Embedded in Neural Networks*

Thank you