NTNU
Norwegian University of Science and Technology

Neural Networks, Differential Equations, and Structure Preservation

Davide Murari

PhD thesis defence,
Trondheim, September 25, 2024

Supervisory Committee: Elena Celledoni, and Brynjulf Owren
Assessment Committee: Virginie Ehrlacher, Matthew Colbrook, and Jo Eidsvik

Papers in my thesis

PART 1: Structure preserving deep learning

- Dynamical Systems–Based Neural Networks

  Celledoni, E., Murari, D., Owren, B., Schönlieb, C. B., & Sherry, F., SIAM Journal of Scientific Computing
- Resilient Graph Neural Networks: A Coupled Dynamical Systems Approach

  Eliasof, M., Murari, D., Sherry, F., & Schönlieb, C. B., 27TH European Conference on Artificial Intelligence
- Predictions Based on Pixel Data: Insights from PDEs and Finite Differences

  Celledoni, E., Jackaman, J., Murari, D., & Owren, B., Submitted

Papers in my thesis

PART 2: Solving and discovering differential equations

- Lie Group integrators for mechanical systems

  Celledoni, E., Çokaj, E., Leone, A., Murari, D., & Owren, B., International Journal of Computer Mathematics

- Learning Hamiltonians of constrained mechanical systems

  Celledoni, E., Leone, A., Murari, D., & Owren, B., Journal of Computational and Applied Mathematics

- Neural networks for the approximation of Euler's elastica

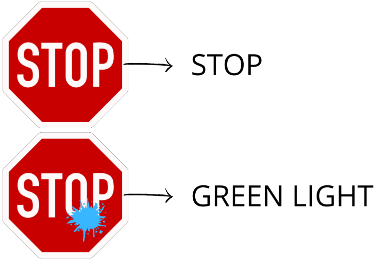
  Celledoni, E., Çokaj, E., Leone, A., Leyendecker, S., Murari, D., Owren, B., Sato Martín de Almagro, R.T. & Stavole, M., Submitted

- Parallel-in-Time Solutions with Extreme Learning Machines

  Betcke, M., Kreusser, L.M., & Murari, D., Submitted

Motivation

(a) ChatGPT: "Generate a picture of a monkey winning a marathon"

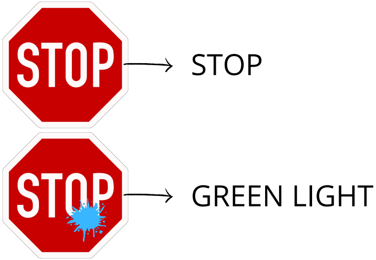


(b) Misclassification of an image that could harm self-driving cars.

- Neural networks can find accurate solutions to many problems but tend not to be interpretable or reproduce desired properties.

Motivation

(a) ChatGPT: "Generate a picture of a monkey winning a marathon"



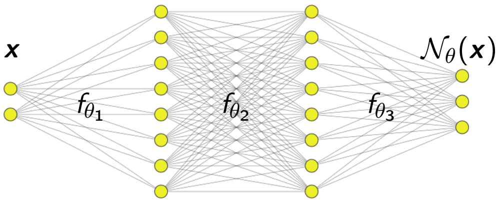
(b) Misclassification of an image that could harm self-driving cars.

- Neural networks can find accurate solutions to many problems but tend not to be interpretable or reproduce desired properties.
- We will see how to deal with some of these issues by applying the theory of dynamical systems and geometric integration.

What is a neural network?

- A neural network is a parametric map usually composed of building blocks called *layers of the network*:

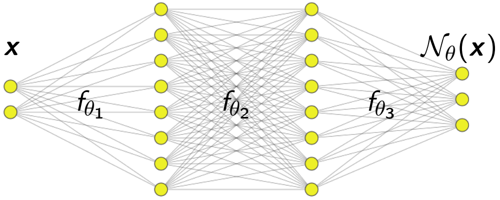
$$\mathcal{N}_\theta(\boldsymbol{x}) = f_{\theta_L} \circ \cdots \circ f_{\theta_1}(\boldsymbol{x}), \;\; \theta = \{\theta_1, \cdots, \theta_L\}\,.$$



What is a neural network?

- A neural network is a parametric map usually composed of building blocks called *layers of the network*:

$$\mathcal{N}_\theta(\boldsymbol{x}) = f_{\theta_L} \circ \cdots \circ f_{\theta_1}(\boldsymbol{x}), \;\; \theta = \{\theta_1, \cdots, \theta_L\}\,.$$

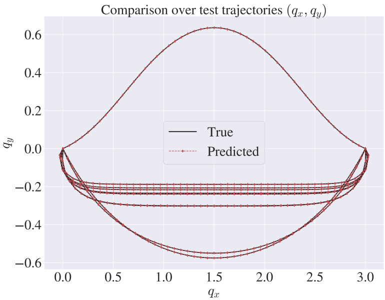


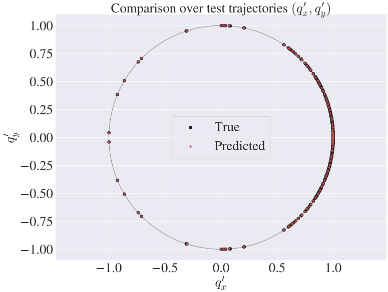
- Example: Residual Neural Networks (ResNets)

$$f_{\theta_i}(\boldsymbol{x}) = \boldsymbol{x} + B_i^\top \sigma\left(A_i\boldsymbol{x} + \boldsymbol{b}_i\right) \in \mathbb{R}^d, \;\; \boldsymbol{x} \in \mathbb{R}^d,$$
$$A_i, B_i \in \mathbb{R}^{h\times d}, \;\; \boldsymbol{b}_i \in \mathbb{R}^h, \;\; \theta_i = \{A_i, B_i, \boldsymbol{b}_i\}\,.$$

Example of the Euler's elastica

- **Goal**: Build an efficient approximate solver of the Euler's elastica
- **Dataset**: A set of boundary data $\boldsymbol{x}_i = \left(\boldsymbol{q}_i^0, \left(\boldsymbol{q}_i^0\right)', \boldsymbol{q}_i^N, \left(\boldsymbol{q}_i^N\right)'\right)$ and the respective approximate solutions $\boldsymbol{y}_i$ at some grid nodes.
- **Loss function**: $\mathcal{L}(\theta) := \frac{1}{M}\sum_{i=1}^{M} \|\mathcal{N}_\theta(\boldsymbol{x}_i) - \boldsymbol{y}_i\|_2^2 \rightarrow \min$.





Neural networks based on dynamical systems

- The layer

$$f_{\theta_i}(\boldsymbol{x}) = \boldsymbol{x} + B_i^\top \sigma\left(A_i \boldsymbol{x} + \boldsymbol{b}_i\right) = \boldsymbol{x} + \mathcal{F}_{\theta_i}(\boldsymbol{x}) \in \mathbb{R}^d$$

is an explicit Euler step of size 1 for the initial value problem

$$\begin{cases} \dot{\boldsymbol{y}}(t) = B_i^\top \sigma(A_i \boldsymbol{y}(t) + \boldsymbol{b}_i) = \mathcal{F}_{\theta_i}(\boldsymbol{y}(t)), \\ \boldsymbol{y}(0) = \boldsymbol{x} \end{cases} .$$

Neural networks based on dynamical systems

- The layer

$$f_{\theta_i}(\boldsymbol{x}) = \boldsymbol{x} + B_i^\top \sigma\left(A_i \boldsymbol{x} + \boldsymbol{b}_i\right) = \boldsymbol{x} + \mathcal{F}_{\theta_i}(\boldsymbol{x}) \in \mathbb{R}^d$$

is an explicit Euler step of size 1 for the initial value problem

$$\begin{cases} \dot{\boldsymbol{y}}(t) = B_i^\top \sigma(A_i \boldsymbol{y}(t) + \boldsymbol{b}_i) = \mathcal{F}_{\theta_i}(\boldsymbol{y}(t)), \\ \boldsymbol{y}(0) = \boldsymbol{x} \end{cases} .$$

- We can define ResNet-like neural networks by choosing a family of parametric functions $\mathcal{S}_\Theta = \left\{\mathcal{F}_\theta : \mathbb{R}^d \to \mathbb{R}^d : \ \theta \in \Theta\right\}$ and a numerical method $\Psi^h_{\mathcal{F}}$, like explicit Euler defined as $\Psi^h_{\mathcal{F}}(\boldsymbol{x}) = \boldsymbol{x} + h\mathcal{F}(\boldsymbol{x})$, and set

$$\mathcal{N}_\theta(\boldsymbol{x}) = \Psi^{h_L}_{\mathcal{F}_{\theta_L}} \circ \cdots \circ \Psi^{h_1}_{\mathcal{F}_{\theta_1}}(\boldsymbol{x}), \ \mathcal{F}_{\theta_1}, ..., \mathcal{F}_{\theta_L} \in \mathcal{S}_\Theta.$$

Imposing structure over a neural network

- To build networks satisfying a desired property, we can either restrict the parametrisation $\mathcal{N}_\theta$ or modify the loss function.

Imposing structure over a neural network

- To build networks satisfying a desired property, we can either restrict the parametrisation $\mathcal{N}_\theta$ or modify the loss function.

- **Restrict the architecture**:

$$\mathcal{N}_\theta(\boldsymbol{x}) = \frac{\widetilde{\mathcal{N}}_\theta(\boldsymbol{x})}{\left\|\widetilde{\mathcal{N}}_\theta(\boldsymbol{x})\right\|_2} \|\boldsymbol{x}\|_2 \,.$$

- **Modify the loss function**:

$$\widetilde{\mathcal{L}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{N}_\theta(\boldsymbol{x}_i) - \boldsymbol{y}_i\|_2^2 + \underbrace{\frac{1}{N}\sum_{i=1}^{N} \left(\|\boldsymbol{x}_i\|_2 - \|\mathcal{N}_\theta(\boldsymbol{x}_i)\|_2\right)^2}_{\text{regulariser}} \,.$$

Imposing structure over a neural network

- To build networks satisfying a desired property, we can either restrict the parametrisation $\mathcal{N}_\theta$ or modify the loss function.

- **Restrict the architecture**:

$$\mathcal{N}_\theta(\boldsymbol{x}) = \frac{\widetilde{\mathcal{N}}_\theta(\boldsymbol{x})}{\left\|\widetilde{\mathcal{N}}_\theta(\boldsymbol{x})\right\|_2} \|\boldsymbol{x}\|_2 \,.$$

- **Modify the loss function**:

$$\widetilde{\mathcal{L}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{N}_\theta(\boldsymbol{x}_i) - \boldsymbol{y}_i\|_2^2 + \underbrace{\frac{1}{N}\sum_{i=1}^{N} \left(\|\boldsymbol{x}_i\|_2 - \|\mathcal{N}_\theta(\boldsymbol{x}_i)\|_2\right)^2}_{\text{regulariser}} \,.$$

- Not all restrictions are equally effective, e.g. $\mathcal{N}_R(\boldsymbol{x}) = R\boldsymbol{x}$, $R^\top R = I_d$, is norm-preserving but probably not expressive enough.

Structured networks based on dynamical systems

- Choose a property $\mathcal{P}$ that the network has to satisfy, e.g. volume preservation.

Structured networks based on dynamical systems

- Choose a property $\mathcal{P}$ that the network has to satisfy, e.g. volume preservation.
- Choose a family of parametric vector fields $\mathcal{S}_\Theta$ whose solutions satisfy $\mathcal{P}$, e.g.

$$\mathcal{F}_\theta(\boldsymbol{x}) = \begin{bmatrix} \sigma\left(A_1\boldsymbol{x}_2 + \boldsymbol{b}_1\right) \\ \sigma\left(A_2\boldsymbol{x}_1 + \boldsymbol{b}_2\right) \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}.$$

Structured networks based on dynamical systems

- Choose a property $\mathcal{P}$ that the network has to satisfy, e.g. volume preservation.
- Choose a family of parametric vector fields $\mathcal{S}_\Theta$ whose solutions satisfy $\mathcal{P}$, e.g.

$$\mathcal{F}_\theta(\boldsymbol{x}) = \begin{bmatrix} \sigma\left(A_1\boldsymbol{x}_2 + \boldsymbol{b}_1\right) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma\left(A_2\boldsymbol{x}_1 + \boldsymbol{b}_2\right) \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}.$$

- Choose a numerical method $\Psi^h_{\mathcal{F}_\theta}$ that preserves the property $\mathcal{P}$ at a discrete level, e.g.

$$\Psi^h_{\mathcal{F}_\theta}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x}_1 + h\sigma\left(A_1\boldsymbol{x}_2 + \boldsymbol{b}_1\right) =: \widetilde{\boldsymbol{x}}_1 \\ \boldsymbol{x}_2 + h\sigma\left(A_2\widetilde{\boldsymbol{x}}_1 + \boldsymbol{b}_2\right) \end{bmatrix}.$$

- The resulting network $\mathcal{N}_\theta = \Psi^{h_L}_{\mathcal{F}_{\theta_L}} \circ \cdots \circ \Psi^{h_1}_{\mathcal{F}_{\theta_1}}$ will preserve $\mathcal{P}$.

Approximation properties

- The inductive bias provided by modelling the network starting from dynamical systems, allows us to study these models using the theory of numerical analysis and dynamical systems.

Approximation properties

- The inductive bias provided by modelling the network starting from dynamical systems, allows us to study these models using the theory of numerical analysis and dynamical systems.

Universal approximation theorem

Let $F : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ be a continuous function, with $\Omega \subset \mathbb{R}^d$ a compact set. Then, for every $\varepsilon > 0$, there exists a finite set of gradient vector fields $\nabla V^1, \cdots, \nabla V^L$, sphere-preserving vector fields $X_S^1, \cdots, X_S^L$, and time steps $h_1, \cdots, h_L \in \mathbb{R}$ such that

$$\left\| F - \Psi^{h_L}_{\nabla V^L} \circ \Psi^{h_L}_{X_S^L} \circ \ldots \circ \Psi^{h_1}_{\nabla V^1} \circ \Psi^{h_1}_{X_S^1} \right\|_{L^p(\Omega)} < \varepsilon.$$

Adversarial robustness for classification tasks

Description of the problem

Classification problem

Let $\Omega \subset \mathbb{R}^d$ be a set whose points are known to belong to C classes. Given part of their labels, we want to label the remaining points with a function $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^C$ where we set

$$\text{predicted class of } \boldsymbol{x} = \underset{c=1,\dots,C}{\arg\max} \left(\mathcal{N}_\theta(\boldsymbol{x})^\top \boldsymbol{e}_c \right).$$

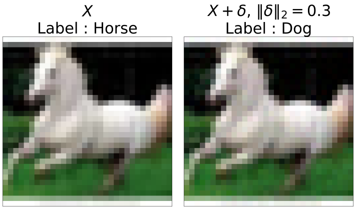
Description of the problem

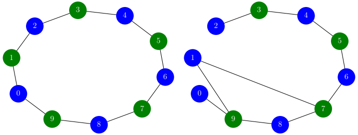
Classification problem

Let $\Omega \subset \mathbb{R}^d$ be a set whose points are known to belong to C classes. Given part of their labels, we want to label the remaining points with a function $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^C$ where we set

$$\text{predicted class of } \boldsymbol{x} = \underset{c=1,\dots,C}{\arg\max} \left(\mathcal{N}_\theta(\boldsymbol{x})^\top \boldsymbol{e}_c \right).$$

Adversarial examples





How to have guaranteed robustness

- Not all correct predictions are equivalent.
- Let $\ell(\boldsymbol{x}) = 2$ be the correct label for the point $\boldsymbol{x} \in \Omega$.
- $\mathcal{N}_{\theta_1}(\boldsymbol{x}) = \begin{bmatrix} 0.49 & 0.51 & 0 \end{bmatrix}$ is not so certain as a prediction.
- $\mathcal{N}_{\theta_2}(\boldsymbol{x}) = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix}$ there is a higher gap here.

How to have guaranteed robustness

- Not all correct predictions are equivalent.
- Let $\ell(\boldsymbol{x}) = 2$ be the correct label for the point $\boldsymbol{x} \in \Omega$.
- $\mathcal{N}_{\theta_1}(\boldsymbol{x}) = \begin{bmatrix} 0.49 & 0.51 & 0 \end{bmatrix}$ is not so certain as a prediction.
- $\mathcal{N}_{\theta_2}(\boldsymbol{x}) = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix}$ there is a higher gap here.

Margin: $\mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x}) := \mathcal{N}_\theta(\boldsymbol{x})^\top \boldsymbol{e}_{\ell(\boldsymbol{x})} - \max_{j \neq \ell(\boldsymbol{x})} \mathcal{N}_\theta(\boldsymbol{x})^\top \boldsymbol{e}_j.$

$\mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x}) > 0 \implies \mathcal{N}_\theta$ correctly classifies $\boldsymbol{x}$.

$$\mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x}) > \sqrt{2}\mathrm{Lip}(\mathcal{N}_\theta)\varepsilon \implies \mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x} + \boldsymbol{\eta}) > 0 \, \forall \|\boldsymbol{\eta}\|_2 \leq \varepsilon.$$

How to have guaranteed robustness

- Not all correct predictions are equivalent.
- Let $\ell(\boldsymbol{x}) = 2$ be the correct label for the point $\boldsymbol{x} \in \Omega$.
- $\mathcal{N}_{\theta_1}(\boldsymbol{x}) = \begin{bmatrix} 0.49 & 0.51 & 0 \end{bmatrix}$ is not so certain as a prediction.
- $\mathcal{N}_{\theta_2}(\boldsymbol{x}) = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix}$ there is a higher gap here.

Margin: $\mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x}) := \mathcal{N}_\theta(\boldsymbol{x})^\top \boldsymbol{e}_{\ell(\boldsymbol{x})} - \max_{j \neq \ell(\boldsymbol{x})} \mathcal{N}_\theta(\boldsymbol{x})^\top \boldsymbol{e}_j.$

$$\mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x}) > 0 \implies \mathcal{N}_\theta \text{ correctly classifies } \boldsymbol{x}.$$

$$\mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x}) > \sqrt{2}\mathrm{Lip}(\mathcal{N}_\theta)\varepsilon \implies \mathcal{M}_{\mathcal{N}_\theta}(\boldsymbol{x} + \boldsymbol{\eta}) > 0 \, \forall \|\boldsymbol{\eta}\|_2 \leq \varepsilon.$$

- We constrain the Lipschitz constant of $\mathcal{N}_\theta$ (and train the network so it maximises the margin).

Lipschitz-constrained networks

Contractive maps

$$\mathcal{F}_\theta^c(\boldsymbol{x}) = -A_c^\top \sigma\left(A_c \boldsymbol{x} + \boldsymbol{b}_c\right), \; A_c^\top A_c = I,$$

$$\Psi_{\mathcal{F}_\theta^c}^{h_c}(\boldsymbol{x}) = \boldsymbol{x} - h_c A_c^\top \sigma\left(A_c \boldsymbol{x} + \boldsymbol{b}_c\right)$$

$$\left\|\Psi_{\mathcal{F}_\theta^c}^{h_c}(\boldsymbol{y}) - \Psi_{\mathcal{F}_\theta^c}^{h_c}(\boldsymbol{x})\right\|_2 \leq \sqrt{1 - h_c + h_c^2}\,\|\boldsymbol{y} - \boldsymbol{x}\|_2$$

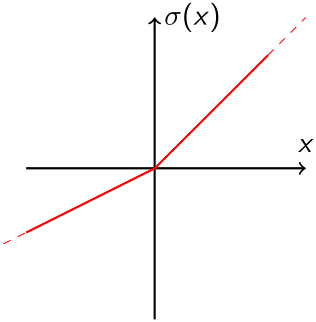


Figure: $\sigma(x) = \max\left\{x, \frac{x}{2}\right\}$

Lipschitz-constrained networks

Contractive maps

$$\mathcal{F}_\theta^c(\boldsymbol{x}) = -A_c^\top \sigma\left(A_c\boldsymbol{x} + \boldsymbol{b}_c\right), \; A_c^\top A_c = I,$$

$$\Psi_{\mathcal{F}_\theta^c}^{h_c}(\boldsymbol{x}) = \boldsymbol{x} - h_c A_c^\top \sigma\left(A_c\boldsymbol{x} + \boldsymbol{b}_c\right)$$

$$\left\|\Psi_{\mathcal{F}_\theta^c}^{h_c}(\boldsymbol{y}) - \Psi_{\mathcal{F}_\theta^c}^{h_c}(\boldsymbol{x})\right\|_2 \leq \sqrt{1 - h_c + h_c^2}\,\|\boldsymbol{y} - \boldsymbol{x}\|_2$$

Expansive maps

$$\mathcal{F}_\theta^e(\boldsymbol{x}) = A_e^\top \sigma\left(A_e\boldsymbol{x} + \boldsymbol{b}_e\right), \; A_e^\top A_e = I,$$

$$\Psi_{\mathcal{F}_\theta^e}^{h_e}(\boldsymbol{x}) = \boldsymbol{x} + h_e A_e^\top \sigma\left(A_e\boldsymbol{x} + \boldsymbol{b}_e\right)$$

$$\left\|\Psi_{\mathcal{F}_\theta^e}^{h_e}(\boldsymbol{y}) - \Psi_{\mathcal{F}_\theta^e}^{h_e}(\boldsymbol{x})\right\|_2 \leq (1 + h_e)\,\|\boldsymbol{y} - \boldsymbol{x}\|_2$$

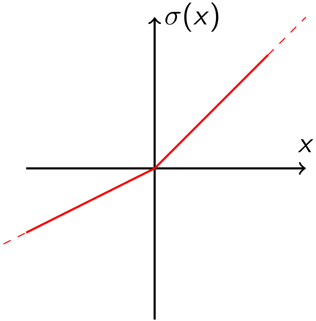


Figure: $\sigma(x) = \max\left\{x, \frac{x}{2}\right\}$

Lipschitz-constrained networks

- To get a $1-$Lipschitz neural network we alternate the one-step methods and restrict the step sizes suitably:

$$\mathcal{N}_\theta = \Psi^{h_{2L}}_{\mathcal{F}^c_{\theta_{2L}}} \circ \Psi^{h_{2L-1}}_{\mathcal{F}^e_{\theta_{2L-1}}} \circ \cdots \circ \Psi^{h_2}_{\mathcal{F}^c_{\theta_2}} \circ \Psi^{h_1}_{\mathcal{F}^e_{\theta_1}}$$

$$\sqrt{1 - h_{2k} + h_{2k}^2}\,(1 + h_{2k-1}) \leq 1, \;\; k = 1, \cdots, L.$$

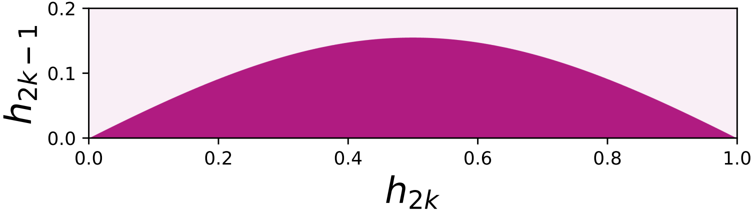
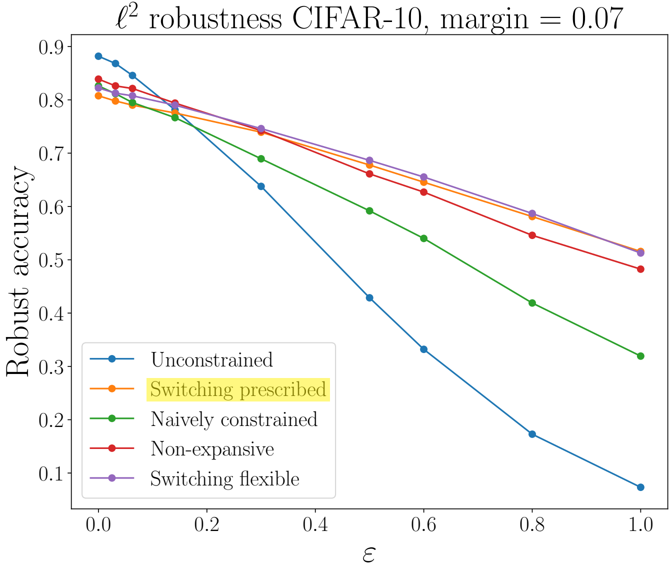


Figure: Admissible time steps to get a $1-$Lipschitz neural network

Numerical experiment with CIFAR-10



Learning tasks involving dynamical systems

Definition of the problem

- **Data**: $\left\{\left(\boldsymbol{x}_i^0, \boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^M\right)\right\}_{i=1,\ldots,N}$, $\boldsymbol{x}_i^j = \phi_{\mathcal{F}}^{jh}\left(\boldsymbol{x}_i^0\right) + \boldsymbol{\delta}_i^j$, $j = 0, \cdots, M$, for an unknown $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$.

Definition of the problem

- **Data**: $\left\{\left(\boldsymbol{x}_i^0, \boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^M\right)\right\}_{i=1,\dots,N}$, $\boldsymbol{x}_i^j = \phi_{\mathcal{F}}^{jh}\left(\boldsymbol{x}_i^0\right) + \boldsymbol{\delta}_i^j$, $j = 0, \cdots, M$, for an unknown $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$.
- **Goal 1**: Approximate the vector field $\mathcal{F}$

Definition of the problem

- **Data**: $\left\{\left(\boldsymbol{x}_i^0, \boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^M\right)\right\}_{i=1,\dots,N}$, $\boldsymbol{x}_i^j = \phi_{\mathcal{F}}^{jh}\left(\boldsymbol{x}_i^0\right) + \boldsymbol{\delta}_i^j$, $j = 0, \cdots, M$, for an unknown $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$.
- **Goal 1**: Approximate the vector field $\mathcal{F}$
- **Goal 2**: Approximate the map $\boldsymbol{x}_i^j \mapsto \boldsymbol{x}_i^{j+1}$, i.e., one step with the exact flow map $\phi_{\mathcal{F}}^h$.

Definition of the problem

- **Data**: $\left\{\left(\boldsymbol{x}_i^0, \boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^M\right)\right\}_{i=1,\dots,N}$, $\boldsymbol{x}_i^j = \phi_{\mathcal{F}}^{jh}\left(\boldsymbol{x}_i^0\right) + \boldsymbol{\delta}_i^j$, $j = 0, \cdots, M$, for an unknown $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$.
- **Goal 1**: Approximate the vector field $\mathcal{F}$
- **Goal 2**: Approximate the map $\boldsymbol{x}_i^j \mapsto \boldsymbol{x}_i^{j+1}$, i.e., one step with the exact flow map $\phi_{\mathcal{F}}^h$.
- **Generic solution strategy**: Introduce a parametric model $\mathcal{F}_\theta : \mathbb{R}^d \to \mathbb{R}^d$, choose a one-step method $\Psi_{\mathcal{F}_\theta}^h : \mathbb{R}^d \to \mathbb{R}^d$, and solve

$$\mathcal{L}(\theta) = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M \left\| \left(\Psi_{\mathcal{F}_\theta}^h\right)^j \left(\boldsymbol{x}_i^0\right) - \boldsymbol{x}_i^j \right\|_2^2 \to \min .$$

Definition of the problem

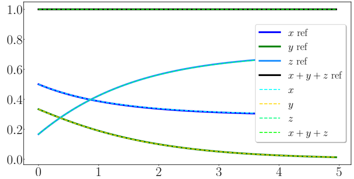
- **Data**: $\left\{\left(\boldsymbol{x}_i^0, \boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^M\right)\right\}_{i=1,\dots,N}$, $\boldsymbol{x}_i^j = \phi_{\mathcal{F}}^{jh}\left(\boldsymbol{x}_i^0\right) + \boldsymbol{\delta}_i^j$, $j = 0, \cdots, M$, for an unknown $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$.
- **Goal 1**: Approximate the vector field $\mathcal{F}$
- **Goal 2**: Approximate the map $\boldsymbol{x}_i^j \mapsto \boldsymbol{x}_i^{j+1}$, i.e., one step with the exact flow map $\phi_{\mathcal{F}}^h$.
- **Generic solution strategy**: Introduce a parametric model $\mathcal{F}_\theta : \mathbb{R}^d \to \mathbb{R}^d$, choose a one-step method $\Psi_{\mathcal{F}_\theta}^h : \mathbb{R}^d \to \mathbb{R}^d$, and solve

$$\mathcal{L}(\theta) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left\| \left(\Psi_{\mathcal{F}_\theta}^h\right)^j \left(\boldsymbol{x}_i^0\right) - \boldsymbol{x}_i^j \right\|_2^2 \to \min .$$

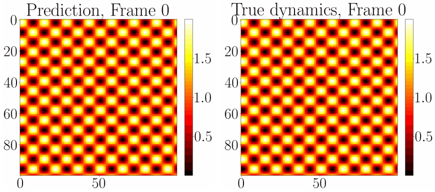
- If we know more about $\mathcal{F}$ or the geometric properties of the flow $\phi_{\mathcal{F}}^h$ we might want to constrain this procedure.

NTNU

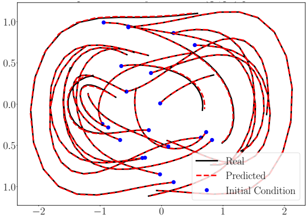
Problems we have considered



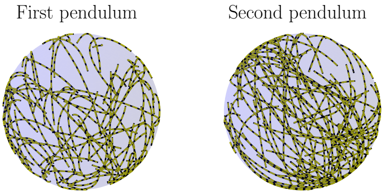
(a) Learning the mass preserving flow map of the SIR model.



(b) Learning the norm-preserving flow map of the linear advection PDE.



(c) Learning the Hamiltonian of unconstrained systems.



(d) Learning the Hamiltonian of constrained systems.

Constrained Hamiltonian systems

- Holonomically constrained Hamiltonian systems can be described by the differential algebraic equation

$$\begin{cases} \dot{\boldsymbol{y}}(t) = \mathbb{J}\nabla H(\boldsymbol{y}(t)), \quad \boldsymbol{y} = (\boldsymbol{q}, \boldsymbol{p}) \\ g(\boldsymbol{q}) = 0, \quad g : \mathbb{R}^d \to \mathbb{R}^c \end{cases}, \quad \mathbb{J} = \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix}.$$

- Its configuration manifold and associated tangent space are

$$\mathcal{Q} = \left\{ \boldsymbol{q} \in \mathbb{R}^d : \; g(\boldsymbol{q}) = 0 \right\} \subset \mathbb{R}^d, \; \dim(\mathcal{Q}) = d - c,$$
$$T_{\boldsymbol{q}}\mathcal{Q} = \left\{ \boldsymbol{v} \in \mathbb{R}^d : \; G(\boldsymbol{q})\boldsymbol{v} = 0 \right\}.$$

Parametrisation of $\mathcal{F}_\theta$

- The constrained dynamics can be reformulated in the more geometric way[1]

$$\begin{cases} \dot{\boldsymbol{q}} = P(\boldsymbol{q})\partial_{\boldsymbol{p}} H(\boldsymbol{q}, \boldsymbol{p}) \\ \dot{\boldsymbol{p}} = -P(\boldsymbol{q})^\top \partial_{\boldsymbol{q}} H(\boldsymbol{q}, \boldsymbol{p}) + W(\boldsymbol{q}, \boldsymbol{p})\partial_{\boldsymbol{p}} H(\boldsymbol{q}, \boldsymbol{p}), \end{cases}$$

where $P(\boldsymbol{q}) : \mathbb{R}^d \to T_{\boldsymbol{q}}\mathcal{Q}$.

[1] T. Lee, M. Leok, and N H. McClamroch. *Global formulations of Lagrangian and Hamiltonian Dynamics on Manifolds*. Vol. 13. Springer, 2017.

Parametrisation of $\mathcal{F}_\theta$

- The constrained dynamics can be reformulated in the more geometric way[1]

$$\begin{cases} \dot{\boldsymbol{q}} = P(\boldsymbol{q})\partial_{\boldsymbol{p}} H(\boldsymbol{q},\boldsymbol{p}) \\ \dot{\boldsymbol{p}} = -P(\boldsymbol{q})^\top \partial_{\boldsymbol{q}} H(\boldsymbol{q},\boldsymbol{p}) + W(\boldsymbol{q},\boldsymbol{p})\partial_{\boldsymbol{p}} H(\boldsymbol{q},\boldsymbol{p}), \end{cases}$$

  where $P(\boldsymbol{q}) : \mathbb{R}^d \to T_{\boldsymbol{q}}\mathcal{Q}$.

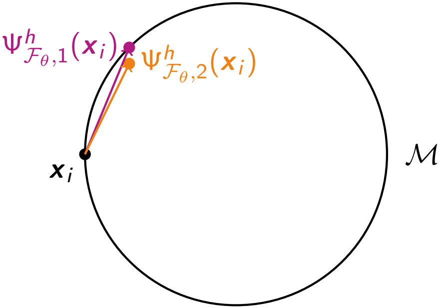
- We thus set

$$\mathcal{F}_\theta(\boldsymbol{q},\boldsymbol{p}) = \begin{bmatrix} P(\boldsymbol{q})\partial_{\boldsymbol{p}} H_\theta(\boldsymbol{q},\boldsymbol{p}) \\ -P(\boldsymbol{q})^\top \partial_{\boldsymbol{q}} H_\theta(\boldsymbol{q},\boldsymbol{p}) + W(\boldsymbol{q},\boldsymbol{p})\partial_{\boldsymbol{p}} H_\theta(\boldsymbol{q},\boldsymbol{p}) \end{bmatrix},$$

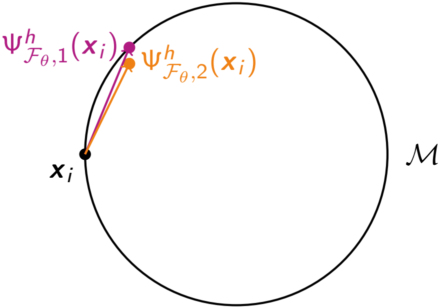
$$H_\theta(\boldsymbol{q},\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^\top M_{\theta_1}^{-1}(\boldsymbol{q})\boldsymbol{p} + \mathcal{N}_{\theta_2}(\boldsymbol{q}),\ \theta = (\theta_1,\theta_2)$$

[1] Lee, Leok, and McClamroch, *Global formulations of Lagrangian and Hamiltonian Dynamics on Manifolds*.

Choice of $\Psi^h_{\mathcal{F}_\theta}$

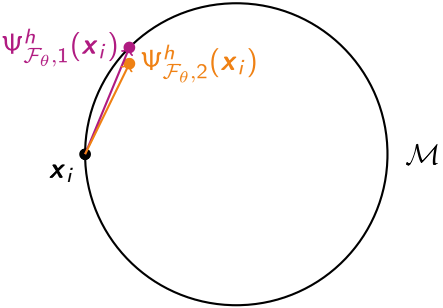


Choice of $\Psi^h_{\mathcal{F}_\theta}$



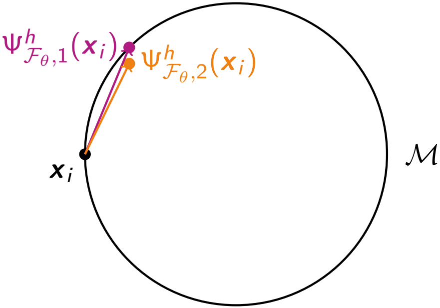
- We assume $\mathcal{M}$ is a homogeneous manifold.

Choice of $\Psi^h_{\mathcal{F}_\theta}$



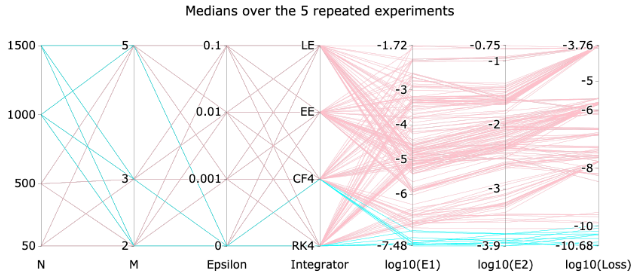
- We assume $\mathcal{M}$ is a homogeneous manifold.
- We consider the transitive Lie group action $\varphi : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$, i.e., for every $m_1, m_2 \in \mathcal{M}$ there is $g \in \mathcal{G}$ with $\varphi(g, m_1) = m_2$.

Choice of $\Psi^h_{\mathcal{F}_\theta}$



- We assume $\mathcal{M}$ is a homogeneous manifold.
- We consider the transitive Lie group action $\varphi : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$, i.e., for every $m_1, m_2 \in \mathcal{M}$ there is $g \in \mathcal{G}$ with $\varphi(g, m_1) = m_2$.
- For $\Psi_{\mathcal{F}_\theta,1}$ we choose a Lie group method, i.e., a method of the form $\Psi^h_{\mathcal{F}_\theta,1}(\mathbf{x}) = \varphi(g(\mathcal{F}_\theta, h, \mathbf{x}), \mathbf{x})$, $g(\mathcal{F}_\theta, h, \mathbf{x}) \in \mathcal{G}$.

Experimental results



$$\mathcal{E}_1 = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left\| \left(\Psi^h_{\mathcal{F}}\right)^j\left(\boldsymbol{x}_i^0\right) - \left(\Psi^h_{\mathcal{F}_\theta}\right)^j\left(\boldsymbol{x}_i^0\right)\right\|_2^2$$

$$\mathcal{E}_2 = \frac{1}{N}\sum_{i=1}^{N}\left| H\left(\boldsymbol{x}_i\right) - H_\theta\left(\boldsymbol{x}_i\right) - \frac{1}{N}\sum_{l=1}^{N}\left(H\left(\boldsymbol{x}_l\right) - H_\theta\left(\boldsymbol{x}_l\right)\right)\right|$$

NTNU

THANK YOU FOR THE ATTENTION