UNIVERSITÀ
DI TRENTO

modely

A Framework for Model-Structured Neural Networks

November 2024

Gastone Pietro Rosati Papini & Mattia Piccinini

Department of Industrial Engineering

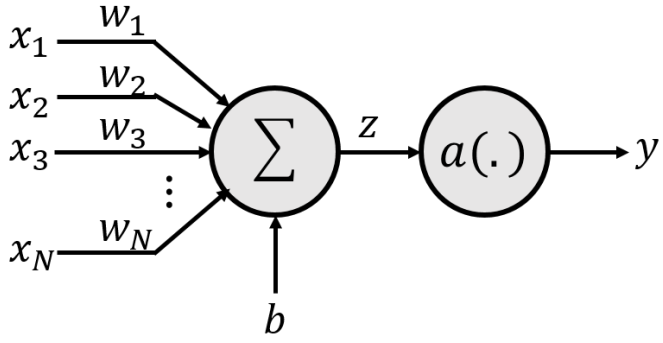
Overview I

1. Introduction & Motivation
2. Related Work
3. NNodely
4. Use Cases
5. Future Work

Introduction & Motivation

The Perceptron

- A **neural network** (NN) is a mathematical model inspired by the structure of the human brain
- The basic component of a NN is the ***Perceptron***[1]

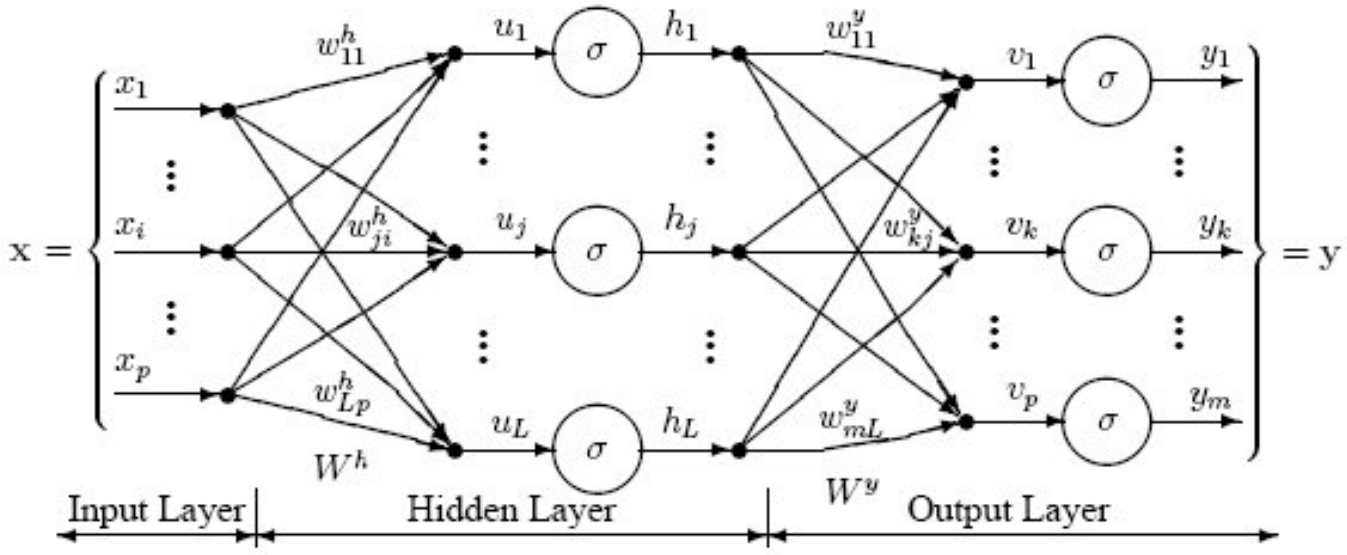


- From a mathematical point of view:

$$z = \bar{w}^T \bar{x} + b, \quad y = a(z)$$

[1] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain.". In: *Psychological review* (1958).

From Single- to Multi-layer Perceptron



- Multiple perceptrons can be concatenated to model general nonlinear functions (see Tensorflow Playground)
- **Universal approximation theorem**: a NN with a single hidden layer can approximate any continuous function[2]

[2] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Math. Control Signal Systems* 2 (1989), pp. 303–314.

The NN Internal Structure Matters

- *Computer vision*: **convolutional neural networks**, embedding the concept of spatial invariant[3]

[3] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (1997).

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* (2020).

[6] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

The NN Internal Structure Matters

- *Computer vision*: **convolutional neural networks**, embedding the concept of spatial invariant[3]
- *Speech recognition*: **long short-term memory networks**, embedding the concept of short and long term memory[4]

[3] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (1997).

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* (2020).

[6] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

The NN Internal Structure Matters

- *Computer vision*: **convolutional neural networks**, embedding the concept of spatial invariant[3]
- *Speech recognition*: **long short-term memory networks**, embedding the concept of short and long term memory[4]
- *Data generation*: **generative adversarial networks**, used for generative AI[5] (example: OpenAI Dall-E)

[3] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (1997).

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* (2020).

[6] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

The NN Internal Structure Matters

- *Computer vision*: **convolutional neural networks**, embedding the concept of spatial invariant[3]
- *Speech recognition*: **long short-term memory networks**, embedding the concept of short and long term memory[4]
- *Data generation*: **generative adversarial networks**, used for generative AI[5] (example: OpenAI Dall-E)
- *Natural language processing*: **transformers** embed the concept of self-attention[6] (example: chatGPT)

---

[3] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (1997).

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* (2020).

[6] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

Model-Structured Neural Networks (MS-NNs)

Model-based approach

- Leverage physics knowledge
- Equation-based formulation

Data-driven approach

Model-Structured Neural Networks (MS-NNs)

Model-based approach

- Leverage physics knowledge
- Equation-based formulation
- Hard to model everything
- System identif. may require ad-hoc experiments

Data-driven approach

Model-Structured Neural Networks (MS-NNs)

Model-based approach

- Leverage physics knowledge
- Equation-based formulation
- Hard to model everything
- System identif. may require ad-hoc experiments

Data-driven approach

- No need for expert knowledge
- Model complex functions

Model-Structured Neural Networks (MS-NNs)

Model-based approach

- Leverage physics knowledge
- Equation-based formulation
- Hard to model everything
- System identif. may require ad-hoc experiments

Data-driven approach

- No need for expert knowledge
- Model complex functions
- Black-box structure
- Data-hungry
- Poor generalization

Model-Structured Neural Networks (MS-NNs)

Model-based approach

- Leverage physics knowledge
- Equation-based formulation
- Hard to model everything
- System identif. may require ad-hoc experiments

Data-driven approach

- No need for expert knowledge
- Model complex functions
- Black-box structure
- Data-hungry
- Poor generalization

Model-Structured Neural Networks (MS-NN)

- **Flexibility** of data-driven (NN) techniques

Model-Structured Neural Networks (MS-NNs)

Model-based approach

- Leverage physics knowledge
- Equation-based formulation
- Hard to model everything
- System identif. may require ad-hoc experiments

Data-driven approach

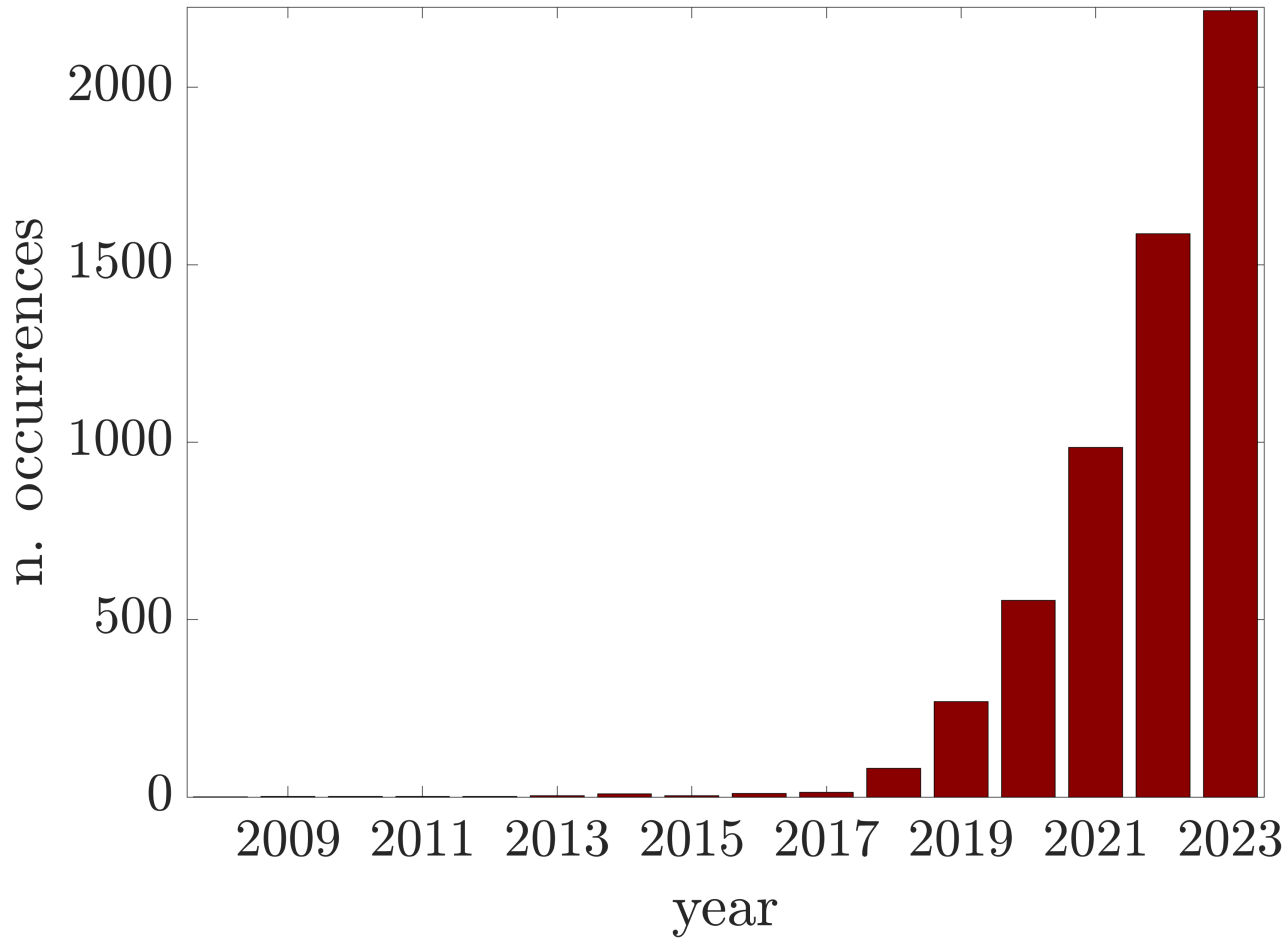
- No need for expert knowledge
- Model complex functions
- Black-box structure
- Data-hungry
- Poor generalization

Model-Structured Neural Networks (MS-NN)

- **Flexibility** of data-driven (NN) techniques
- Embed **domain-specific knowledge** into the NN's internal **structure** $\longrightarrow$ increased **explainability** + **generalization**

Related Work

Emerging Interest in Explainable AI



Keywords "*interpretable machine learning*", "*explainable machine learning*" and "*explainable neural networks*" in Scopus

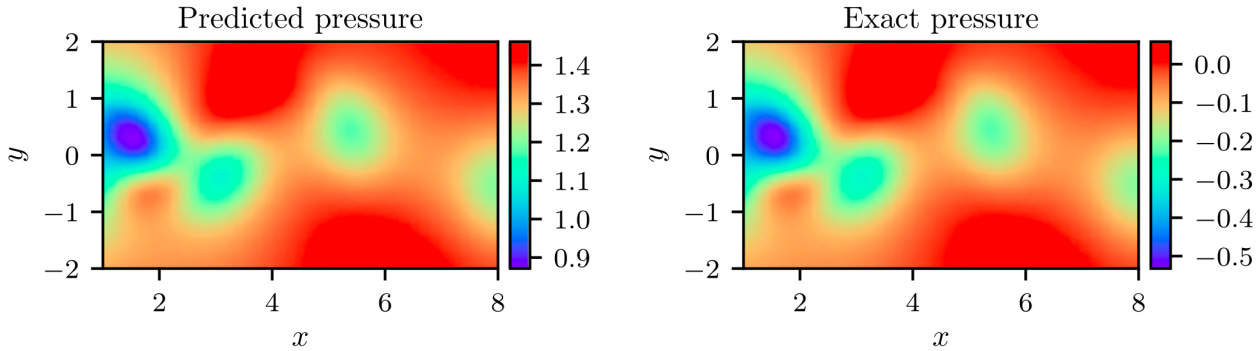
Overview of Physics-Driven NN Approaches

How to embed physics- or model-based knowledge in NNs?

- **Decide on problem**, e.g. defining input and output
- **Curate the data**, operation on data before fit
- **Design the internal architecture**:
  - NN to mimic/generalize the structure of a physical model
  - Equation learning, learn to combine a set of operators (e.g. trig functions) to approximate an unknown dynamics
- **Design a loss function** for the NN training, e.g. to conserve energy or satisfy a PDE
- **Design of optimization algorithms**

Physics-informed NNs (PINNs), Raissi et at. (2019)

- **Idea**: Data-driven solution or discovery of complex PDEs
- **How**: *Structured loss function* based on the PDE residuals, but using deep *unstructured* NNs
- **Pros**: Learn the solution of PDEs using small training datasets; accelerate simulations of complex systems
- **Examples**: Fluid dynamics (Navier Stokes), quantum mechanics, solid mechanics, reaction-diffusion[7]. **Note**: no modeling of real physical systems, only PDEs



[7] M. Raissi, P. Perdikaris, and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* (2019).

Approaches related to PINNs

- **Hamiltonian**[8] and **Lagrangian NNs**[9] that learn to conserve energy in mechanical systems and robotic tasks
- **Pontryagin AI**[10] to learn the policies of optimal control tracking problems (OCPs):
  - NN to approximate the controls of an OCP
  - NN trained with a tracking loss function
  - No need to solve the adjoint equations of the OCP
- NNs to approximate the value function of Hamilton-Jacobi-Bellman equations for **dynamic programming**[11]

---

[8] Sam Greydanus, Misko Dzamba, and Jason Yosinski. *Hamiltonian Neural Networks*. 2019. arXiv: 1906.01563.

[9] Michael Lutter, Kim Listmann, and Jan Peters. "Deep Lagrangian Networks for end-to-end learning of energy-based control for under-actuated systems". In: Nov. 2019, pp. 7718–7725.

[10] Lucas Böttcher, Nino Antulov-Fantulin, and Thomas Asikis. "AI Pontryagin or how artificial neural networks learn to control dynamical systems". In: *Nature Communications* 13 (Jan. 2022).

[11] Murad Abu-Khalaf and Frank L. Lewis. "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach". In: *Automatica* (2005).

Combining NNs and analytical models

- **Idea**: Use NNs to improve the prediction of analytical models
- **How**: Augment an analytical model with NNs to better learn some complex dynamics
- **Pros**: No need to learn the entire model from scratch; partly preserve the physical meaning of the model
- **Examples**:
  - NN to learn the parameters of multibody vehicle models[12]
  - Neural tire models in multibody vehicle models[13][14]
  - Neural ODE to predict the heart rate during workouts[15]

[12] John Chrosniak, Jingyun Ning, and Madhur Behl. "Deep Dynamics: Vehicle Dynamics Modeling With a Physics-Constrained Neural Network for Autonomous Racing". In: *IEEE Robotics and Automation Letters* 9.6 (2024), pp. 5292–5297.

[13] Jan Węgrzynowski et al. *Learning dynamics models for velocity estimation in autonomous racing*. 2024. arXiv: 2408.15610 [cs.RO].

[14] Taekyung Kim, Hojin Lee, and Wonsuk Lee. "Physics Embedded Neural Network Vehicle Model and Applications in Risk-Aware Autonomous Driving Using Latent Features". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 4182–4189.

[15] Achille Nazaret et al. "Modeling personalized heart rate response to exercise and environmental factors with wearables data". In: *npj Digital Medicine* 6 (Nov. 2023).

Our Model-Structured NNs

- **Idea**: Embed domain-specific prior knowledge into the NN's internal structure
- **How**: Neuralize the structure of a physics-based model
- **Pros**: Model interpretability; improved generalization
- **Examples**: Modeling *physical systems* (unknown dyna./param.)
  - Model and control the long.-lat. vehicle dynamics[16] [17]
  - Vehicle state estimation[18]
  - Model mechanical systems[19];
  - Human motion prediction[20]

[16] Mauro Da Lio, Daniele Bortoluzzi, and Gastone Pietro Rosati Papini. "Modelling longitudinal vehicle dynamics with neural networks". In: *Vehicle System Dynamics* (2020).

[17] Mattia Piccinini et al. "A Physics-Driven Artificial Agent for Online Time-Optimal Vehicle Motion Planning and Control". In: *IEEE Access* 11 (2023), pp. 46344–46372.

[18] Mauro Da Lio, Mattia Piccinini, and Francesco Biral. "Robust and Sample-Efficient Estimation of Vehicle Lateral Velocity Using Neural Networks With Explainable Structure Informed by Kinematic Principles". In: *IEEE Transactions on Intelligent Transportation Systems* 24.12 (2023), pp. 13670–13684.

[19] Antonio Di Dino, Francesco Biral, and Paolo Bosetti. "Hybrid Modeling of Non-Linear Mechanical Systems: The Case of a Vehicle Shock Absorber". In: vol. 4. Jan. 2011.

[20] Alessandro Antonucci et al. "Efficient Prediction of Human Motion for Real-Time Robotics Applications With Physics-Inspired Neural Networks". In: *IEEE Access* 10 (2022), pp. 144–157.

NNodely Framework

Motivation

- *Modeling*, *control* and *estimation* of **physical systems**, whose internal dynamics may be partially unknown

Motivation

- *Modeling*, *control* and *estimation* of **physical systems**, whose internal dynamics may be partially unknown
- **Speed-up** the development of MS-NN, which is complex to design using standard deep-learning framework

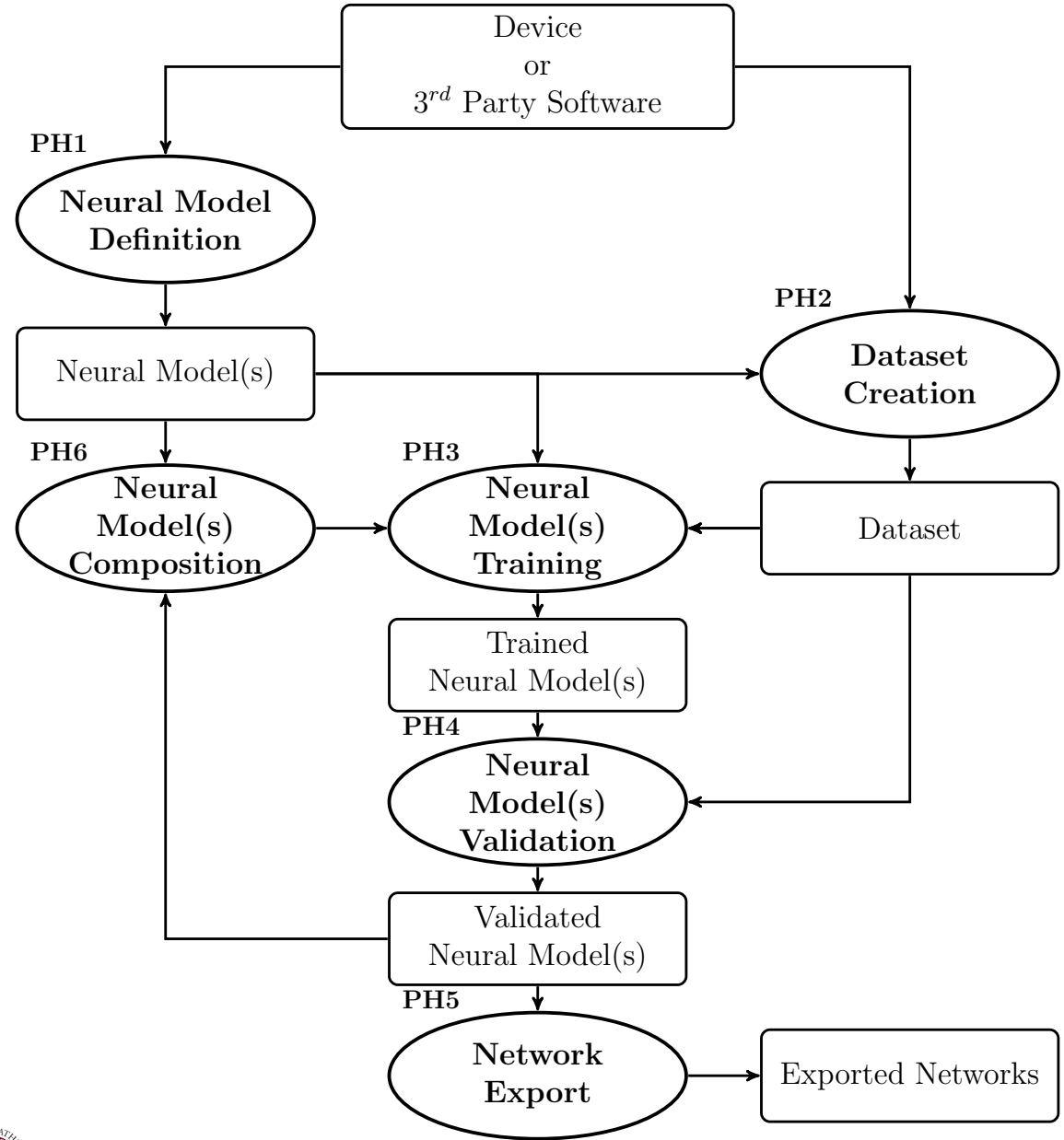
Motivation

- *Modeling*, *control* and *estimation* of **physical systems**, whose internal dynamics may be partially unknown
- **Speed-up** the development of MS-NN, which is complex to design using standard deep-learning framework
- **Support professionals** with classical modeling backgrounds, such as **physicists** and **engineers**, in using data-driven approaches (but embedding knowledge inside) to address their challenges.

Motivation

- *Modeling*, *control* and *estimation* of **physical systems**, whose internal dynamics may be partially unknown
- **Speed-up** the development of MS-NN, which is complex to design using standard deep-learning framework
- **Support professionals** with classical modeling backgrounds, such as **physicists** and **engineers**, in using data-driven approaches (but embedding knowledge inside) to address their challenges.
- A **repository** of incremental **know-how** that effectively collects approaches with the same purpose, i.e. building an increasingly advanced library of MS-NNs for various applications.

Framework workflow



nmodely
neuralize your model

Model-Structured Neural Networks with nnodely

MS-NNs Definition (PH1)

- Defining **Inputs**
- Defining **Outputs**
- Defining **Relations**:
  - Aritmetic operations
  - Trigonometric operations
  - Custom functions
  - Local models
  - Fuzzification
  - ...

MS-NN Manipulation

- Data loading and dataset creation (PH2)
- Definition of the fitting objectives (PH3)
- Definition of the network composition and connection (PH6)
- Training (PH3)
- Validation using model based criteria (PH4)
- MS-NN export (PH5) in PyTorch and ONNX format

Embedding Physical Principles (Main Ideas)

The **MS-NN** embeds and generalizes **physical principles** and prior-model knowledge, where the parameters are learned from **experimental data**

Embedding Physical Principles (Main Ideas)

Main nnodely concepts:

- Linear **superposition of forces**:

$$\dot{x} = f(x, u) = f_x(x) + f_1(x, u_1) + f_2(x, u_2) + \ldots + f_n(x, u_n)$$

Embedding Physical Principles (Main Ideas)

Main nnodely concepts:

- Linear **superposition of forces**:

$$\dot{x} = f(x,u) = f_x(x) + f_1(x,u_1) + f_2(x,u_2) + \ldots + f_n(x,u_n)$$

- The system dynamics are treated using **FIR** or **IIR** filter/layer

Embedding Physical Principles (Main Ideas)

Main nnodely concepts:

- Linear **superposition of forces**:

$$\dot{x} = f(x,u) = f_x(x) + f_1(x,u_1) + f_2(x,u_2) + \ldots + f_n(x,u_n)$$

- The system dynamics are treated using **FIR** or **IIR** filter/layer
- The non-linearities can be addressed using **local models** or **custom functions**

Embedding Physical Principles (Main Ideas)

Main nnodely concepts:

- Linear **superposition of forces**:

$$\dot{x} = f(x,u) = f_x(x) + f_1(x,u_1) + f_2(x,u_2) + \ldots + f_n(x,u_n)$$

- The system dynamics are treated using **FIR** or **IIR** filter/layer
- The non-linearities can be addressed using **local models** or **custom functions**
- Capturing the residuals of analytical models (such as friction, delays, and backlash) through **ad-hoc neuralization**

Embedding Physical Principles (Main Ideas)

Main nnodely concepts:

- Linear **superposition of forces**:

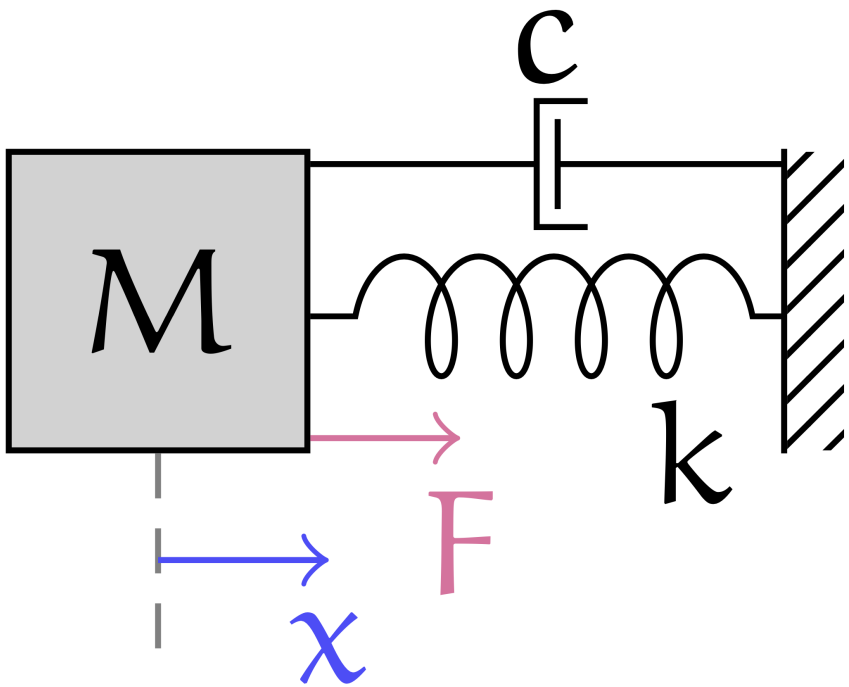
$$\dot{x} = f(x,u) = f_x(x) + f_1(x,u_1) + f_2(x,u_2) + \ldots + f_n(x,u_n)$$

- The system dynamics are treated using **FIR** or **IIR** filter/layer
- The non-linearities can be addressed using **local models** or **custom functions**
- Capturing the residuals of analytical models (such as friction, delays, and backlash) through **ad-hoc neuralization**
- Embedding easily the concept of **time** inside the NN

Use Cases

Mass-spring-damper

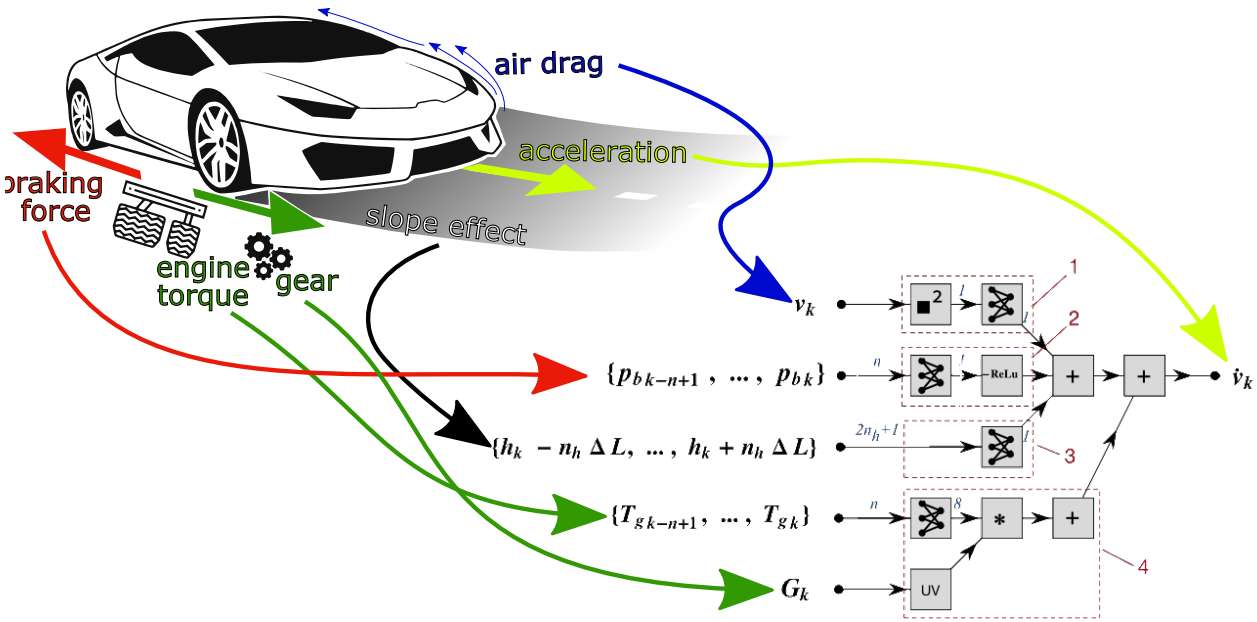
- Learn the solution of a mass-spring-damper system using NNodely
- The code is available **here**[21]



[21] https://github.com/tonegas/nnodely-applications

Modeling the forward longitudinal vehicle dynamics[22]

$$m\,\dot{v}_x = F_{x_f} + F_{x_r} - k_d v_x^2 - m\left(g c_r \cos(\theta) + g \sin(\theta)\right)$$



$$\dot{v}_x = \sum_i^{\text{gear}} g_i \text{FIR}_i(\bar{T}_g) - \text{Relu}(\text{FIR}(\bar{p_b})) + \text{FIR}(v_x^2) - \text{Linear}(\bar{h_k})$$

$(F_{x_f}+F_{x_r})/m$ $\quad$ $k_d v_x^2/m$ $\quad$ $g c_r \cos(\theta)+g \sin(\theta)$

[22] Mauro Da Lio, Daniele Bortoluzzi, and Gastone Pietro Rosati Papini. "Modelling longitudinal vehicle dynamics with neural networks". In: *Vehicle System Dynamics* (2020).

Estimating the mass of a vehicle – 1/3

- Start from the **analytical** longitudinal vehicle dynamics

$$\left(a_x + g c_r \cos(\theta) + g \sin(\theta)\right) m = F_{x_f} + F_{x_r} - c_v v_x - k_d v_x^2$$

Estimating the mass of a vehicle – 1/3

- Start from the **analytical** longitudinal vehicle dynamics
- Effect of the lateral tire forces

$$\left(a_x + g c_r \cos(\theta) + g \sin(\theta)\right) m = F_{x_f} + F_{x_r} - c_v v_x - k_d v_x^2 - \underbrace{F_{y_f} \sin(\delta)}_{\text{lateral forces}}$$

Estimating the mass of a vehicle – 1/3

- Start from the **analytical** longitudinal vehicle dynamics
- Effect of the lateral tire forces
  - Pacejka tire models require expensive experiments to be fitted $\longrightarrow F_{y_f} = F_{y_f}(\alpha_f, \kappa_f, F_{z_f}, \gamma_f)$

$$\left(a_x + g c_r \cos(\theta) + g \sin(\theta)\right) m = F_{x_f} + F_{x_r} - c_v v_x - k_d v_x^2 - \underbrace{F_{y_f} \sin(\delta)}_{\text{lateral forces}}$$

Estimating the mass of a vehicle – 1/3

- Start from the **analytical** longitudinal vehicle dynamics
- Effect of the lateral tire forces
  - Pacejka tire models require expensive experiments to be fitted $\longrightarrow F_{y_f} = F_{y_f}(\alpha_f, \kappa_f, F_{z_f}, \gamma_f)$
  - Let's use a physics-based learnable polynomial model $F_{y_f} = f_{y_{ij}}(a_y, a_x) \longrightarrow$ this is a local model around $a_y = a_{y_{0i}}$ and $a_x = a_{x_{0j}}$

$$\left(a_x + g c_r \cos(\theta) + g \sin(\theta)\right) m = F_{x_f} + F_{x_r} - c_v v_x - k_d v_x^2 - \underbrace{F_{y_f} \sin(\delta)}_{\text{lateral forces}}$$

Estimating the mass of a vehicle – 1/3

- Start from the **analytical** longitudinal vehicle dynamics
- Effect of the lateral tire forces
  - Pacejka tire models require expensive experiments to be fitted $\longrightarrow F_{y_f} = F_{y_f}(\alpha_f, \kappa_f, F_{z_f}, \gamma_f)$
  - Let's use a physics-based learnable polynomial model $F_{y_f} = f_{y_{ij}}(a_y, a_x) \longrightarrow$ this is a local model around $a_y = a_{y_{0i}}$ and $a_x = a_{x_{0j}}$
  - We use N_y and N_x local models to cover the entire range of a_y and $a_x \longrightarrow$ **fuzzification**

$$\left(a_x + g c_r \cos(\theta) + g \sin(\theta)\right) m = F_{x_f} + F_{x_r} - c_v v_x - k_d v_x^2 - \underbrace{F_{y_f} \sin(\delta)}_{\text{lateral forces}}$$

Estimating the mass of a vehicle – 1/3

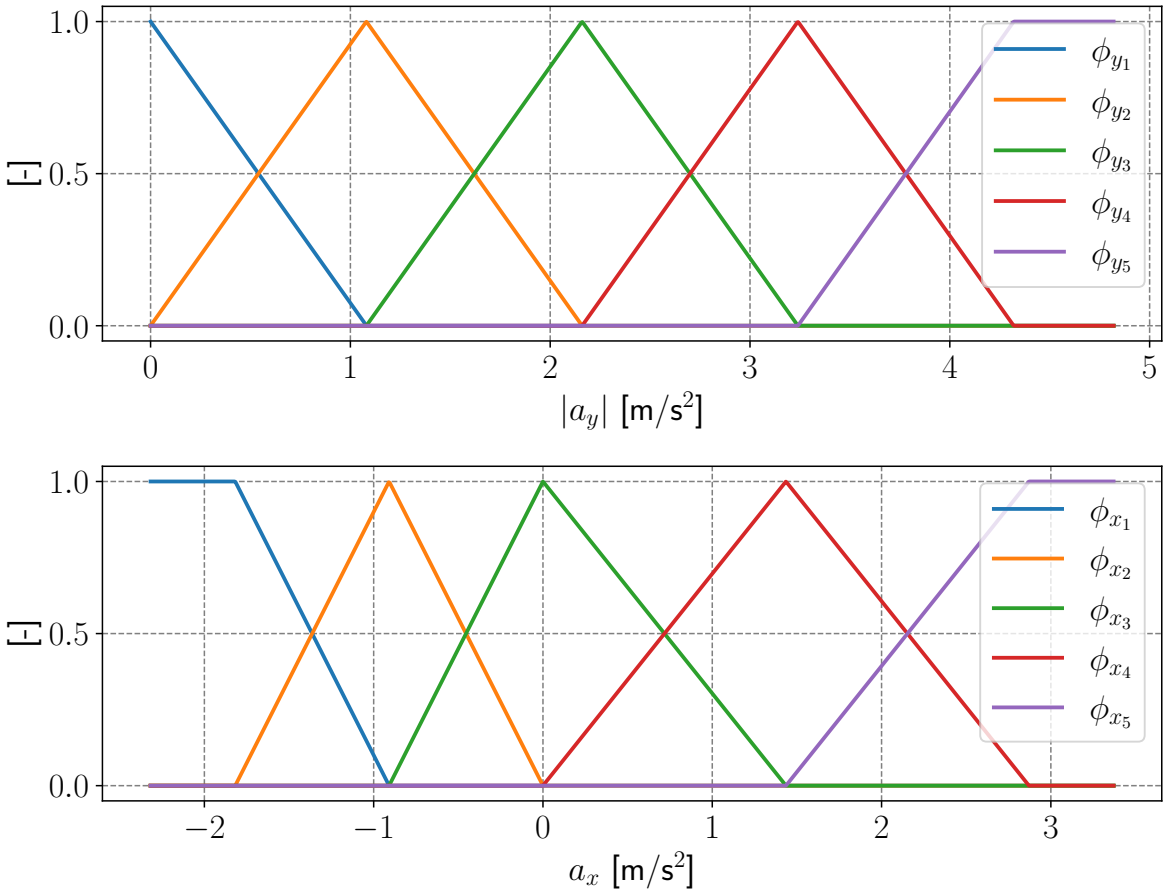
- Start from the **analytical** longitudinal vehicle dynamics
- Effect of the lateral tire forces
  - Pacejka tire models require expensive experiments to be fitted $\longrightarrow F_{y_f} = F_{y_f}(\alpha_f, \kappa_f, F_{z_f}, \gamma_f)$
  - Let's use a physics-based learnable polynomial model $F_{y_f} = f_{y_{ij}}(a_y, a_x) \longrightarrow$ this is a local model around $a_y = a_{y_{0i}}$ and $a_x = a_{x_{0j}}$
  - We use N_y and N_x local models to cover the entire range of a_y and $a_x \longrightarrow$ **fuzzification**

$$\left(a_x + g c_r \cos(\theta) + g \sin(\theta)\right) m = F_{x_f} + F_{x_r} - c_v v_x - k_d v_x^2 - \underbrace{F_{y_f} \sin(\delta)}_{\text{lateral forces}}$$

$$F_{y_f} = \sum_{i=1}^{N_y} \sum_{j=1}^{N_x} f_{y_{ij}}(a_y, a_x)\ \Phi_{ij}(|a_y|, a_x)$$

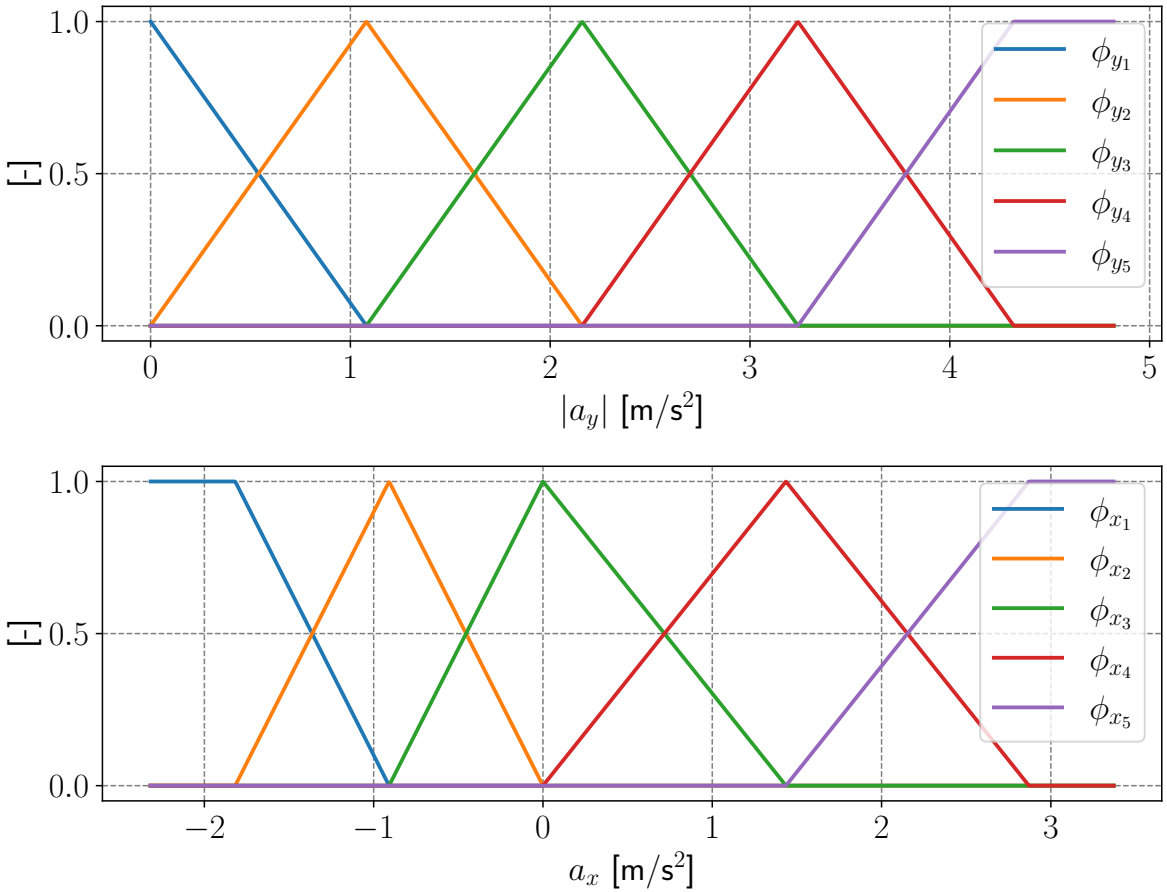
Estimating the mass of a vehicle – 2/3

- Fuzzy activation functions $\Phi_{ij}(|a_y|, a_x) = \phi_{y_i}(|a_y|)\ \phi_{x_j}(a_x)$



Estimating the mass of a vehicle – 2/3

- Fuzzy activation functions $\Phi_{ij}(|a_y|, a_x) = \phi_{y_i}(|a_y|)\ \phi_{x_j}(a_x)$
- A similar approach is applied to neuralize the rolling resistance coefficient c_r



Estimating the mass of a vehicle – 3/3

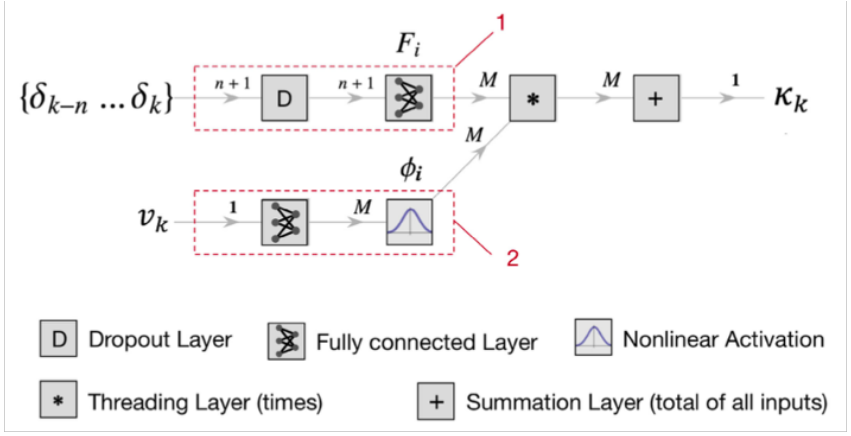
- The vehicle mass m is finally estimated by integrating the recursive least squares (RLS) into the MS-NN

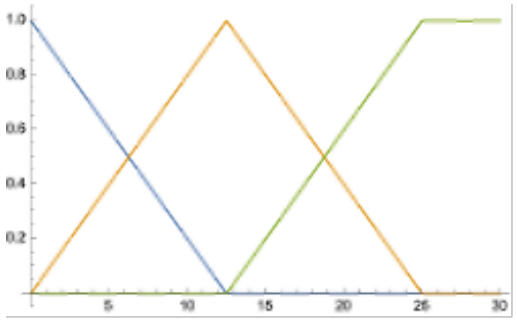
$$\begin{cases} L_k = \dfrac{P_{k-1}\,\phi_k}{\lambda + \phi_k^T\,P_{k-1}\,\phi_k} & \text{(1a)} \\ P_k = \dfrac{1}{\lambda}\left(I - L_k\,\phi_k\right)P_{k-1} & \text{(1b)} \\ \widehat{m}_k = \widehat{m}_{k-1} + L_k\,\left(y_k - \phi_k \widehat{m}_{k-1}\right) & \text{(1c)} \end{cases}$$

- L_k is the (Kalman) gain matrix
- P_k is the covariance matrix of the estimation error
- $\widehat{m}_k$ is the estimated vehicle mass
- λ is the forgetting factor

Modeling the lateral vehicle dynamics

Forward model of a vehicle's lateral dynamics[23]



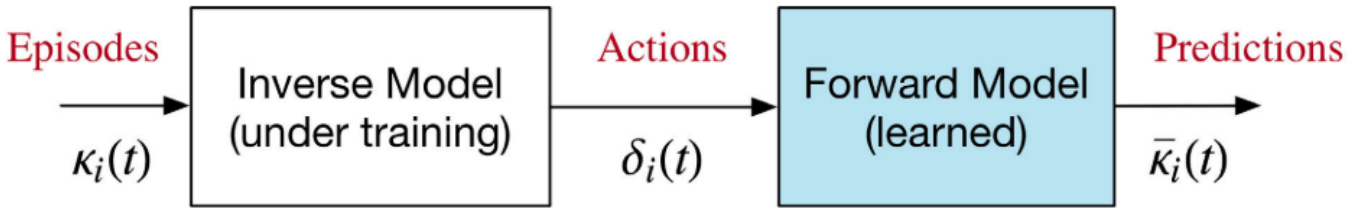


with $\phi_i(v) = \frac{1}{1+Av^2}\Lambda\left(\frac{v-v_i}{\Delta v}\right)$ where v_i are the centres of the receptive fields, Δv the receptive fields radius and $\Lambda(\cdot)$ is the Heaviside Lambda function: $\Lambda(x) = \max(1 - |x|, 0)$. The scaling factor $1 + Av^2$, with learnable A (understeering gradient)

[23] Mauro Da Lio et al. "A mental simulation approach for learning neural-network predictive control (in self-driving cars)". In: *IEEE Access* (2020).

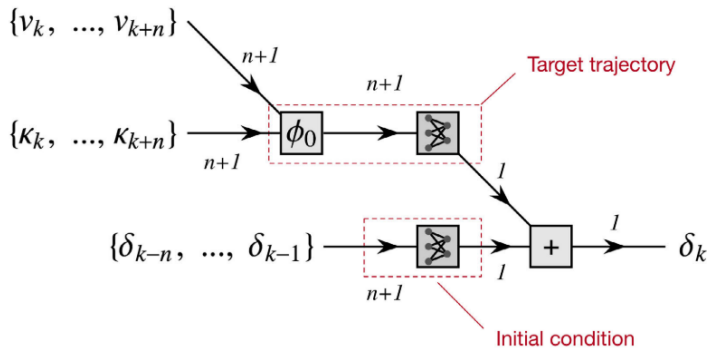
Controlling the lateral vehicle dynamics

1. Train the **forward model** of the vehicle's lateral dynamics
2. Use the trained forward model to train the **inverse model**, which is a control system ⟶ unsupervised learning



Controlling the lateral vehicle dynamics

Inverse model of a vehicle's lateral dynamics[24]



with ϕ_0 threading v, and κ into $(1 + Av^2)\kappa$, which models the understeering gradient effect re-using the understeering coeffcient A from the forward dynamics

[24] Mauro Da Lio et al. "A mental simulation approach for learning neural-network predictive control (in self-driving cars)". In: *IEEE Access* (2020).

Future Work

Future Work

- **Bridge the gaps** between machine learning, control theory, mechanics and more
- Apply NNodely to neuralize models and controllers from **different domains**
- Possibly impose energy conservation during training (PINNs-like) or **structured loss** functions
- Starting from a NNodely model, **automatically** generate and train a NNodely **controller**

Apply the framework on **your** task

Gastone Pietro Rosati Papini & Mattia Piccinini

gastone.rosatipapini@unitn.it
mattia.piccinini@tum.de