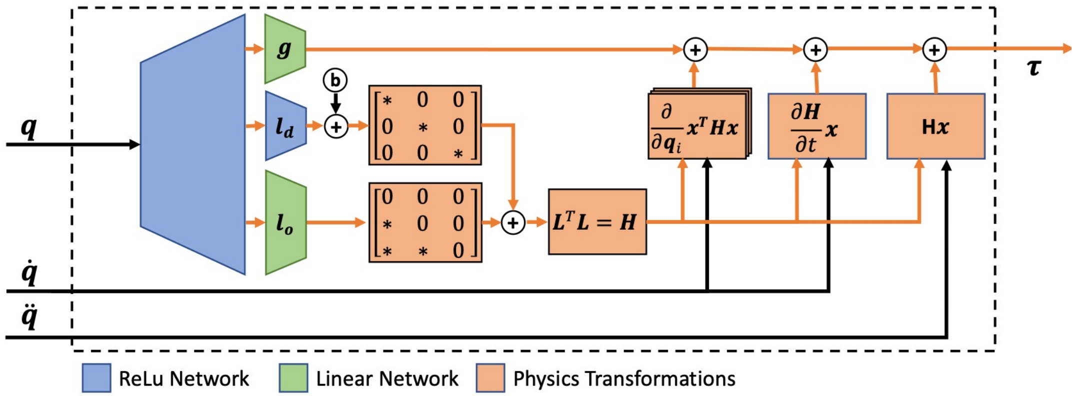
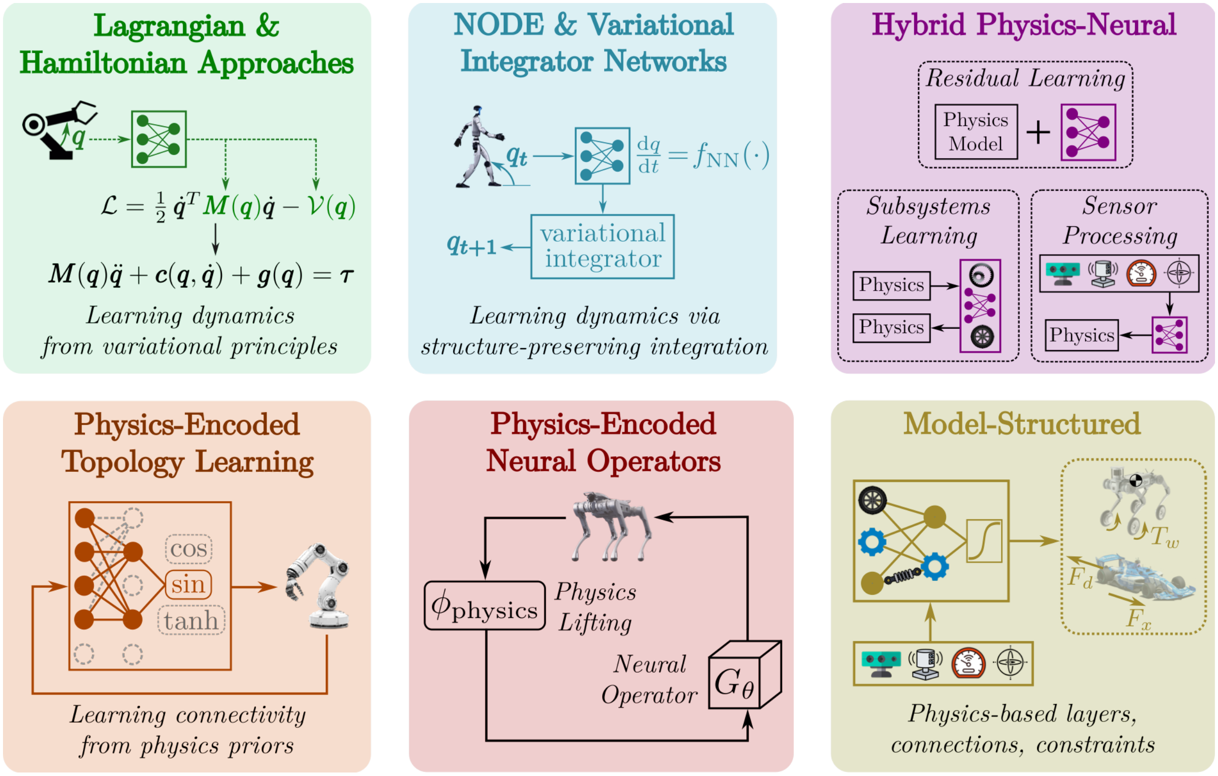
Physics-Encoded Architectures Part I: From Lagrangian to Variational Approaches

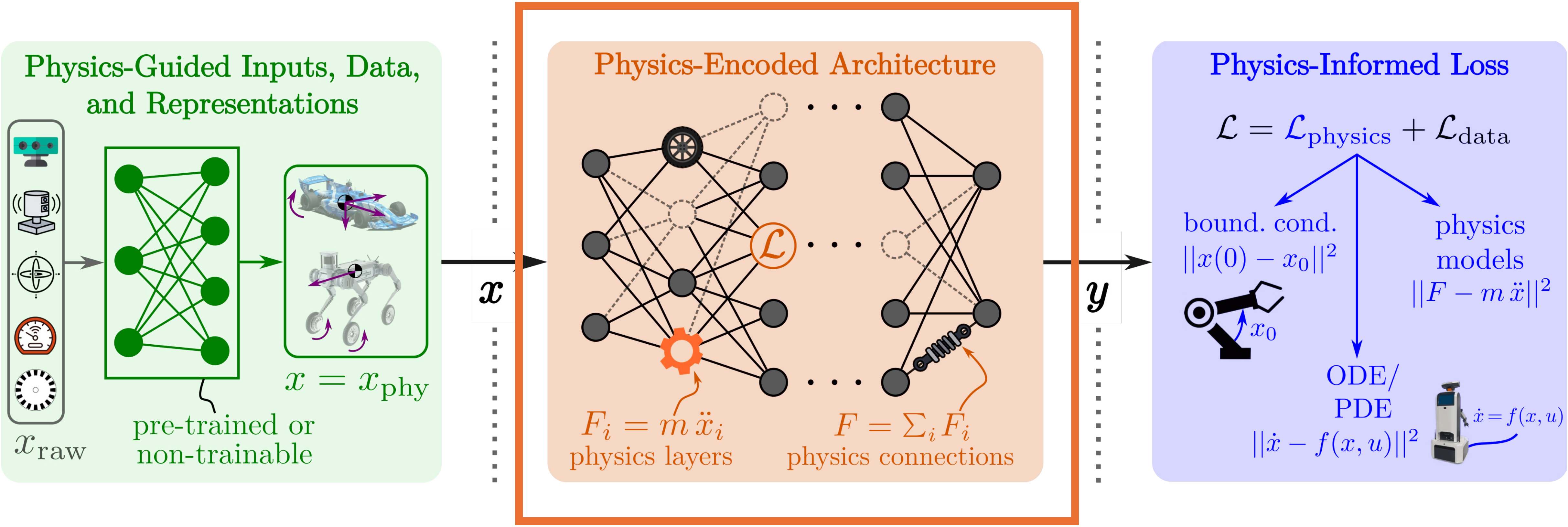
Dr. Mattia Piccinini &
Prof. Gastone Pietro Rosati Papini

University of Trento &
Technical University of Munich,
Autonomous Vehicle Systems Lab





Physics-Encoded Architectures: Positioning



Taxonomy

Physics-Encoded Neural Networks *forcibly encode known physics into their core architecture* (Faroughi et al., 2024)

ASME SETTING THE STANDARD

ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

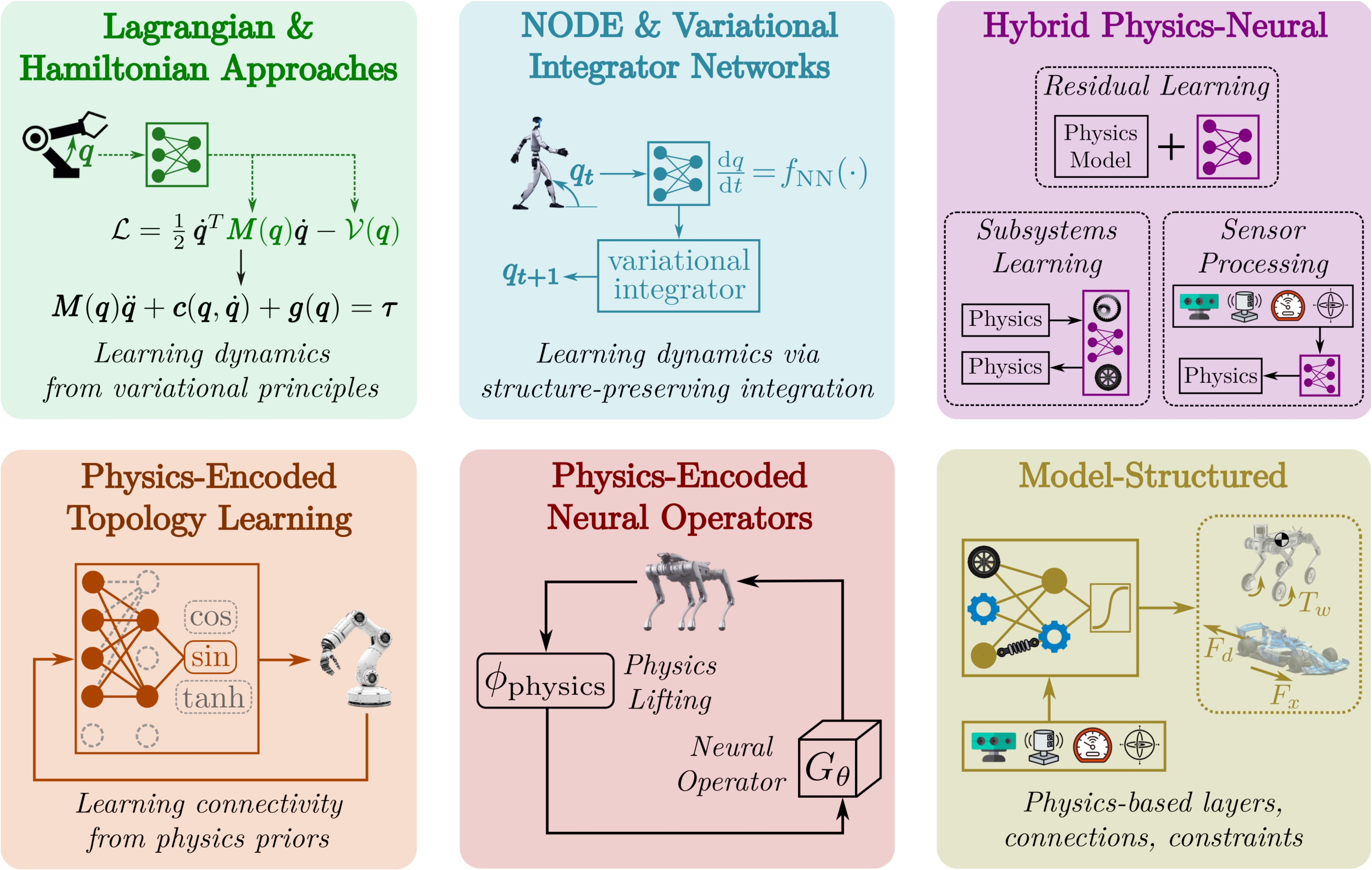
Salah A. Faroughi[1]
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666
e-mail: salah.faroughi@txstate.edu

Nikhil M. Pawar
Geo-Intelligence Laboratory,
Ingram School of Engineering,
Texas State University,
San Marcos, TX 78666

Célio Fernandes
Geo-Intelligence Laboratory,

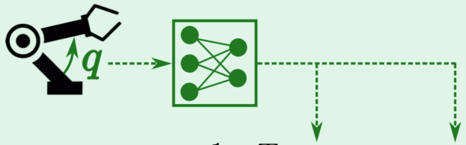
Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks and Operators in Scientific Computing: Fluid and Solid Mechanics

Physics-Encoded Learning Architectures: Multiple Classes



Physics-Encoded Learning Architectures: Multiple Classes

Lagrangian & Hamiltonian Approaches

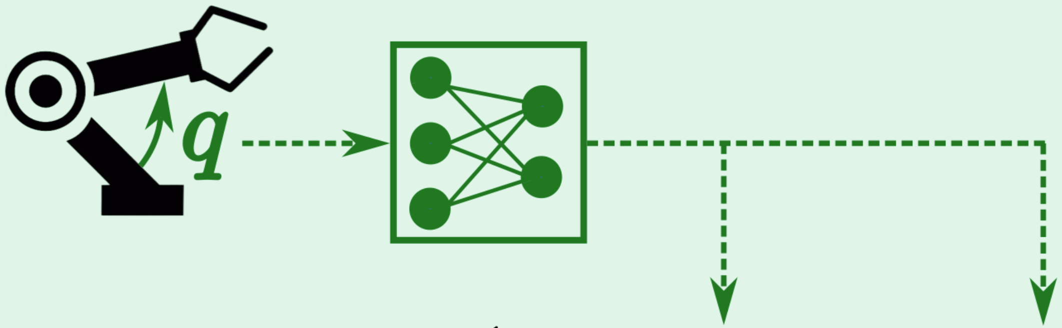


$$\mathcal{L} = \frac{1}{2}\,\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \mathcal{V}(\boldsymbol{q})$$

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

Learning dynamics from variational principles

Lagrangian & Hamiltonian Approaches



$$\mathcal{L} = \frac{1}{2}\,\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \mathcal{V}(\boldsymbol{q})$$

$$\big\downarrow$$

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

Learning dynamics
from variational principles

Formulating Equations of Motion

How can we describe the equations of motion for a physical system?

- **Newton** formulation
  - $\sum_i F_i = m\, a, \quad \sum_i M_i = I\, \alpha$ for each body
  - Explicit modeling of **forces** and **moments**

- **Lagrange** formulation
  - **Energy**-based approach
  - No need to directly model forces and moments
  - More suited for **multibody systems** with complex kinematics

⇒ Let‘s focus on the **Lagrange** formulation now

Lagrangian Equations of Motion

For a physical (mechanical) system, the **Lagrangian $\mathcal{L}$** is *typically* defined as:

$$\mathcal{L} = T - V$$

T is the kinetic energy, V is the potential energy

$$T = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad \frac{dV}{d\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q})$$

- $\boldsymbol{q}, \dot{\boldsymbol{q}}$ are generalized coordinates and velocities
- $\boldsymbol{M}(\boldsymbol{q})$ = inertia matrix (symmetric and positive definite); $\boldsymbol{g}(\boldsymbol{q})$ = grav. forces

Euler-Lagrange equations (from Calculus of Variations)

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \boldsymbol{\tau}$$

$\boldsymbol{\tau}$ = generalized (non-conservative) forces

C. C. de Wit, B. Siciliano, and G. Bastin, “Theory of robot control”, *Springer Science& Business Media*, 2012.

Lagrangian Equations of Motion

Substituting $\mathcal{L} = T - V$, $T = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$, $\frac{dV}{d\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q})$ in the Euler-Lagrange Equations:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \boldsymbol{\tau} \implies \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \underbrace{\dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{q}}\left(\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}\right)\right)^T}_{\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

$$\Downarrow$$

$$\boxed{\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}}$$

C. C. de Wit, B. Siciliano, and G. Bastin, “Theory of robot control”, *Springer Science& Business Media*, 2012.

Lagrangian Equations of Motion

Substituting $\mathcal{L} = T - V$, $T = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$, $\frac{dV}{d\boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q})$ in the Euler-Lagrange Equations:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \boldsymbol{\tau} \quad \Longrightarrow \quad \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

- $\boldsymbol{M}(\boldsymbol{q})$: inertia matrix (symmetric and positive definite)
- $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$: centripetal and Coriolis forces
- $\boldsymbol{g}(\boldsymbol{q})$: generalized gravitational forces
- $\boldsymbol{\tau}$: generalized (non-conservative) forces

*What if some of these quantities are **not known**?*

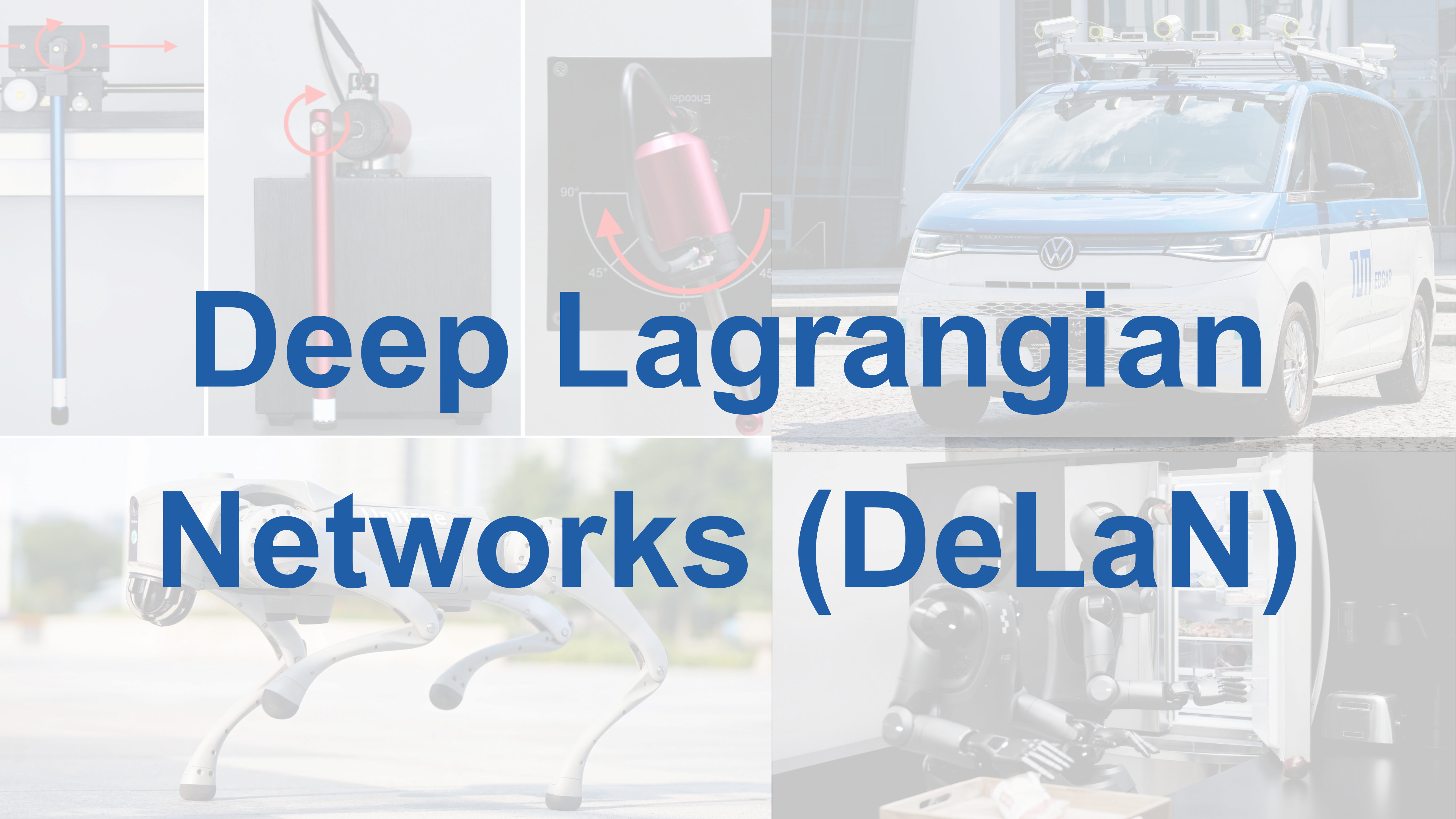
C. C. de Wit, B. Siciliano, and G. Bastin, “Theory of robot control”, *Springer Science& Business Media*, 2012.

Traditional Engineering and Deep Learning

- **Traditional engineering**: compute $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{g}(\boldsymbol{q})$ from measured / estimated masses, inertias, lengths
  - **Pros**: entirely analytic, interpretable, limited complexity
  - **Cons**: model approximations and limiting assumptions remain; real measurements / estimators are needed

- **Deep learning**: ignore the Lagrangian structure and learn everything from **data**
  - **Pros**: flexible, no reliance on approximated models
  - **Cons**: data-hungry, prone to overfit

*Can we integrate **prior physics knowledge** into learning?*

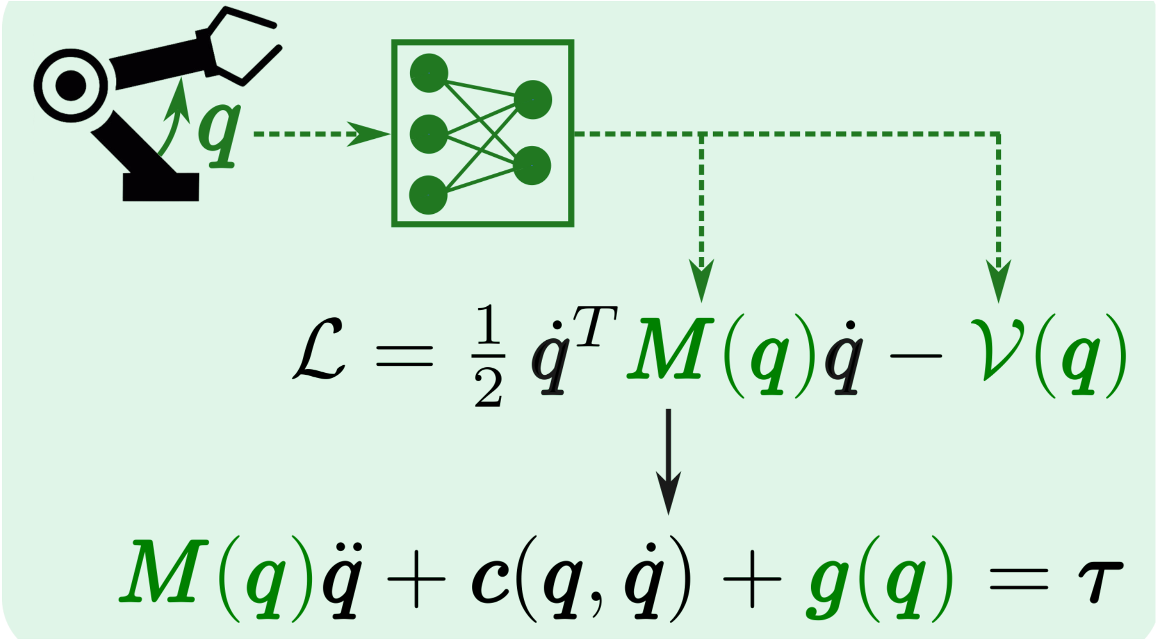


Deep Lagrangian Networks (DeLaN)

Deep Lagrangian Networks (DeLaN)

Idea: use **neural networks** (multi-layer perceptrons) to learn the unknown:

- $\boldsymbol{M}(\boldsymbol{q})$ (mass and inertia matrix)
- $V(\boldsymbol{q})$ (potential energy function), or equivalently $\boldsymbol{g}(\boldsymbol{q})$



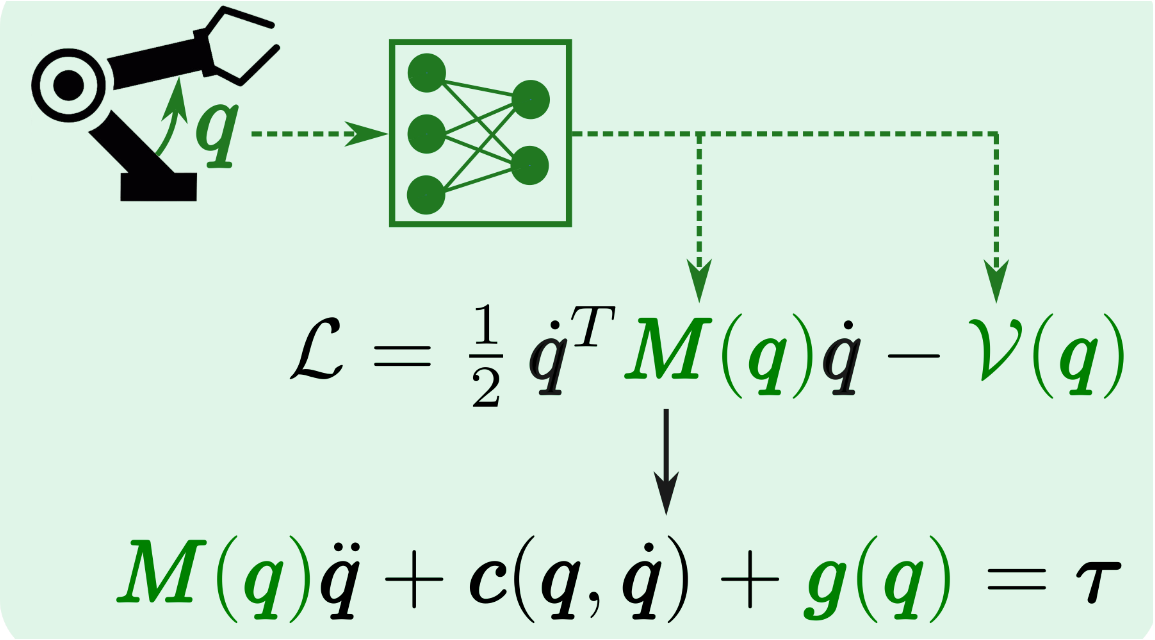
M. Lutter, C. Ritter, J. Peters, “Deep Lagrangian Networks Using Physics as Model Prior for Deep Learning”, *ICLR*, 2019

Deep Lagrangian Networks (DeLaN)

What about the **centripetal** and **Coriolis forces** $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$?

We don't need to learn them, as they can be computed from $\boldsymbol{M}(\boldsymbol{q})$ and $\dot{\boldsymbol{q}}$.

Remember their definition: $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \; = \; \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{q}}(\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}})\right)^T$

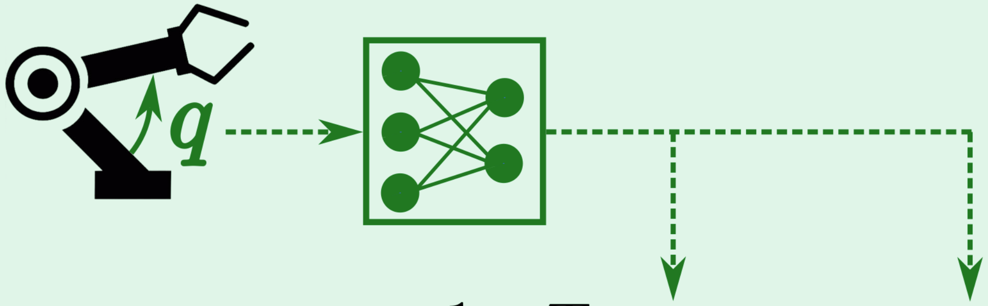


M. Lutter, C. Ritter, J. Peters, "Deep Lagrangian Networks Using Physics as Model Prior for Deep Learning", *ICLR*, 2019

Deep Lagrangian Networks (DeLaN)

Ensuring physical consistency: the mass matrix $\boldsymbol{M}(\boldsymbol{q})$ must be symmetric and positive definite ($\boldsymbol{M}(\boldsymbol{q}) \succ 0$) ➔ learn its Cholesky factoriz. rather than a full matrix

$\boldsymbol{M}(\boldsymbol{q}) = \boldsymbol{L}(\boldsymbol{q})\,\boldsymbol{L}(\boldsymbol{q})^T$ with $\boldsymbol{L}(\boldsymbol{q})$ = learnable lower-triangular matrix



$$\mathcal{L} = \frac{1}{2}\,\dot{q}^T M(q)\dot{q} - \mathcal{V}(q)$$

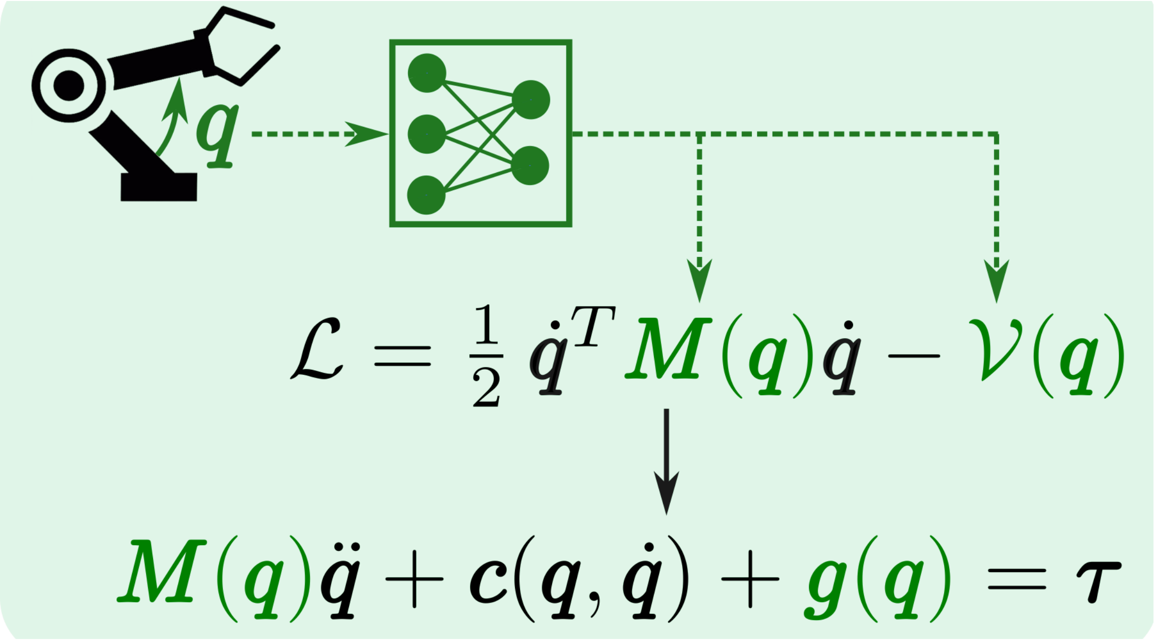
$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = \tau$$

M. Lutter, C. Ritter, J. Peters, "Deep Lagrangian Networks Using Physics as Model Prior for Deep Learning", *ICLR*, 2019

Deep Lagrangian Networks (DeLaN)

Learn $M(q)$ and $g(q)$ with ***supervised learning***: minimize the deviations between

- The predicted generalized forces/moments τ computed from
  $\tau = M(q)\ddot{q} + c(q,\dot{q}) + g(q)$
- The actual actions $\hat{\tau}$ applied to the physical system ➔ loss function $||\tau - \hat{\tau}||^2$



M. Lutter, C. Ritter, J. Peters, "Deep Lagrangian Networks Using Physics as Model Prior for Deep Learning", *ICLR*, 2019

Deep Lagrangian Networks (DeLaN) - Examples

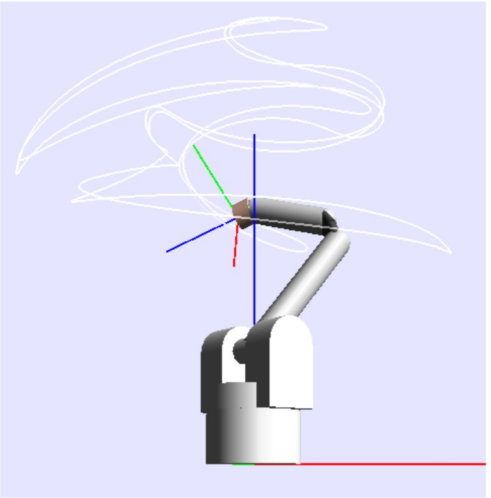
Inverse dynamics learning for robot control

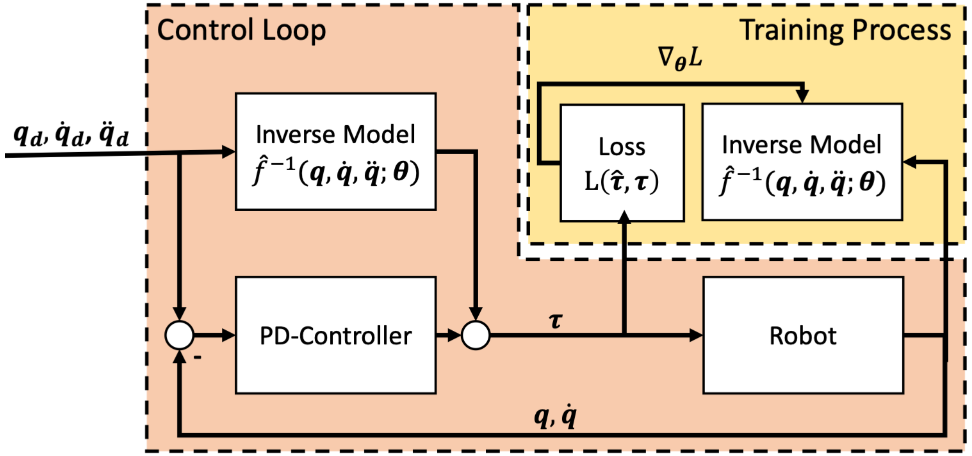
- Trajectory tracking with a 7-DoF robot
- Desired joint pos., vel., acc. $\{q_d, \dot{q}_d, \ddot{q}_d\}$
- Feedforward + feedback (PD) control

$$\boldsymbol{\tau} = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + \boldsymbol{\tau}_{\mathrm{ff}}$$

$$\boldsymbol{\tau}_{\mathrm{ff}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$$

with $\boldsymbol{\tau}$ = motor torques at the joints





M. Lutter, C. Ritter, J. Peters, “Deep Lagrangian Networks Using Physics as Model Prior for Deep Learning”, *ICLR*, 2019

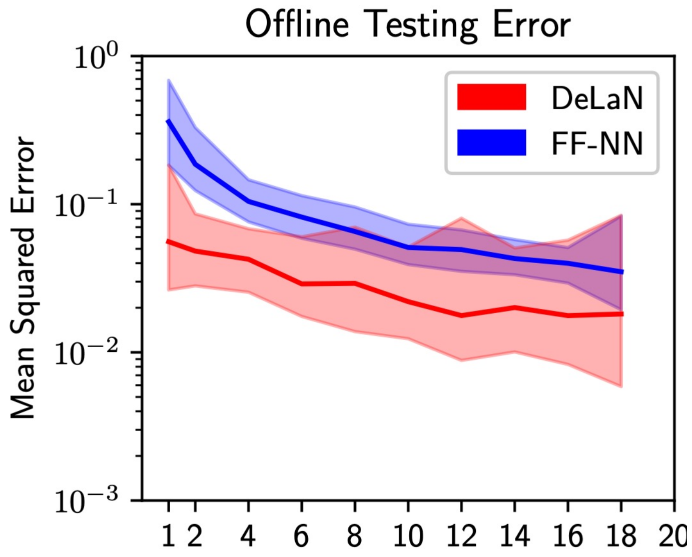
Deep Lagrangian Networks (DeLaN) - Examples

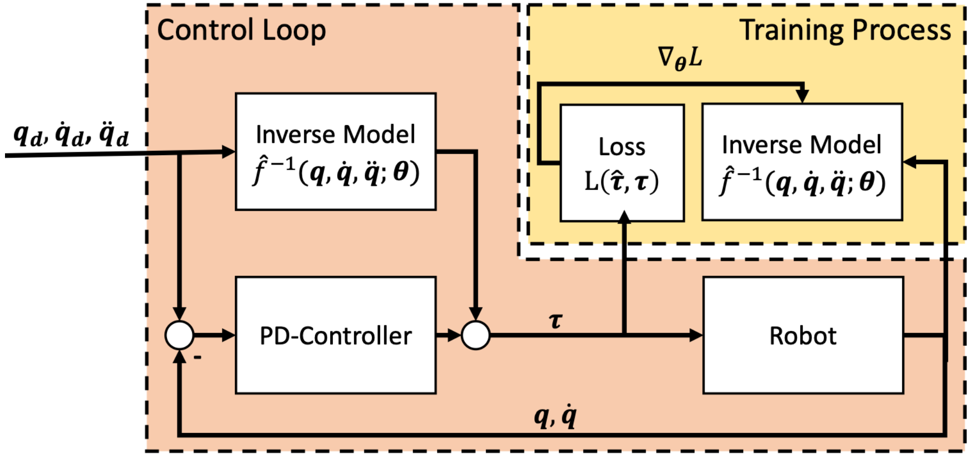
Inverse dynamics learning for robot control

- Trajectory tracking with a 7-DoF robot
- Desired joint pos., vel., acc. $\{q_d, \dot{q}_d, \ddot{q}_d\}$
- Feedforward + feedback (PD) control

$$\boldsymbol{\tau} = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + \boldsymbol{\tau}_{\mathrm{ff}}$$
$$\boldsymbol{\tau}_{\mathrm{ff}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$$

- Outperforming a generic feedforward NN (FF-NN) & **better generalization** to new data

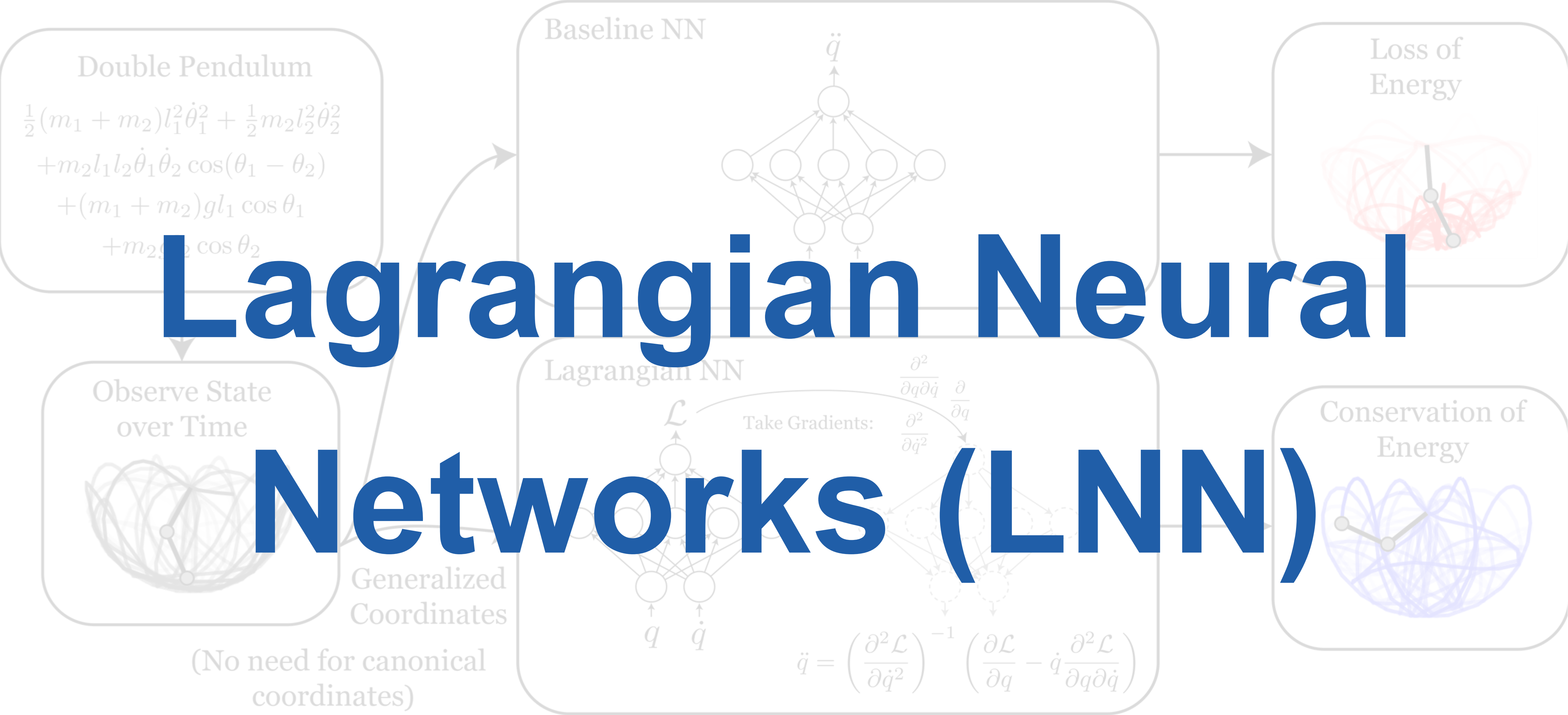




M. Lutter, C. Ritter, J. Peters, “Deep Lagrangian Networks Using Physics as Model Prior for Deep Learning”, *ICLR*, 2019

Questions & Discussion

Lagrangian Neural Networks (LNN)



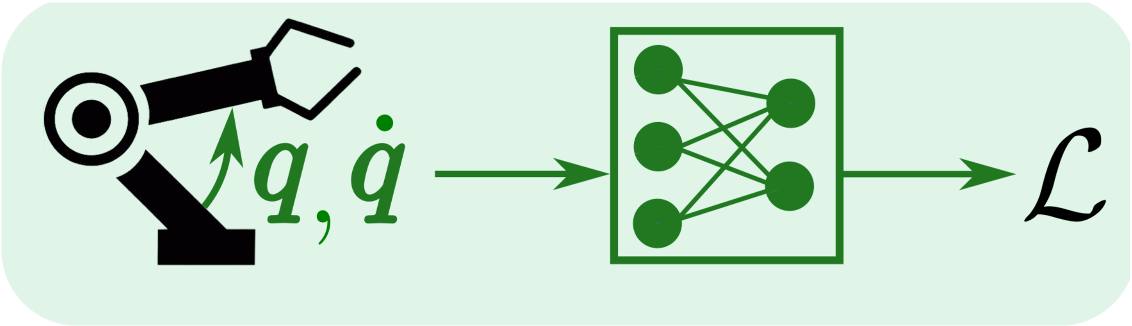
Lagrangian Neural Networks (LNN)

Idea: Learn the **whole Lagrangian** $\mathcal{L}$ from measured generalized coordinates $\boldsymbol{q}, \dot{\boldsymbol{q}}$

- **DeLaNs** assume **specific structures** for $\mathcal{L}$ and the kinetic energy T, which hold for rigid body dynamics but not for other systems (*e.g.*, charged electric particle)

$$\mathcal{L} = T(\boldsymbol{q}, \dot{\boldsymbol{q}}) - V(\boldsymbol{q}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - V(\boldsymbol{q})$$

- In contrast, **LNNs** can learn **more general** Lagrangian structures
- **LNNs** aim to ensure **energy conservation**



M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, “Lagrangian Neural Networks”, *ICLR Workshop*, 2020

Lagrangian Neural Networks (LNN)

Compute $\ddot{\boldsymbol{q}}$ from $\boldsymbol{q}, \dot{\boldsymbol{q}}$ using the Euler-Lagrange equations:

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} = \nabla_q\mathcal{L}$$

Use the chain rule to expand $\frac{d}{dt}$ considering that $\mathcal{L} = \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})$

$$(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L})\ddot{q} + (\nabla_q\nabla_{\dot{q}}^{\top}\mathcal{L})\dot{q} = \nabla_q\mathcal{L}$$

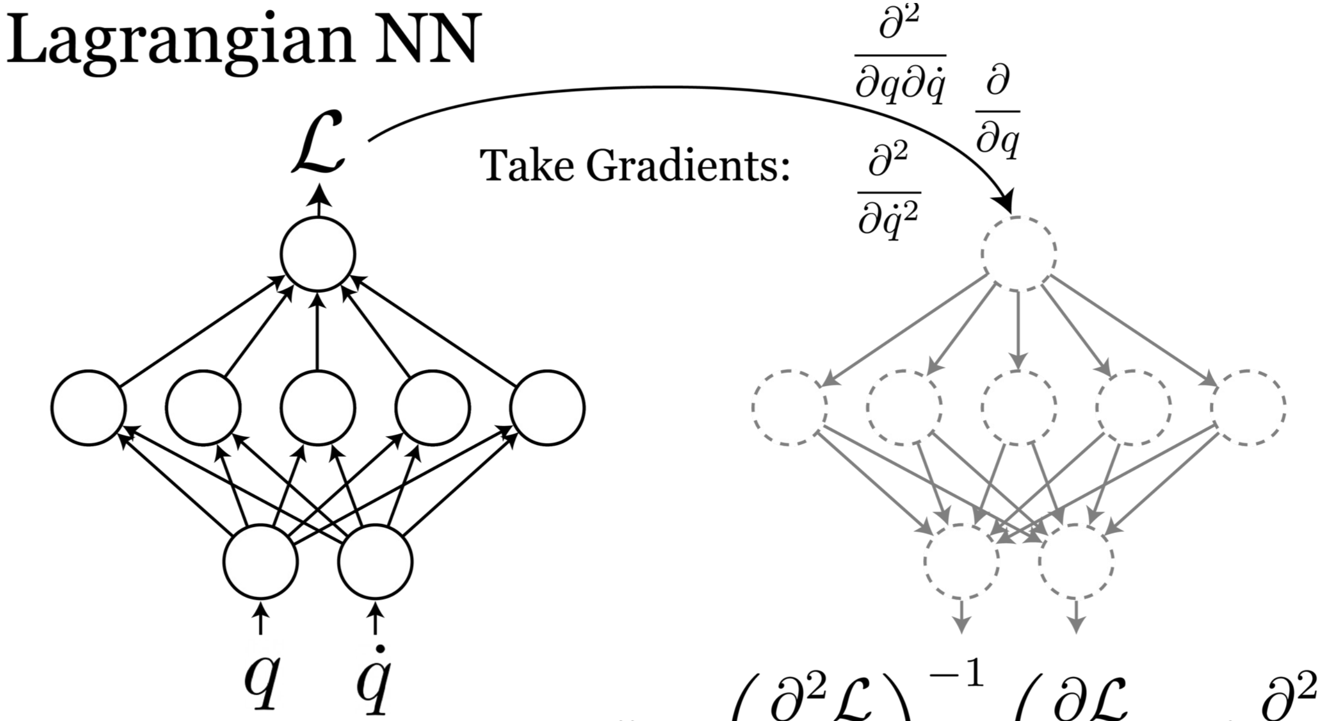
$$\boxed{\ddot{q} = (\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L})^{-1}[\nabla_q\mathcal{L} - (\nabla_q\nabla_{\dot{q}}^{\top}\mathcal{L})\dot{q}]}$$

➔ loss function $||\ddot{q} - \hat{\ddot{q}}||^2$

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, "Lagrangian Neural Networks", *ICLR Workshop*, 2020

Lagrangian Neural Networks (LNN)

$$\boxed{\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^{\top} \mathcal{L})^{-1} [\nabla_q \mathcal{L} - (\nabla_q \nabla_{\dot{q}}^{\top} \mathcal{L}) \dot{q}]}$$

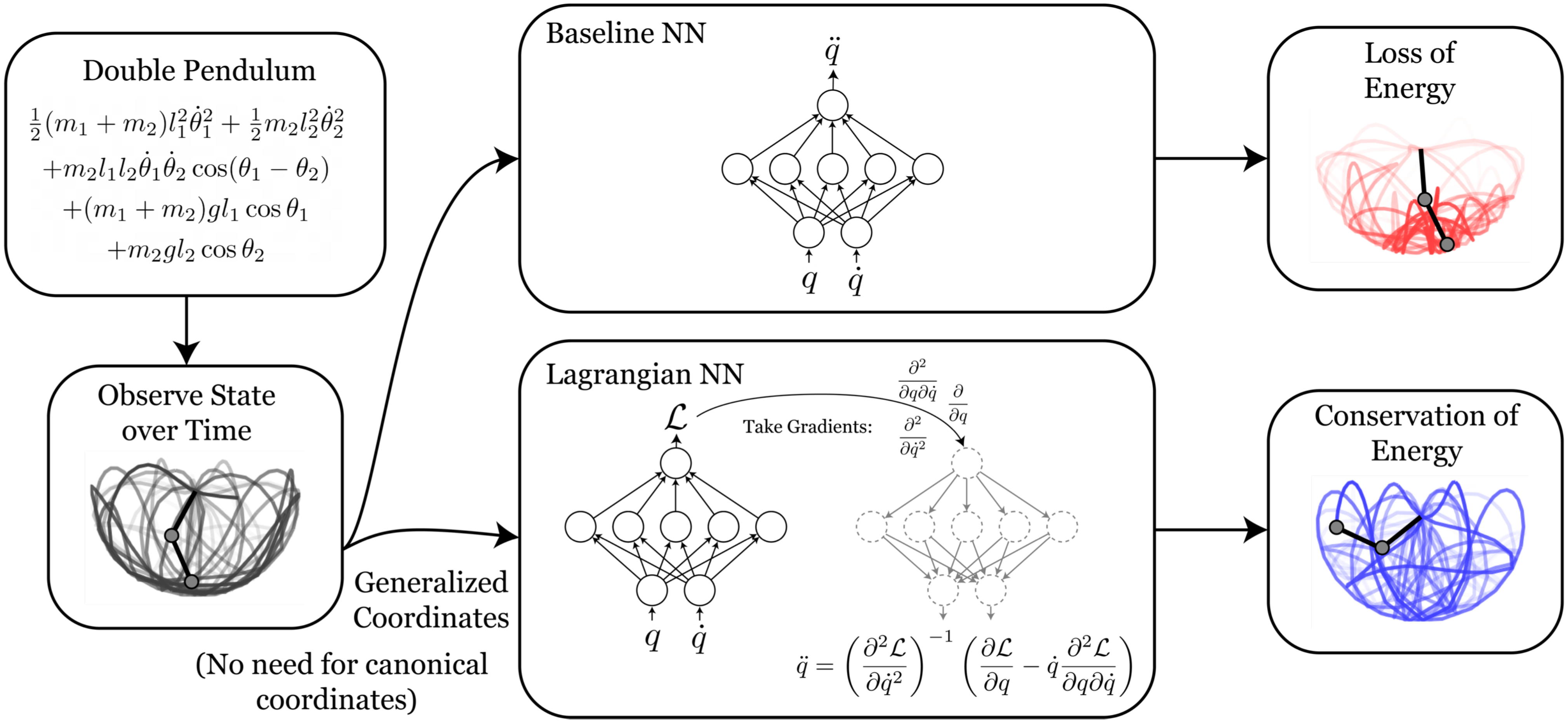


$$\ddot{q} = \left(\frac{\partial^2 \mathcal{L}}{\partial \dot{q}^2} \right)^{-1} \left(\frac{\partial \mathcal{L}}{\partial q} - \dot{q} \frac{\partial^2 \mathcal{L}}{\partial q \partial \dot{q}} \right)$$

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, “Lagrangian Neural Networks”, *ICLR Workshop*, 2020

Lagrangian Neural Networks (LNN)

General-purpose NNs lose energy over time ⟷ LNNs conserve energy

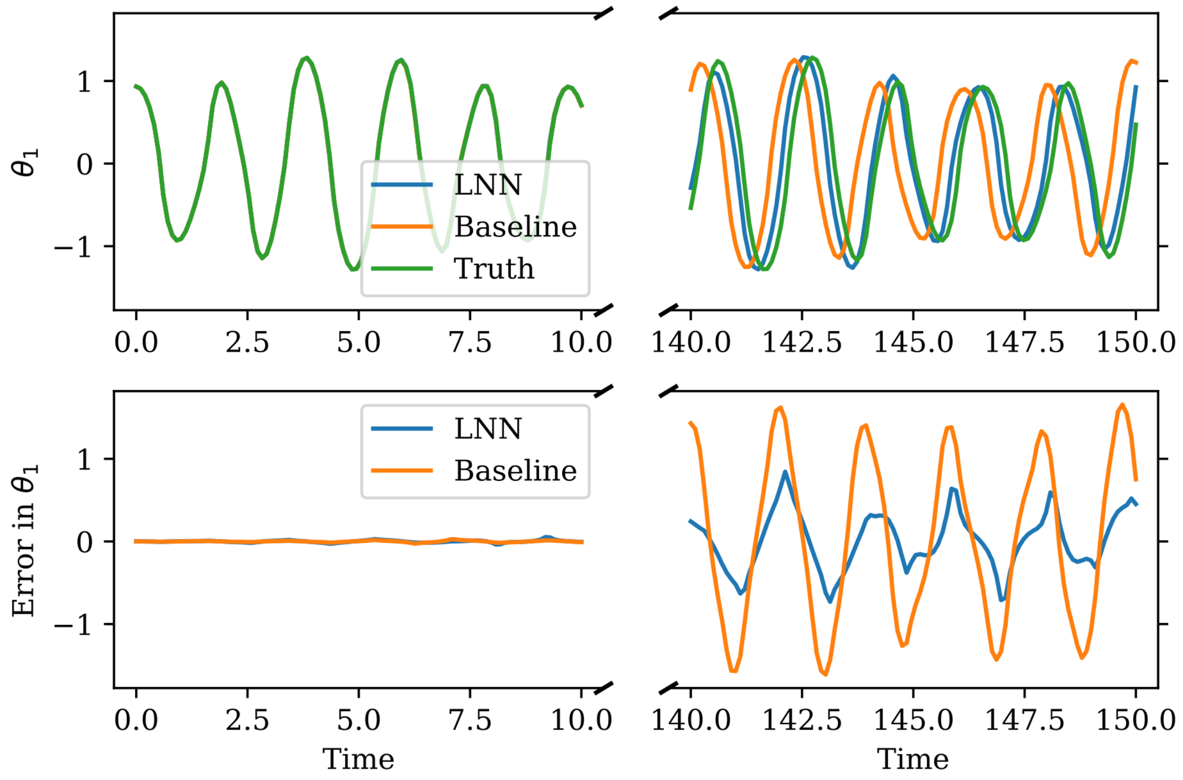


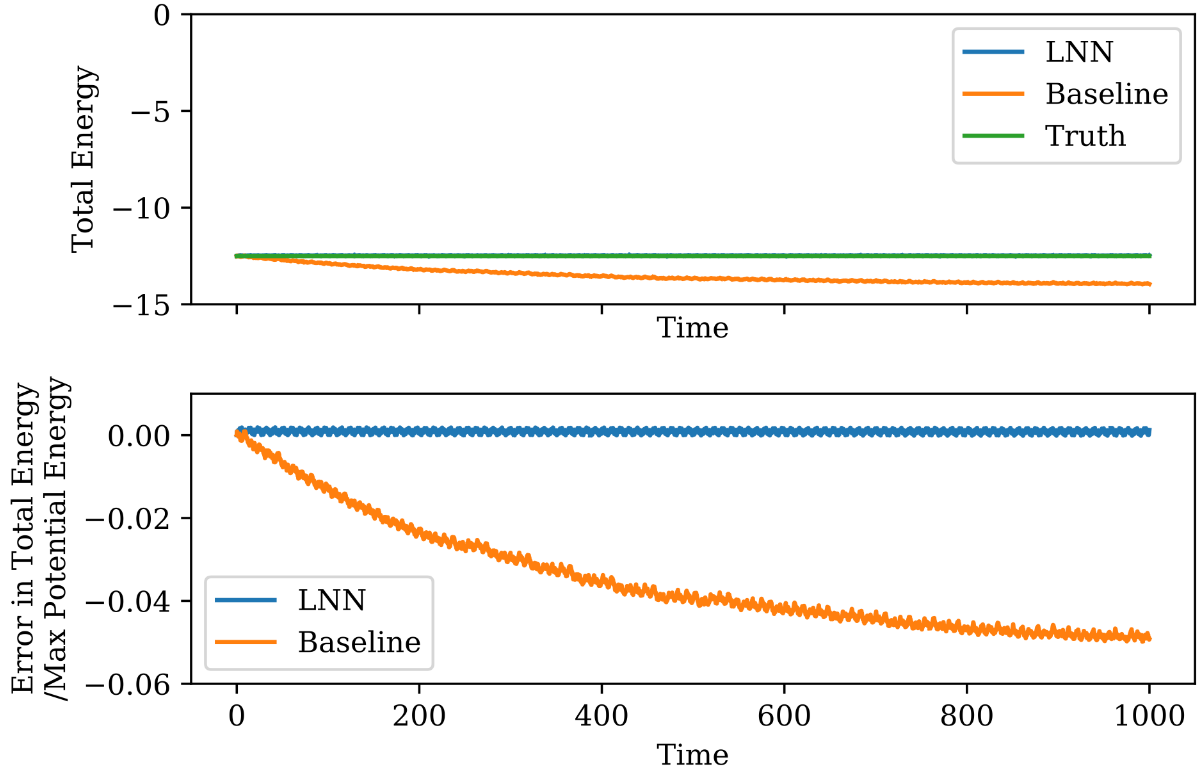
M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, "Lagrangian Neural Networks", *ICLR Workshop*, 2020

Lagrangian Neural Networks (LNN) - Examples

General-purpose NNs lose energy over time ⟷ **LNNs conserve energy**

Example: Dynamics learning with a **double pendulum**

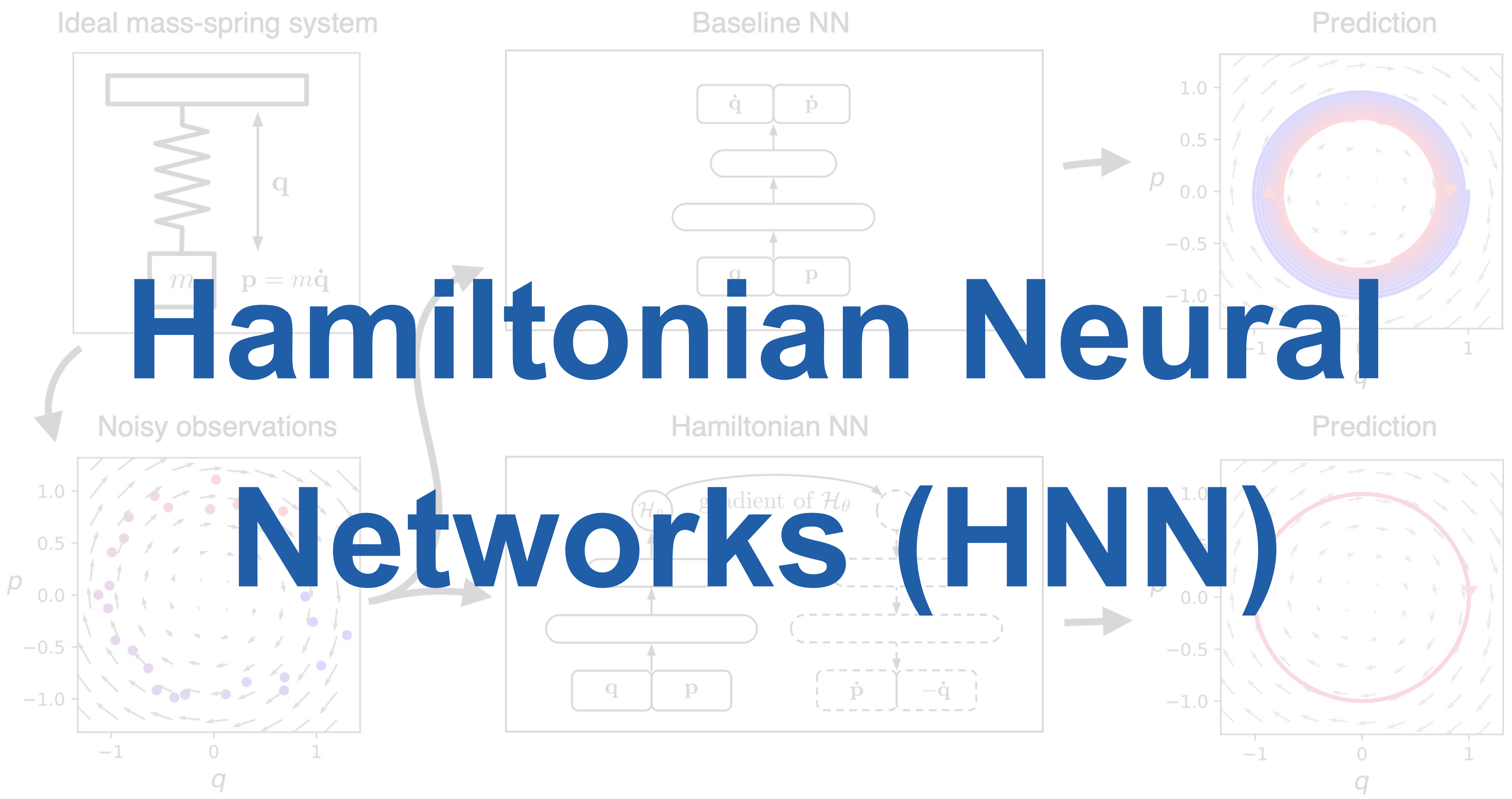




M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, "Lagrangian Neural Networks", *ICLR Workshop*, 2020

Questions & Discussion

Hamiltonian Neural Networks (HNN)



Formulating Equations of Motion

- **Newton** formulation
  - $\sum_i F_i = m\, a, \quad \sum_i M_i = I\, \alpha$ for each body
  - Explicit modeling of **forces** and **moments**

- **Lagrangian** formulation
  - **Energy**-based approach $\mathcal{L} = T - V$
  - No need to directly model forces and moments
  - More suited for **multibody systems** with complex kinematics

- **Hamiltonian** formulation
  - **Energy**-based approach, similar to the Lagrangian
  - Suited for systems with many DoFs (quantum systems, celestial mechanics, fluid dynamics, …)

Hamiltonian Equations of Motion

For a physical system, the **Hamiltonian $\mathcal{H}$** is related to the **Lagrangian $\mathcal{L}$** through the Legendre transformation:

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \dot{\mathbf{q}}^{\mathrm{T}} \frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} - \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})$$

With $\boldsymbol{q}$ = generalized coordinates, $\boldsymbol{p}$ = momentum

Hence, the **Euler-Lagrange** equations can be **rewritten** using $\mathcal{H}$:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \boldsymbol{\tau} \quad \Rightarrow \quad \boxed{\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} + \boldsymbol{\tau}}$$

With $\boldsymbol{\tau}$ = generalized (non-conservative) forces
→ in the autonomous conservative case, $\boldsymbol{\tau} = 0$ and $\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}$

S. Greydanus, M. Dzamba, J. Yosinski, “Hamiltonian Neural Networks”, *NeurIPS*, 2019

Hamiltonian Equations of Motion

The Hamiltonian $\mathcal{H}$ can be designed to be equal to the **total system's energy**:

$$\mathcal{H} = E_{\mathrm{tot}} \,, \quad \text{and typically} \quad \mathcal{H} = T + V$$

with T = kinetic energy, V = potential energy

Designing $\mathcal{H}$ **requires domain knowledge** and can be hard for **complex systems**

↓

*Can we **learn $\mathcal{H}$ from data** and ensure **energy conservation**?*

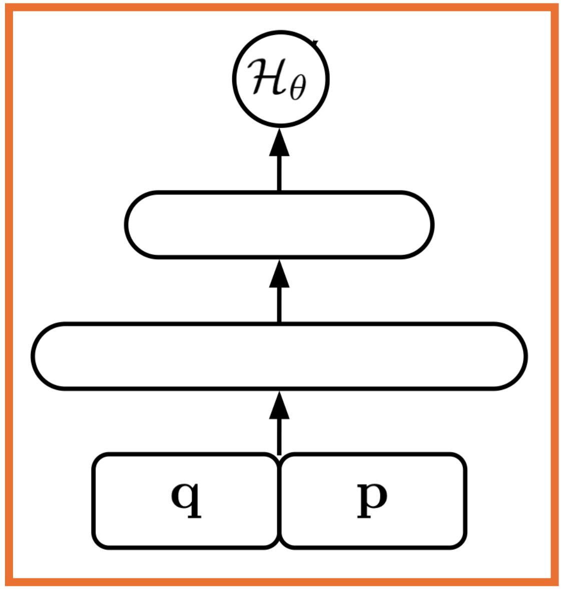
S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN)

Idea: Learn the **Hamiltonian** $\mathcal{H}$ **from data** → parametrized model $\mathcal{H}_\theta$

- Goal: satisfy the Hamiltonian equations $\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}$
- ➔ Loss function: $\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|_2$ Note: $\frac{\partial \boldsymbol{q}}{\partial t} = \dot{\boldsymbol{q}}, \ \frac{\partial \boldsymbol{p}}{\partial t} = \dot{\boldsymbol{p}}$

NN to learn $\mathcal{H}_\theta$



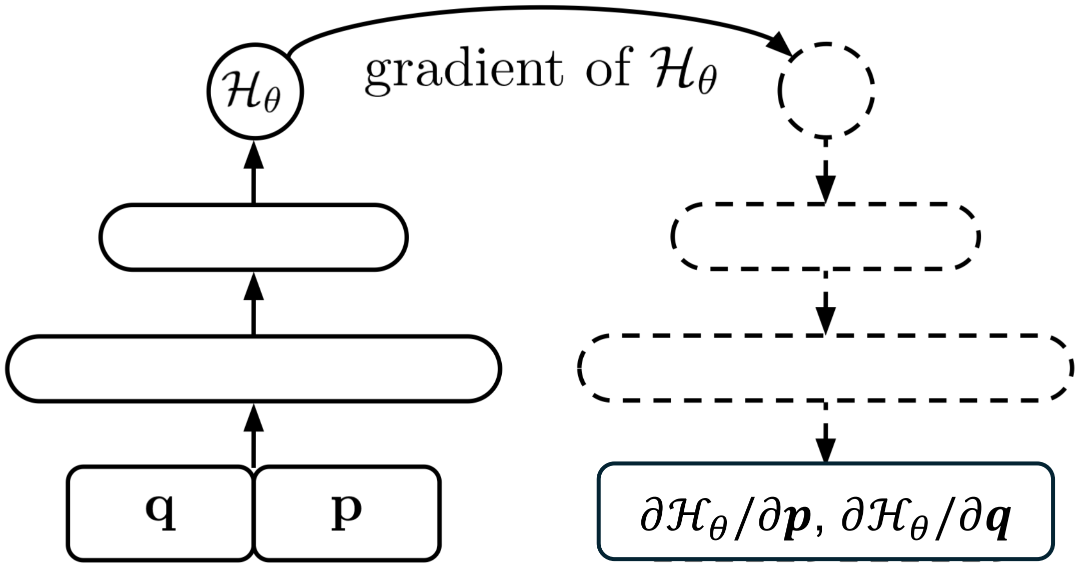
S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN)

Idea: Learn the **Hamiltonian $\mathcal{H}$ from data** → parametrized model $\mathcal{H}_\theta$

- Goal: satisfy the Hamiltonian equations $\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}$
- ➔ Loss function: $\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|_2$

NN to learn $\mathcal{H}_\theta$



Get $\frac{\partial \mathcal{H}_\theta}{\partial \boldsymbol{p}}, \frac{\partial \mathcal{H}_\theta}{\partial \boldsymbol{q}}$ (with automatic diff.) and compute $\dot{\boldsymbol{p}}, \dot{\boldsymbol{q}}$ (numerically from $\boldsymbol{q}, \boldsymbol{p}$)

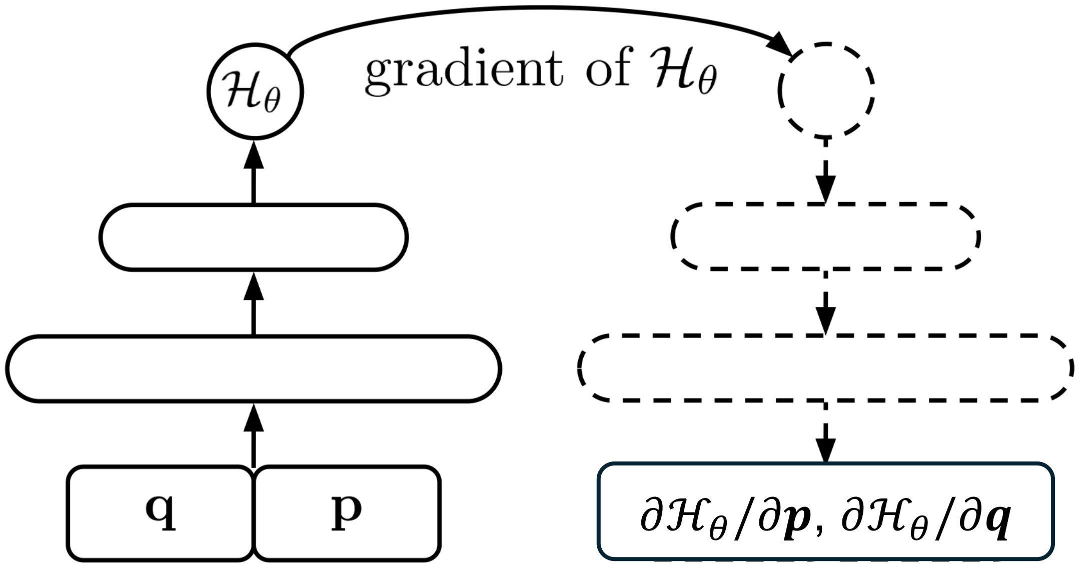
S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN)

Idea: Learn the **Hamiltonian** $\mathcal{H}$ **from data** → parametrized model $\mathcal{H}_\theta$

- Goal: satisfy the Hamiltonian equations $\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}$
- ➔ Loss function: $\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|_2$

NN to learn $\mathcal{H}_\theta$

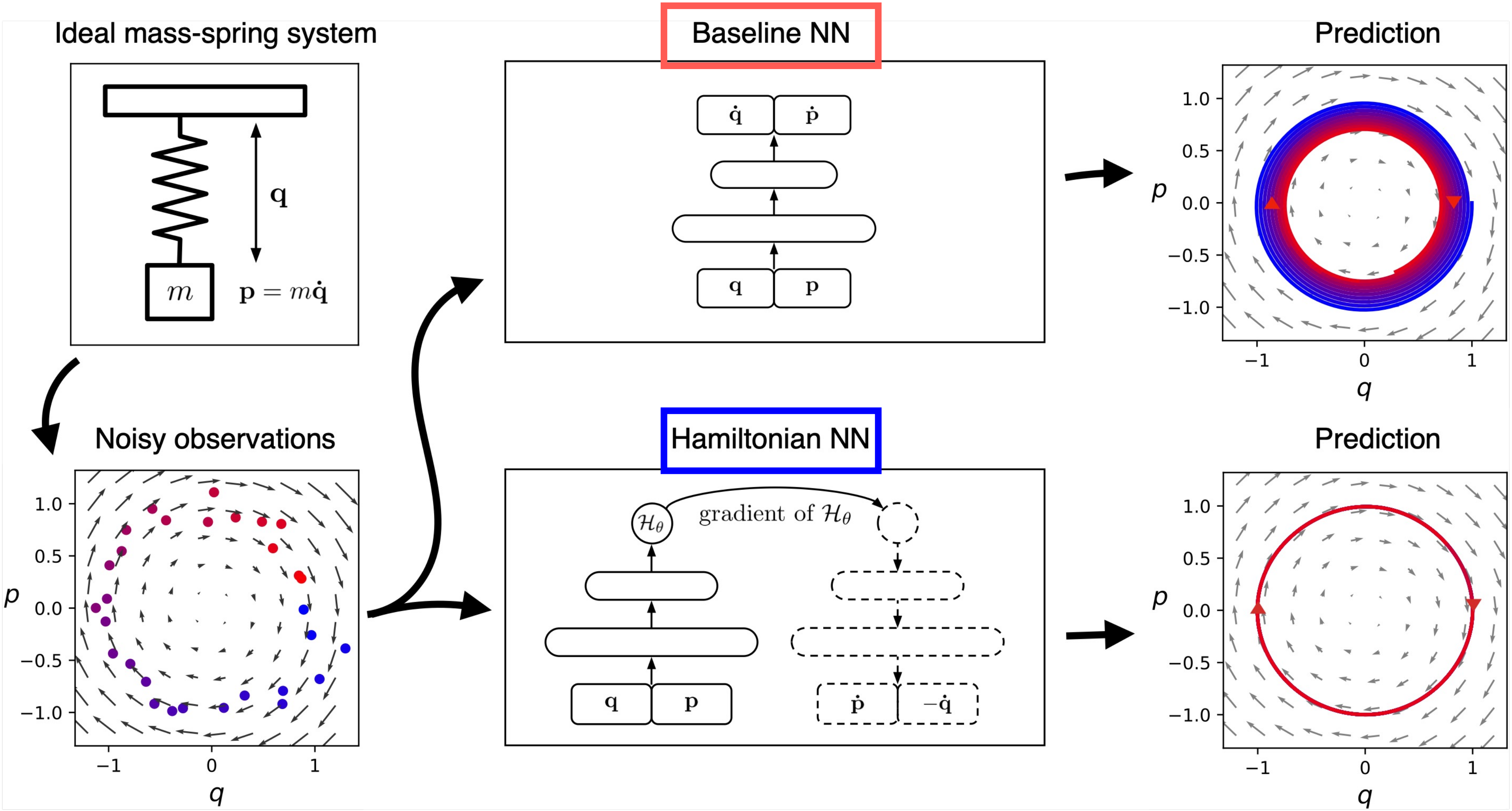


Train $\mathcal{H}_\theta$: minimize the deviations between $\frac{\partial \mathcal{H}_\theta}{\partial \boldsymbol{p}}, \frac{\partial \mathcal{H}_\theta}{\partial \boldsymbol{q}}$ and the computed (ground-truth) $\dot{\boldsymbol{p}}$, $\dot{\boldsymbol{q}}$

S. Greydanus, M. Dzamba, J. Yosinski, “Hamiltonian Neural Networks”, *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN) - Examples

General-purpose NNs lose energy over time ⟺ **HNNs conserve energy**

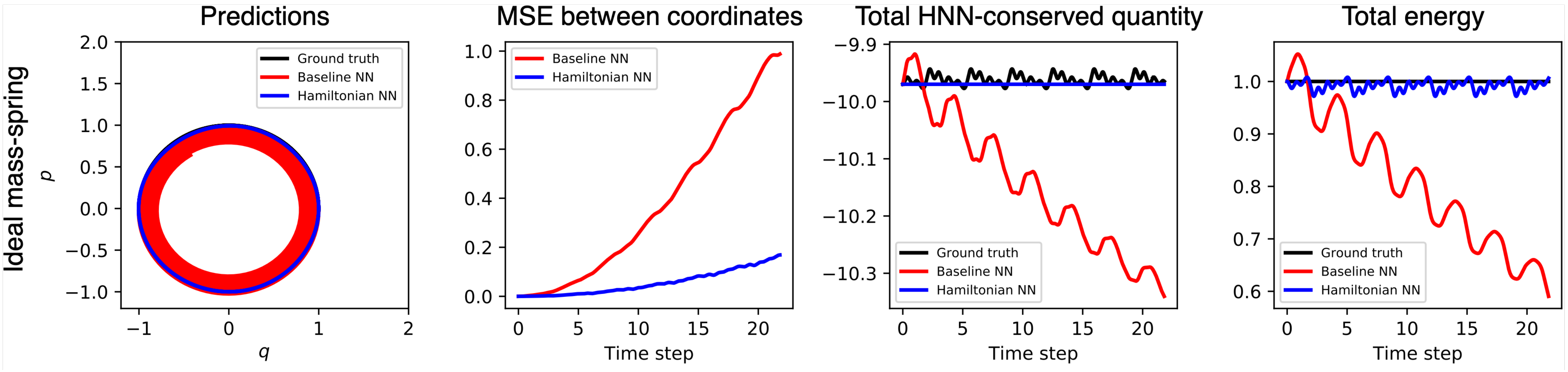


S. Greydanus, M. Dzamba, J. Yosinski, “Hamiltonian Neural Networks”, *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN) - Examples

General-purpose NNs lose energy over time ⟷ **HNNs conserve energy**

Frictionless mass-spring system

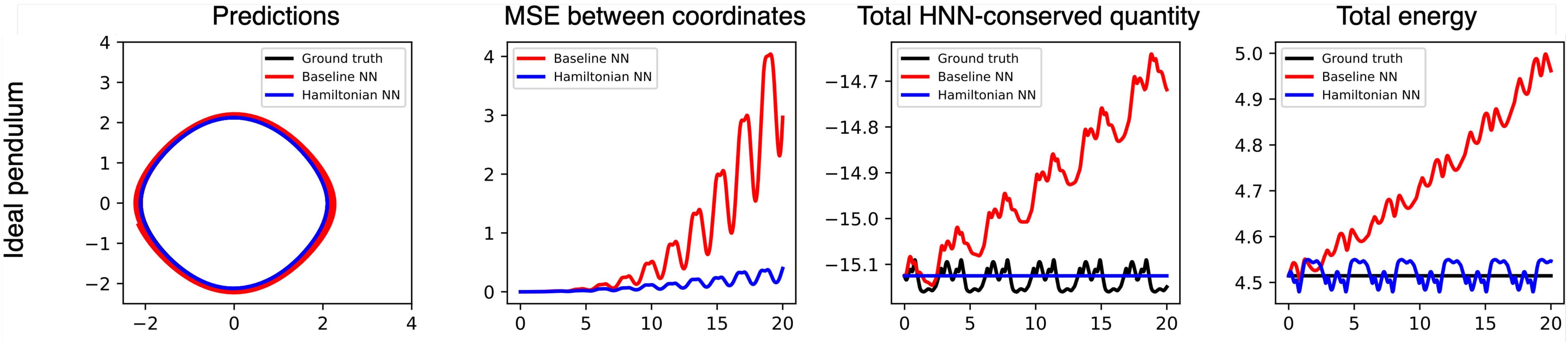


S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN) - Examples

General-purpose NNs lose energy over time ⟷ **HNNs conserve energy**

Frictionless pendulum

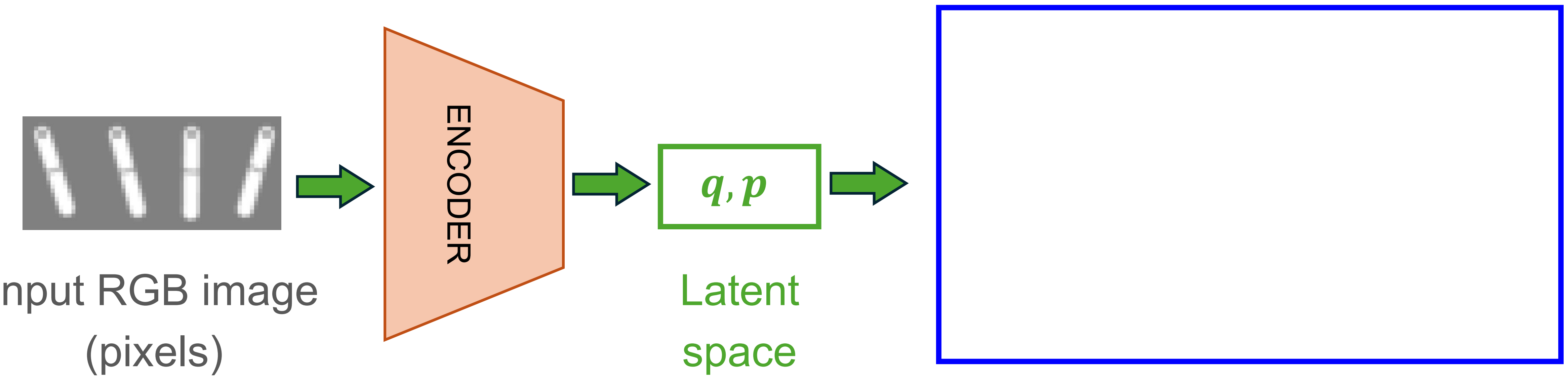


S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN) - Examples

Can a Hamiltonian $\mathcal{H}_\theta$ be **learned from pixel** observations?

- Coupling an **HNN** with an **autoencoder** to map pixels to a compact latent space
- Instead of training the HNN on the position ($\boldsymbol{q}$) and momentum ($\boldsymbol{p}$) coordinates, use the **latent representation** of the input image

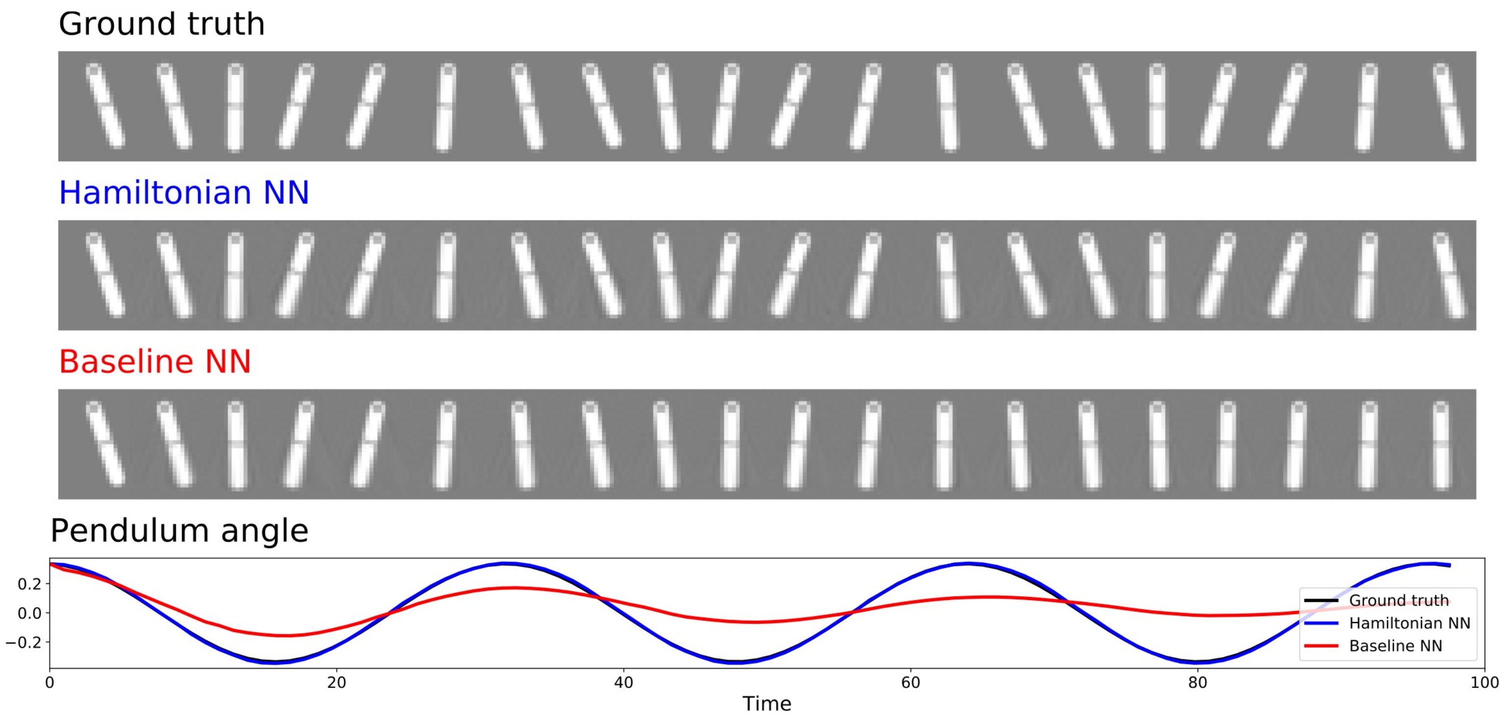


S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Hamiltonian Neural Networks (HNN) - Examples

Can a Hamiltonian $\mathcal{H}_\theta$ be **learned from pixel** observations?

- Coupling an **HNN** with an **autoencoder** to map pixels to a compact latent space
- Instead of training the HNN on the position ($\boldsymbol{q}$) and momentum ($\boldsymbol{p}$) coordinates, use the **latent representation** of the input image

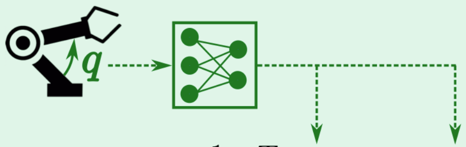


S. Greydanus, M. Dzamba, J. Yosinski, "Hamiltonian Neural Networks", *NeurIPS*, 2019

Questions & Discussion

Physics-Encoded Learning Architectures: Multiple Classes

Lagrangian & Hamiltonian Approaches



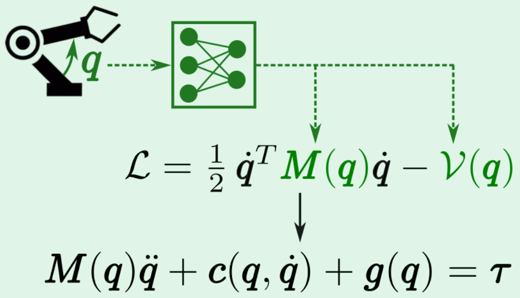
$$\mathcal{L} = \frac{1}{2}\,\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \mathcal{V}(\boldsymbol{q})$$

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

Learning dynamics
from variational principles

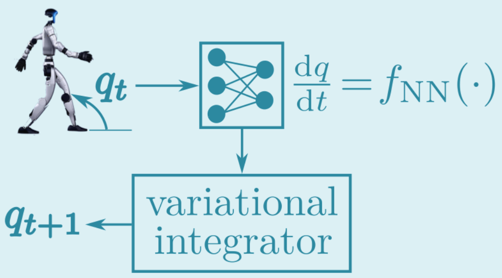
Physics-Encoded Learning Architectures: Multiple Classes

Lagrangian & Hamiltonian Approaches



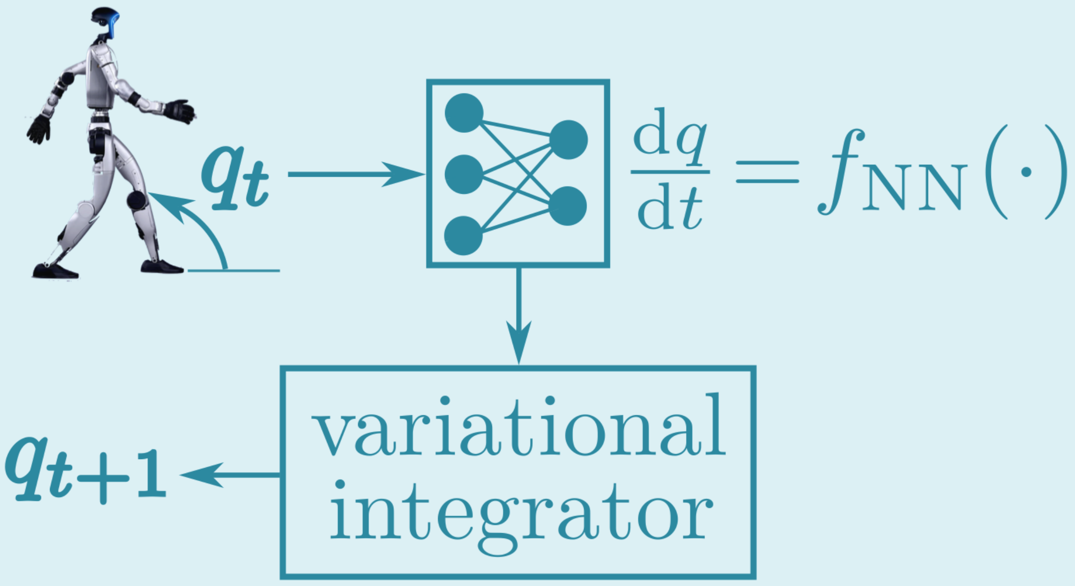
Learning dynamics from variational principles

NODE & Variational Integrator Networks



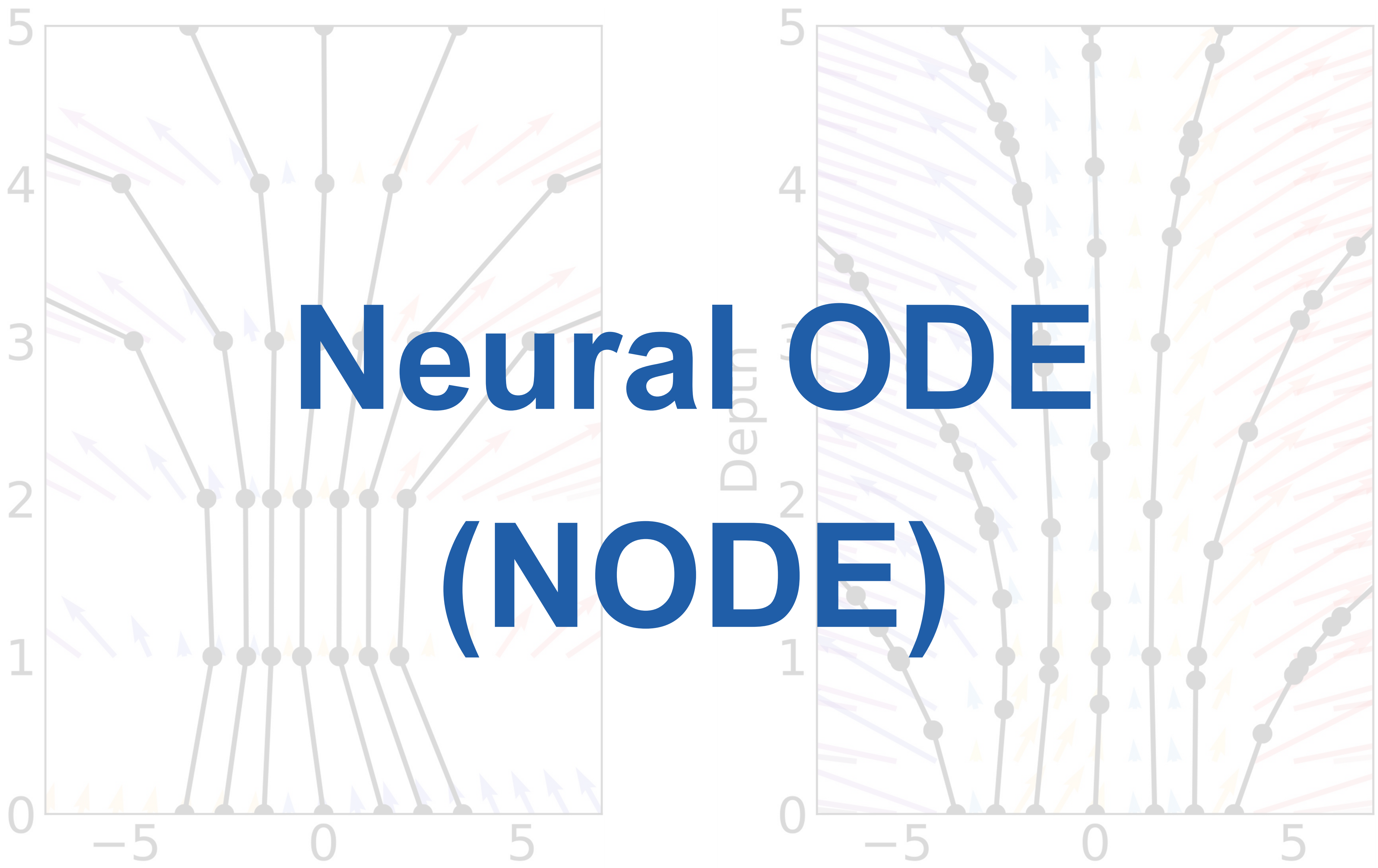
Learning dynamics via structure-preserving integration

NODE & Variational Integrator Networks



Learning dynamics via structure-preserving integration

Neural ODE
(NODE)



Neural ODE (NODE)

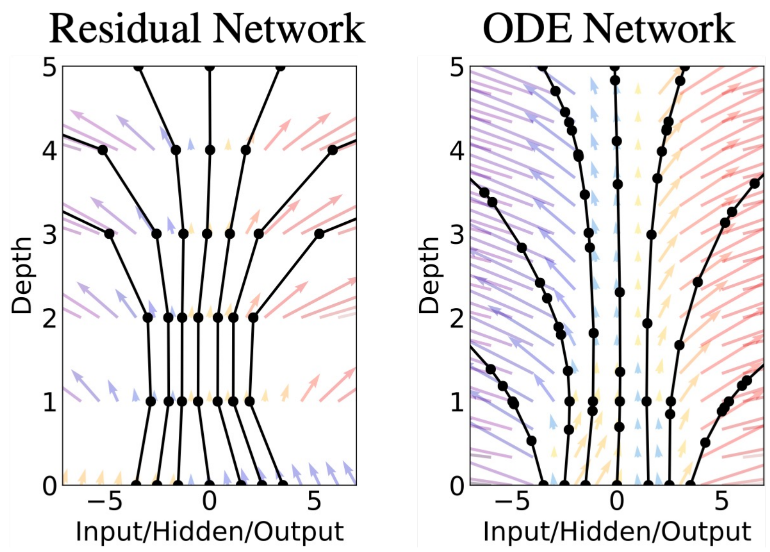
Consider a continuous-time Ordinary Differential Equation (ODE)

$$\frac{d\mathbf{h}(t)}{dt} = \boxed{f(\mathbf{h}(t), t, \theta)}$$

Neural ODEs use a neural network f to **learn** the right-hand side from data

Discrete-time case: residual network f, obtained by discretizing a NODE

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \boxed{f(\mathbf{h}_t, \theta_t)}$$



R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, “Neural Ordinary Differential Equations”, *NeurIPS*, 2018

Neural ODE (NODE)

$$\frac{d\mathbf{h}(t)}{dt} = \boxed{f(\mathbf{h}(t), t, \theta)}$$

Example loss function: $L = ||\boldsymbol{h} - \hat{\boldsymbol{h}}||^2$

Gradient-based training (simplified): $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k \nabla L_k$, with $\boldsymbol{f}$ parametrized by $\boldsymbol{\theta}$

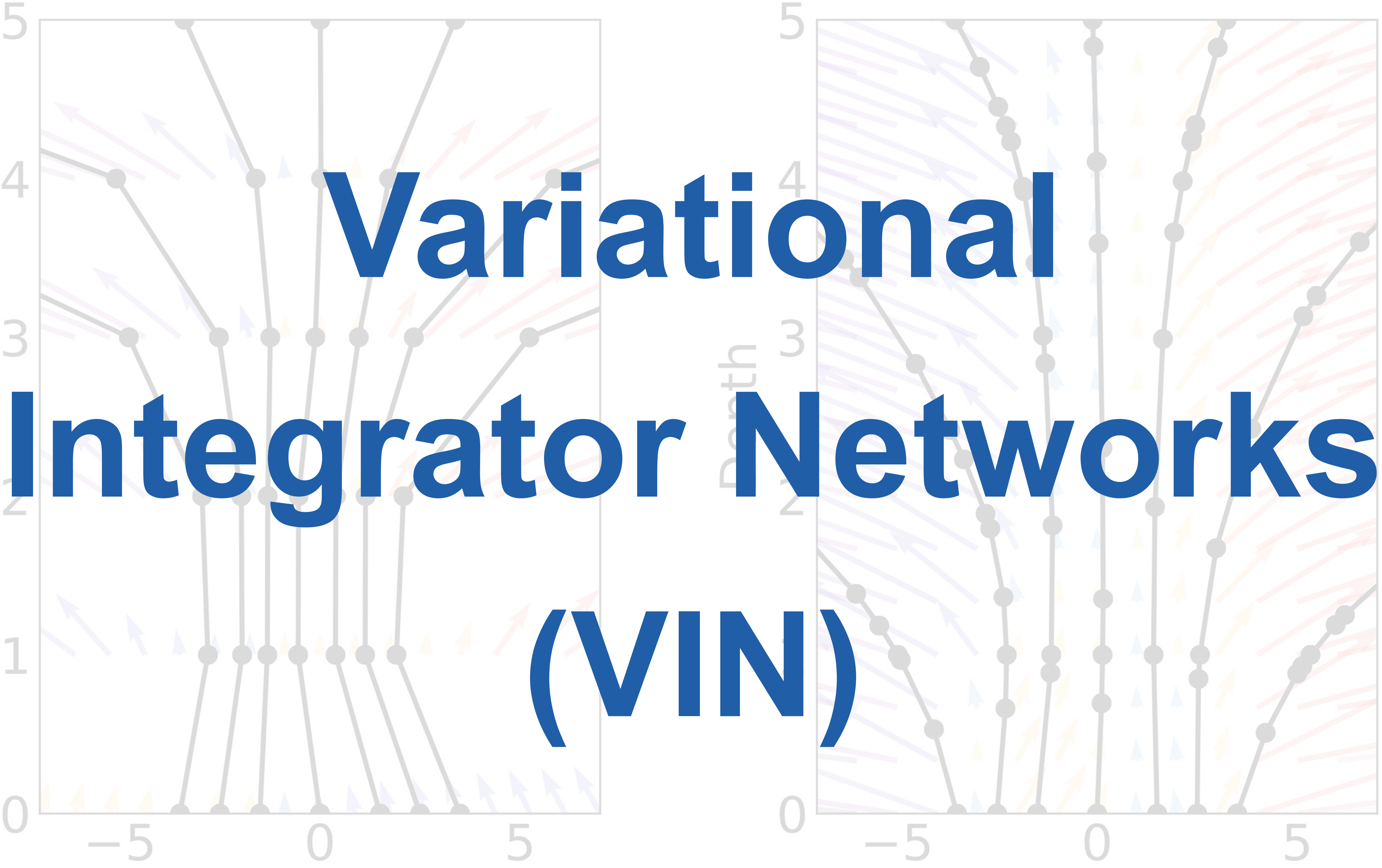
Challenge: How to backpropagate ∇L_k through an ODE solver?

Solution:

- Neural ODEs compute ∇L_k via the **adjoint sensitivity** method for **automatic differentiation**, which applies to ***any*** ODE solver (Euler, Runge-Kutta, …)
- Scales linearly with problem size, has low memory cost, and provides numerical error control

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, "Neural Ordinary Differential Equations", *NeurIPS*, 2018

Variational Integrator Networks (VIN)



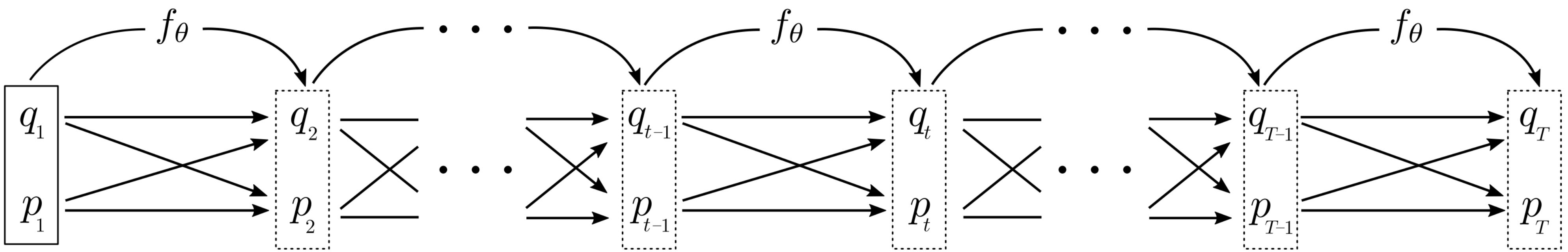
Variational Integrator Networks (VIN)

Limitation of Neural ODEs: Euler / Runge-Kutta solution schemes, which may **not** accurately preserve the **geometric** and **energy** properties of **physical** systems

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Idea of VIN: Use **structure-preserving** Variational Integrators (VIs) instead of a standard ODE solver to solve **Euler-Lagrange** ODEs

- **VIs** are ***symplectic* integrators**: they approximately **conserve energy** & **momentum** (introducing 3rd or higher-order errors)

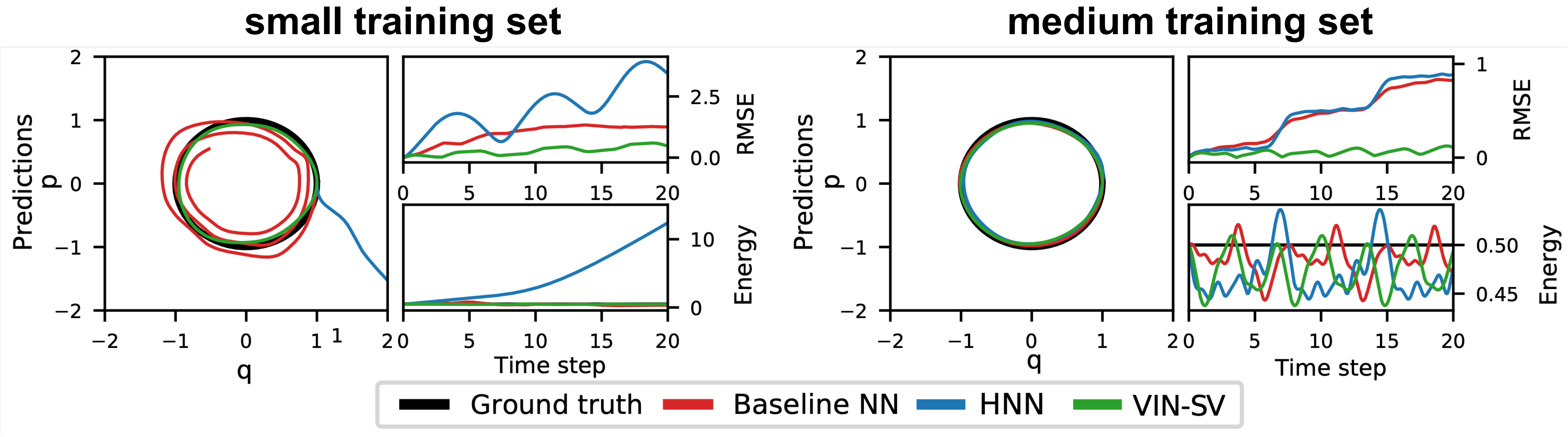


S. Sæmundsson, A. Terenin, K. Hofmann, M. P. Deisenroth "Variational Integrator Networks for Physically Structured Embeddings", *International Conference on Artificial Intelligence and Statistics*, 2020

Variational Integrator Networks (VIN) - Examples

Frictionless mass-spring system with ***noisy*** observations $[q_t, p_t, \dot{q}_t, \dot{p}_t]$ (q being position and q being momentum):

- With a small training set, **Hamiltonian NN** (**HNN**) tends to overfit and underperform
- **VIN** outperforms **HNN** and a **general-purpose NN**



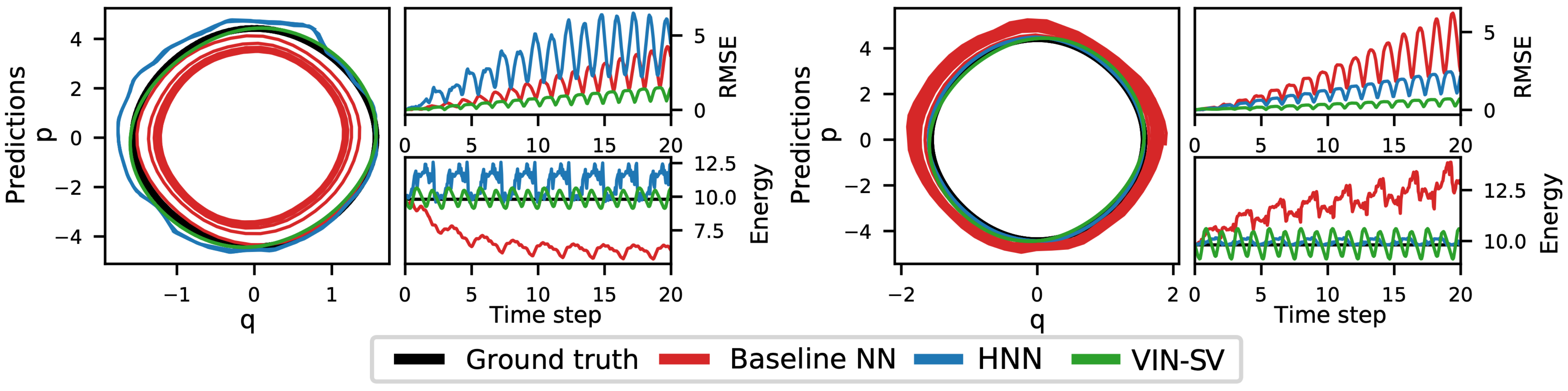
S. Sæmundsson, A. Terenin, K. Hofmann, M. P. Deisenroth "Variational Integrator Networks for Physically Structured Embeddings", *International Conference on Artificial Intelligence and Statistics*, 2020

Variational Integrator Networks (VIN) - Examples

Frictionless pendulum with ***noisy*** observations $[q_t, p_t, \dot{q}_t, \dot{p}_t]$ (q being position and q being momentum):

- With a small training set, **Hamiltonian NN** (**HNN**) tends to overfit and underperform
- **VIN** outperforms **HNN** and a **general-purpose NN**

small training set **medium training set**



S. Sæmundsson, A. Terenin, K. Hofmann, M. P. Deisenroth "Variational Integrator Networks for Physically Structured Embeddings", *International Conference on Artificial Intelligence and Statistics*, 2020

Which Approach Shall We Choose?

Comparing Neural ODE, HNN, LNN, DeLaN

How to choose?

- Do we have any **physics priors** available?
- Do we want to learn **differential equations**?
- Is **energy conservation** important?
- Is the **structure** of the **Lagrangian** known for the specific problem?

	Neural net	Neural ODE	HNN	DeLaN	LNN
Can model dynamical systems	✓	✓	✓	✓	✓
Learns differential equations		✓	✓	✓	✓
Learns exact conservation laws			✓	✓	✓
Learns from arbitrary coords.	✓	✓		✓	✓
Learns arbitrary Lagrangians					✓

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, “Lagrangian Neural Networks”, *ICLR Workshop*, 2020

Hamiltonian vs. Lagrangian Neural Networks (HNNs vs. LNNs)

In principle, **HNNs** and **LNNs** act **similarly**:

- Both learn a **function** (Hamiltonian or Lagrangian) and aim to **conserve energy**
- For HNNs, **momenta** ($\boldsymbol{p}$) can be **hard to formulate** for complex systems
- Thus, the canonical coordinates required by HNNs can be a bottleneck
- **LNNs** are more general and can learn from **arbitrary coordinates**

	Neural net	Neural ODE	HNN	DeLaN	LNN
Can model dynamical systems	✓	✓	✓	✓	✓
Learns differential equations		✓	✓	✓	✓
Learns exact conservation laws			✓	✓	✓
Learns from arbitrary coords.	✓	✓		✓	✓
Learns arbitrary Lagrangians					✓

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, "Lagrangian Neural Networks", *ICLR Workshop*, 2020

Questions & Discussion

Funded with the contribution of the Ministry of University and Research pursuant to Decree No. 1236 of 01/08/2023 – FIS 2 Call; project FIS-2023-03684 'Neu4mes', CUP: E53C24003800001