Structure Preserving Deep Learning

Matthias J. Ehrhardt

Institute for Mathematical Innovation, University of Bath, UK

January 27, 2020

Joint work with:
M. Benning (Queen Mary, UK), C. Etmann, C.-B. Schönlieb, F. Sherry (all Cambridge, UK), E. Celledoni, B. Owren (both NTNU, Norway), R. McLachlan (Massey, New Zealand)

The Leverhulme Trust

Engineering and Physical Sciences Research Council

THE FARADAY INSTITUTION

Outline

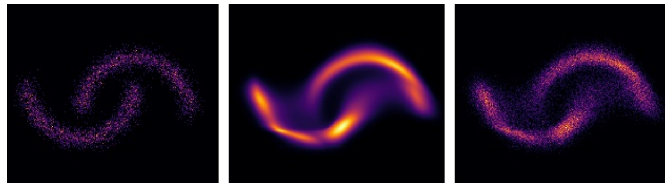
Towards deep learning with **guarantees**: stability, invertibility, equivariance, invariance, existence of solutions

Outline

Towards deep learning with **guarantees**: stability, invertibility, equivariance, invariance, existence of solutions

1) Deep learning and differential equations: Optimal control, stability and deep limits

2) Structure preserving deep neural networks: Equivariance and Invertibility



[1] Celledoni, MJE, Etmann, McLachlan, Owren, Schönlieb, Sherry, "Structure preserving deep learning," arxiv:2006.03364, 2020.

[2] Benning, Celledoni, MJE, Owren, and Schönlieb, "Deep learning as optimal control problems: models and numerical methods," J. Comput. Dyn. 6(2) 2019.

[3] Haber and Ruthotto, "Stable architectures for deep neural networks," Inverse Probl. 34(1) 2018.

Classification with Deep Learning

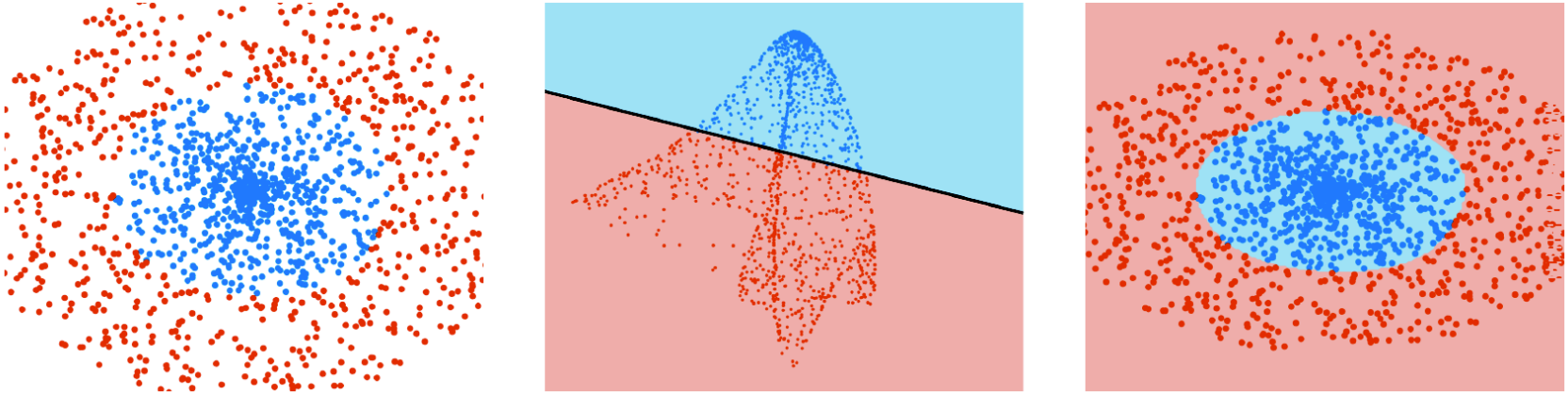
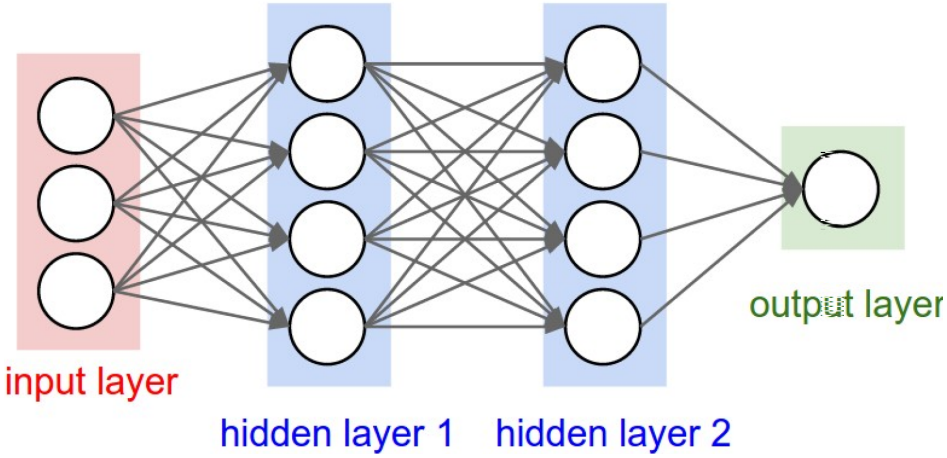


Figure courtesy of L. Ruthotto.

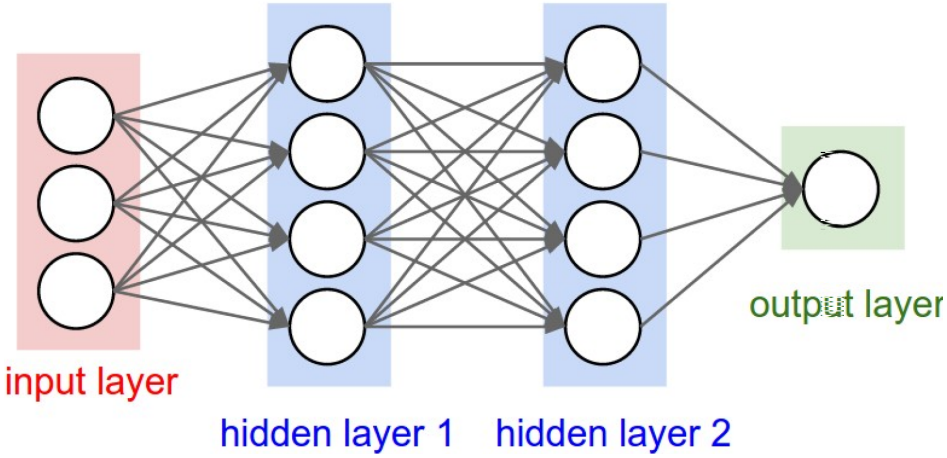
- *labeled training data* $(x_k, y_k)_{k=1,\dots,K}$
- transform data with *hidden layers*
- perform *linear classification*
- *generalization* to unseen data

Deep Learning / Deep Neural Networks (DNN)

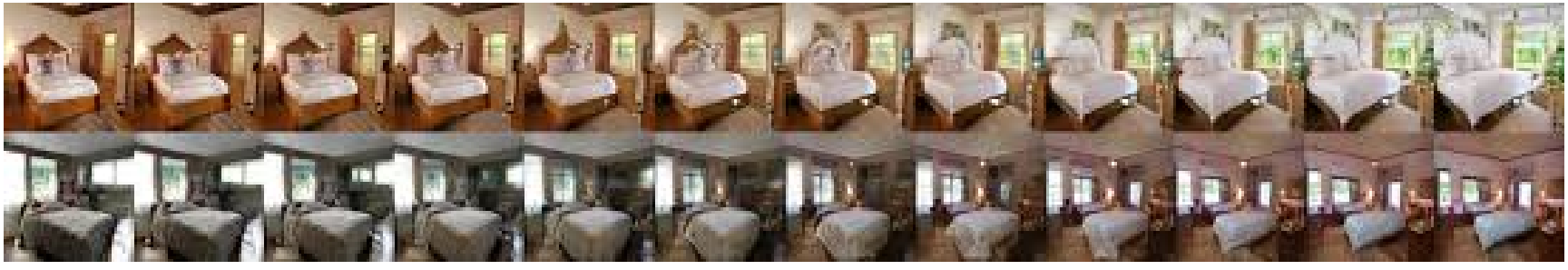


- *input layer*: given data, features, e.g. image of dog
- *hidden layers*: transform data
- *output layer* classification result, e.g. label "dog"
- invented in the 50's
- recent success by massive data and computing power
- applications: image classification, face recognition, self-driving cars, ...

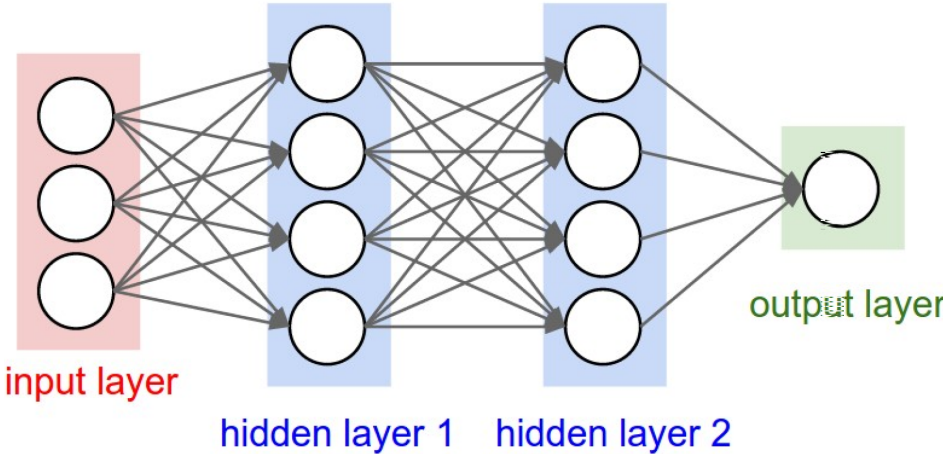
Deep Learning / Deep Neural Networks (DNN)



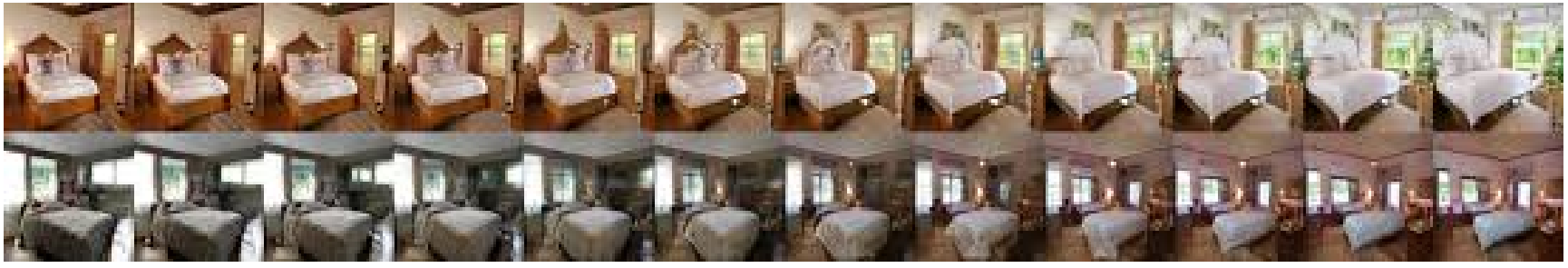
- *input layer*: given data, features, e.g. image of dog
- *hidden layers*: transform data
- *output layer* classification result, e.g. label "dog"
- invented in the 50's
- recent success by massive data and computing power
- applications: image classification, face recognition, self-driving cars, ...



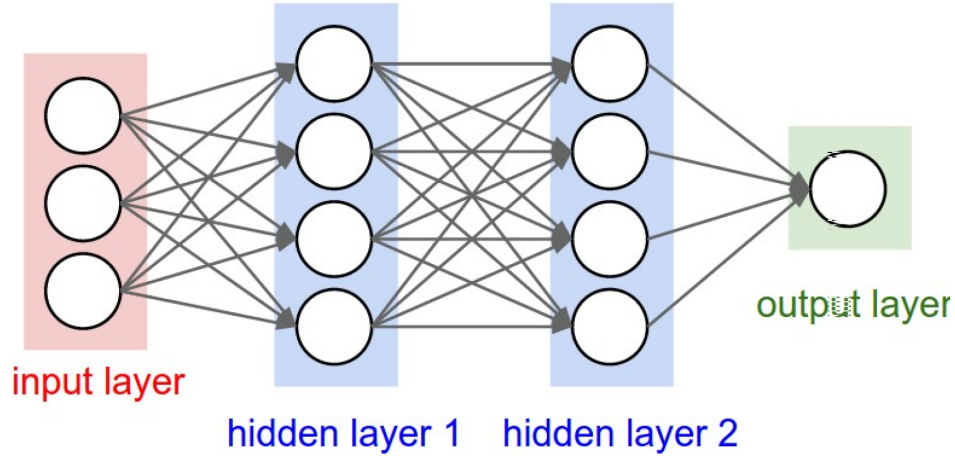
Deep Learning / Deep Neural Networks (DNN)



- *input layer*: given data, features, e.g. image of dog
- *hidden layers*: transform data
- *output layer* classification result, e.g. label "dog"
- invented in the 50's
- recent success by massive data and computing power
- applications: image classification, face recognition, self-driving cars, ...



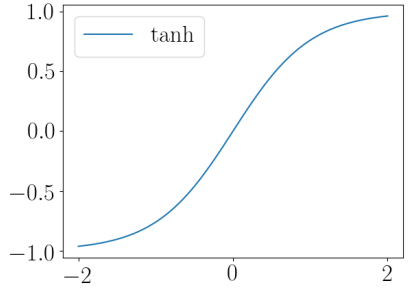
Mathematical Formulation of Deep Learning

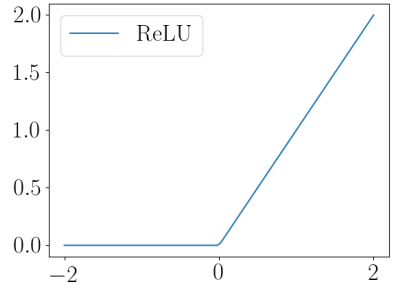


- *input layer*: $x^0 \in \mathbb{R}^n$
- *hidden layers* $\{x^k\}_{k=1,\dots,K}$, K *depth*/number of layers of DNN, *forward propagation*

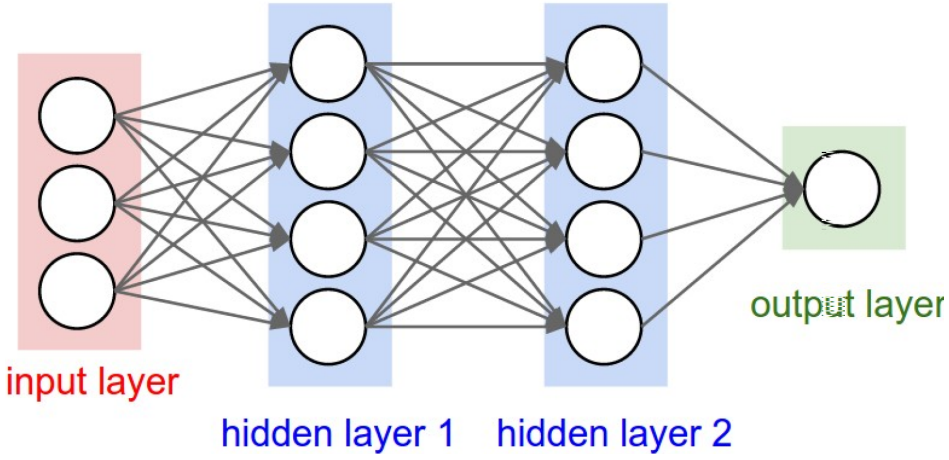
$$x^{k+1} = \sigma(A^k x^k + b^k)$$

activation function: $\sigma : \mathbb{R} \to \mathbb{R}$ applied component-wise, e.g. $\sigma(x) = \tanh(x)$, ReLU: $\sigma(x) = \max(x, 0)$





Mathematical Formulation of Deep Learning

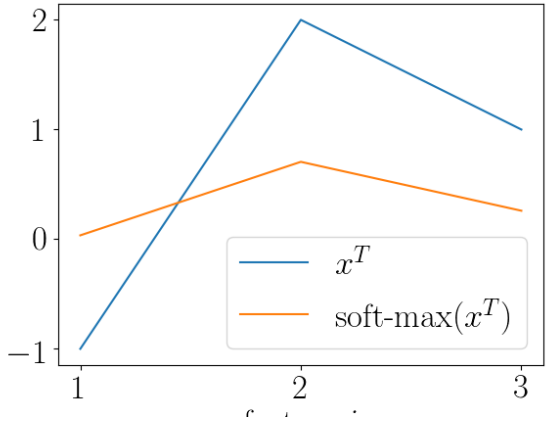


- *output layer* $y \in \mathbb{S}^{m-1}$, classification result, e.g. $m = 2$, $y = [1, 0]$ may correspond to "dog"

$$y = \tau(Wx^K + \omega)$$

$\tau : \mathbb{R}^m \to \mathbb{R}^m$, e.g. soft-max:

$$\tau(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$



Supervised Learning with Deep Neural Networks

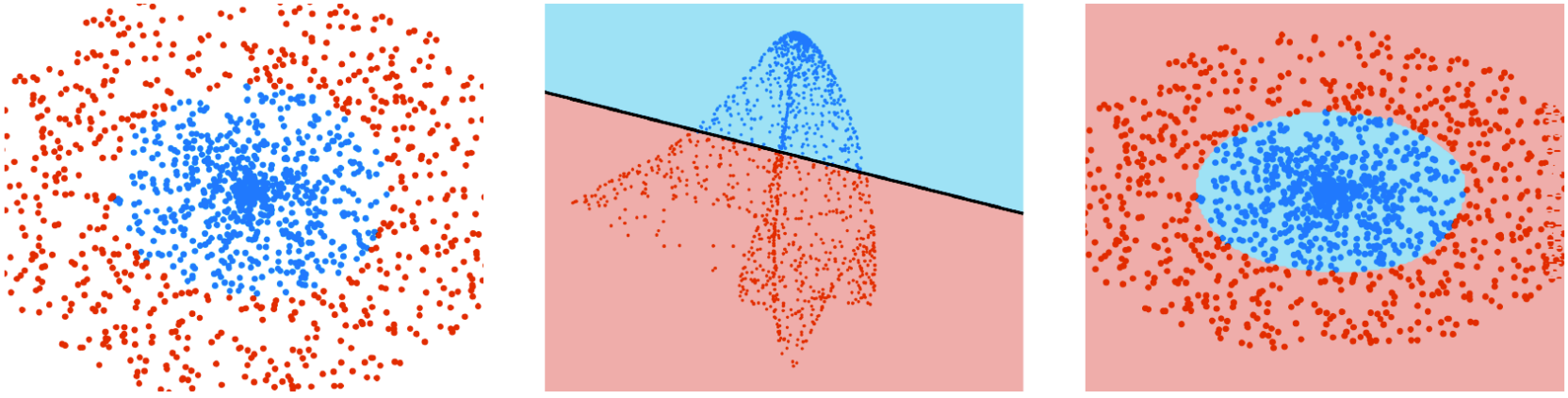


Figure courtesy of L. Ruthotto.

The Learning Problem

Given *training data* $(x_n, y_n)_{n=1,\dots,N}$ find $A = \{A^k\}_{k=1,\dots,K}$, $b = \{b^k\}_{k=1,\dots,K}$, W, ω that solve

$$\min_{A,b,W,\omega} \sum_{n=1}^{N} \|\tau(W x_n^K + \omega) - y_n\|^2 + R(A, b, W, \omega)$$

$$\text{s.t.} \quad x_n^{k+1} = \sigma(A^k x_n^k + b^k), \quad x_n^0 = x_n$$

- Quality of learning measured by *generalization*

Word of warning

Deep learning is great, but ...

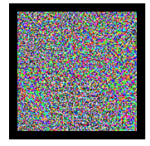
- need a lot of data
- difficult to train
- can be fooled
- ...

Deep learning needs ...

- mathematical guarantees
- explainable models
- ...

"panda"

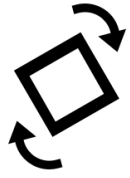
Adversarial Noise



"gibbon"

"vulture"

Adversarial Rotation



"orangutan"

"not hotdog"

Adversarial Photographer

"hotdog"

https://ai.googleblog.com/2018/09/introducing-unrestricted-adversarial.html

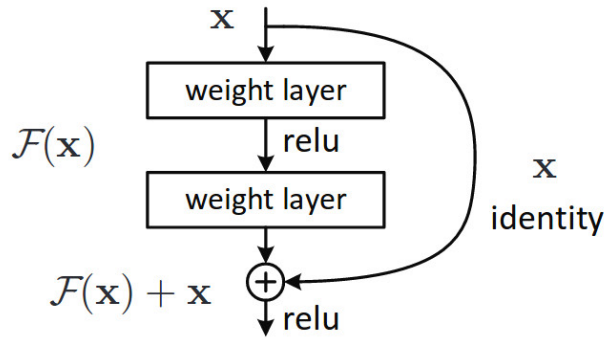
Deep Learning meets Optimal Control

Deep Residual Neural Networks (ResNet)

- "Standard" Neural Networks

$$x^{k+1} = \sigma(A^k x^k + b^k)$$

- Deep Residual Neural Networks (ResNet) He, Zhang, Ren, Sun 2015 ($>$ 68000 citations on GoogleScholar)



$$x^{k+1} = x^k + \Delta t\, \sigma(A^k x^k + b^k)$$

Linking ResNet with ODEs

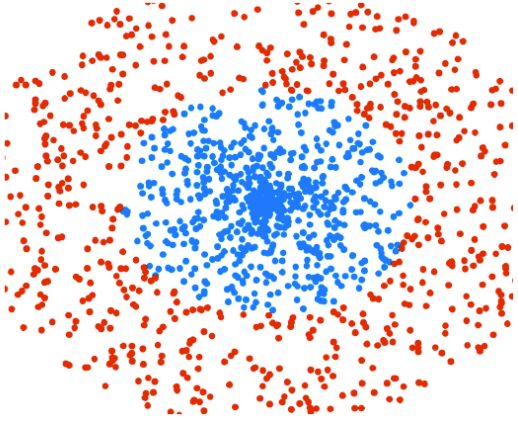
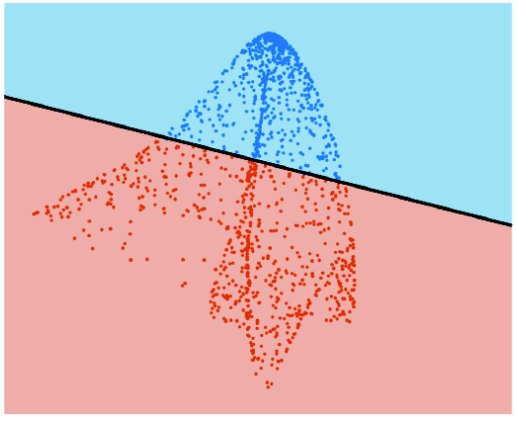
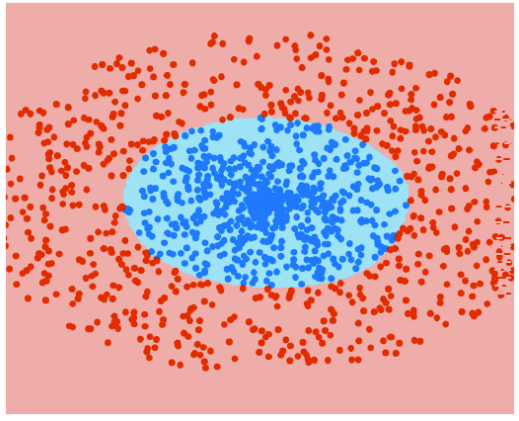
ResNet is Forward Euler discretization $\dot{x}(t) \approx \frac{x(t+\Delta t)-x(t)}{\Delta t}$ of

$$\dot{x}(t) = \sigma(A(t)x(t) + b(t)), \quad t \in [0, T]$$

with continuous-time mappings A, b. $x^k := x(k\Delta t)$ ...

Haber and Ruthotto 2018, Li et al. 2018, Benning et al. 2019, ...

Optimal Control meets for Deep Learning

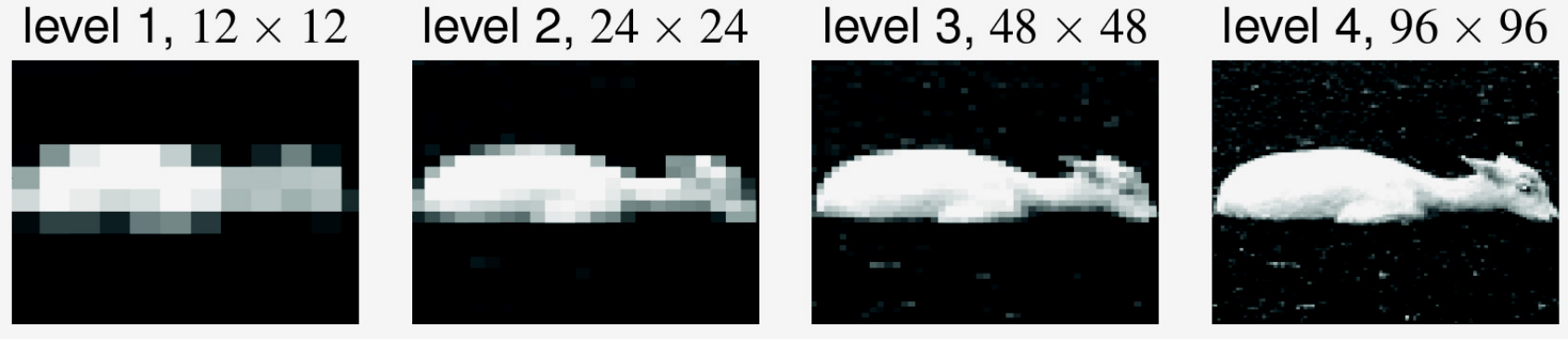
The Optimal Control Learning Problem

Given *training data* $(x_n, y_n)_{n=1,\dots,N}$ find $A : [0, T] \to \mathbb{R}^{M\times M}$, $b : [0, T] \to \mathbb{R}^M$, W, ω that solve

$$\min_{A,b,W,\omega} \sum_{n=1}^{N} \|\tau(W x_n(T) + \omega) - y_n\|^2 + R(A, b, W, \omega)$$

$$\text{s.t.} \quad \dot{x}_n = \sigma(A x_n + b), \quad x_n(0) = x_n$$

Potential advantages from Optimal Control / ODEs



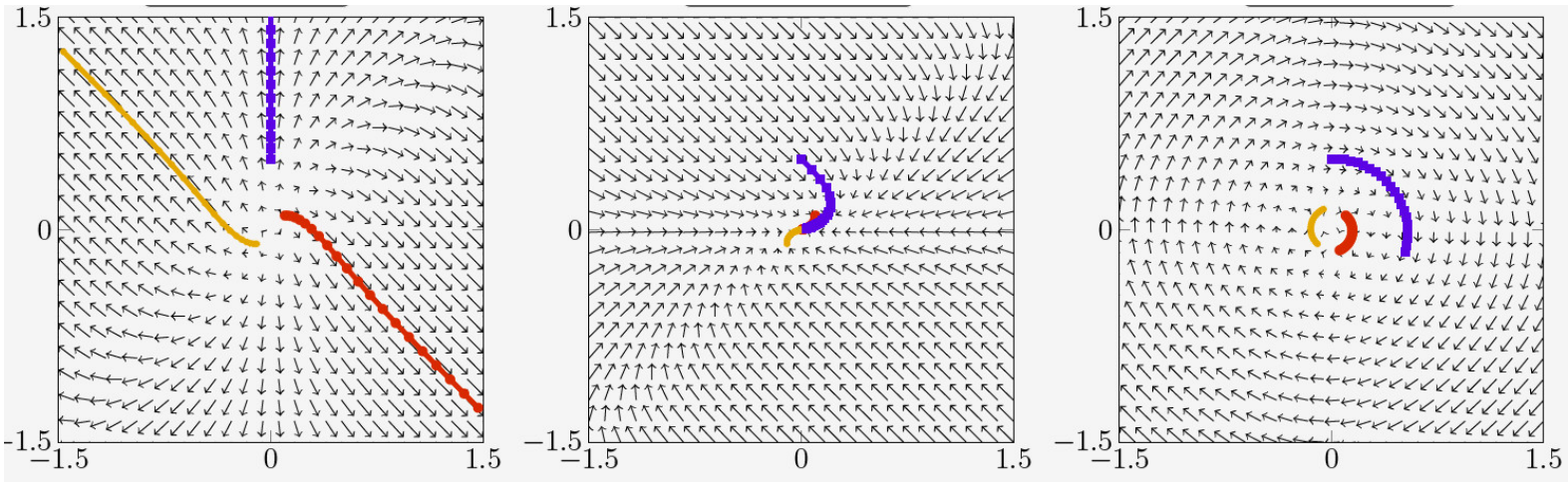
- Rich theory (see stability in a bit)
- New architectures
- New models, e.g. parametrization, regularization
- Advanced algorithms, e.g. Multi-resolution Learning
- ...

Utilize ODE knowledge

Stability and Generalization

- x_1 similar to x_2, then $x_1(T)$ should be similar to $x_2(T)$. Otherwise instability to small errors **prevents generalization**.
- Examples. $\sigma(y) = y, b = 0$

$$A_+ = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}, \quad A_- = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix}, \quad A_0 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$



Haber and Ruthotto 2018

Stability and Generalization

Theorem (very old)

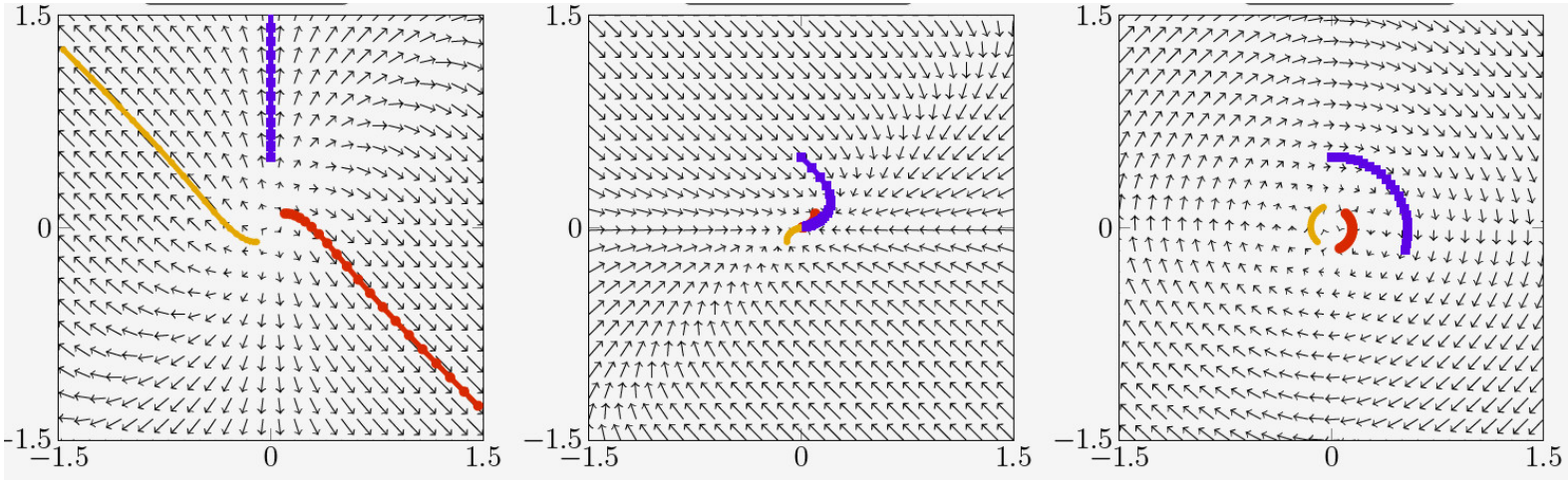
The **autonomous** *ODE* $\dot{x} = f(x)$ *is stable if the real parts of the eigenvalues of the Jacobian* Df *are non-positive.*

Corollary

Let $\dot{\sigma} \geq 0$. *Then forward propagation is* **stable** *if* $\mathrm{Re}(\lambda(A)) \leq 0$.

- Examples. $\sigma(y) = y, b = 0$

$$\lambda(A_+) = (2, 2), \quad \lambda(A_-) = (-2, -2), \quad \lambda(A_0) = (i, -i)$$

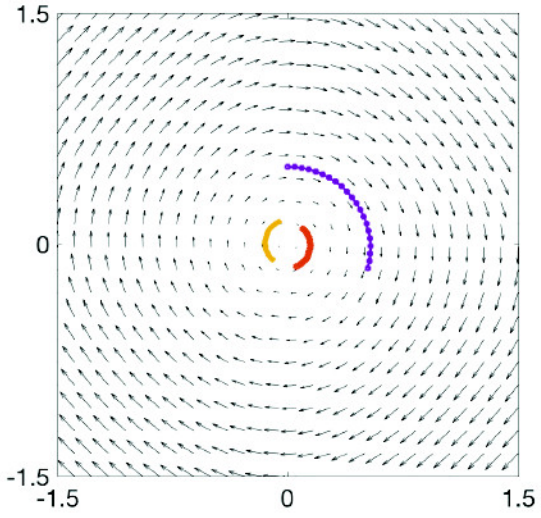


Haber and Ruthotto 2018

New **Unconditionally Stable** Architectures

- ResNet with **antisymmetric** transformation matrix

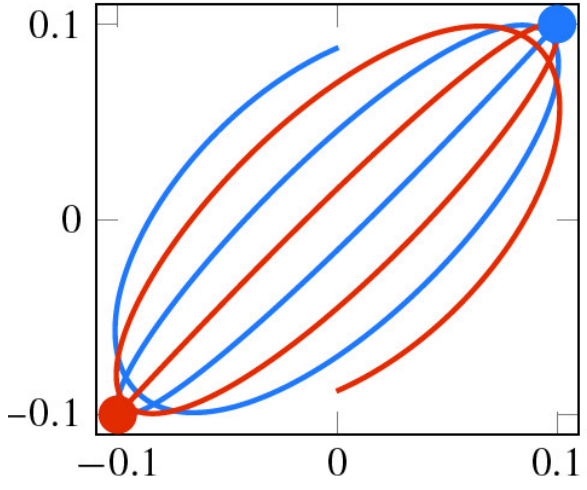
$$\dot{x} = \sigma\left((A - A^T)x + b\right)$$



- **Hamiltonian inspired Network**: ResNet with auxiliary variable and antisymmetric matrix

$$\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} = \sigma\left(\begin{pmatrix} 0 & A \\ -A^T & 0 \end{pmatrix}\begin{pmatrix} x \\ z \end{pmatrix} + b\right)$$

$$x(0) = x_0, \quad z(0) = 0$$

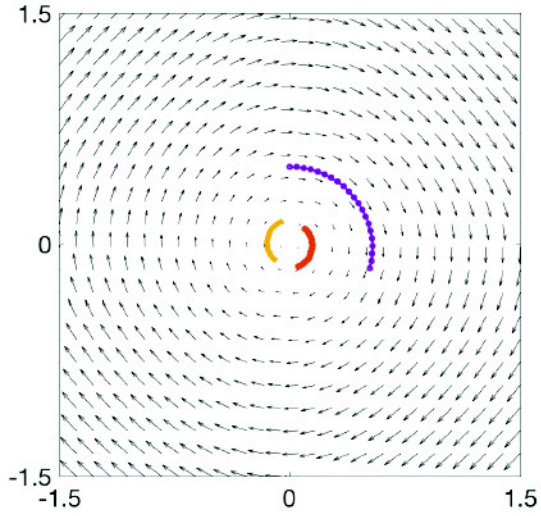


Haber and Ruthotto 2018

New **Unconditionally Stable** Architectures

- ResNet with **antisymmetric** transformation matrix

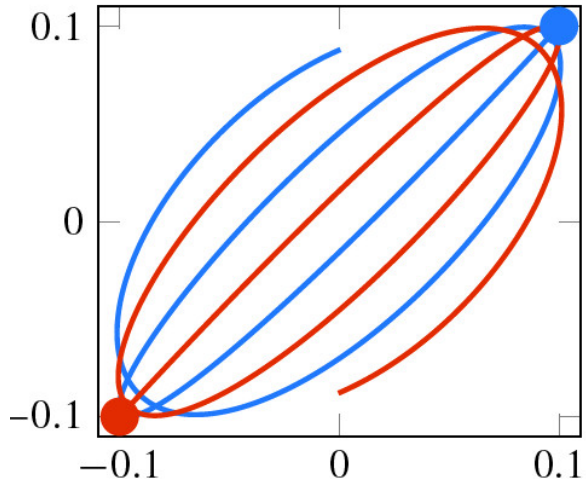
$$\dot{x} = \sigma\left((A - A^T)x + b\right)$$



- **Hamiltonian inspired Network**: ResNet with auxiliary variable and antisymmetric matrix

$$\begin{pmatrix}\dot{x}\\ \dot{z}\end{pmatrix} = \sigma\left(\begin{pmatrix}0 & A\\ -A^T & 0\end{pmatrix}\begin{pmatrix}x\\ z\end{pmatrix} + b\right)$$

$$x(0) = x_0, \quad z(0) = 0$$



Haber and Ruthotto 2018

Problem: this statement is **only true** for autonomous systems!

If the vector-field f depends on time, then similar statements are true but the theory is **rather weak**.

Revisiting Stability

Consider $\Phi_\theta(u) = u(T)$ with u solving $\dot{u}(t) = f(t, u(t)), t \in [0, T]$

What is a useful definition of **stability** for neural networks?

Definition (Stability 1: Lipschitz)

There exists $C > 0$ such that for all u, v we have

$$\|\Phi_\theta(u) - \Phi_\theta(v)\| \leq C\|u - v\| \tag{Lip}$$

Definition (Stability 2: Non-expansive)

For all u, v (Lip) holds with $C = 1$.

When is Φ_θ stable?

Recall $\Phi_\theta(u) = u(T)$ with u solving $\dot{u}(t) = f(t, u(t)), t \in [0, T]$

- Arguments based on **Lipschitz continuity** of f
  - If $f(t, \cdot)$ being L-Lipschitz, e.g. $f(t, u) = \sigma(A(t)u + b(t))$ with σ being S-Lipschitz and A continuous, then Def 1 holds with

$$C = \exp(T \cdot L) \quad (= \exp(T \cdot S \max_t \|A(t)\|)).$$

Celledoni et al. 2020, Zhang and Schaeffer 2020

When is Φ_θ stable?

Recall $\Phi_\theta(u) = u(T)$ with u solving $\dot{u}(t) = f(t, u(t)), t \in [0, T]$

- Arguments based on **Lipschitz continuity** of f
  - If $f(t, \cdot)$ being L-Lipschitz, e.g. $f(t, u) = \sigma(A(t)u + b(t))$ with σ being S-Lipschitz and A continuous, then Def 1 holds with

$$C = \exp(T \cdot L) \quad (= \exp(T \cdot S \max_t \|A(t)\|)).$$

Can't satisfy Def 2 since $C > 1$ for all non-trivial cases.

Celledoni et al. 2020, Zhang and Schaeffer 2020

When is Φ_θ stable?

Recall $\Phi_\theta(u) = u(T)$ with u solving $\dot{u}(t) = f(t, u(t)), t \in [0, T]$

- Arguments based on **Lipschitz continuity** of f
  - If $f(t, \cdot)$ being L-Lipschitz, e.g. $f(t, u) = \sigma(A(t)u + b(t))$ with σ being S-Lipschitz and A continuous, then Def 1 holds with

$$C = \exp(T \cdot L) \quad (= \exp(T \cdot S \max_t \|A(t)\|)).$$

Can't satisfy Def 2 since $C > 1$ for all non-trivial cases.

- Arguments based on **"one-sided" Lipschitz continuity** of f

$$\langle f(t, u_1) - f(t, u_2), u_1 - u_2 \rangle \leq \nu \|u_1 - u_2\|^2$$

  - If f is L-Lipschitz, then f is "one-sided" Lipschitz with $\nu = L$

$$\begin{aligned} \langle f(t, u_1) - f(t, u_2), u_1 - u_2 \rangle &\leq \|f(t, u_1) - f(t, u_2)\| \|u_1 - u_2\| \\ &\leq L \|u_1 - u_2\|^2 \end{aligned}$$

  - Then Ψ_θ is Lipschitz with $C = \exp(T \cdot \nu)$.

Celledoni et al. 2020, Zhang and Schaeffer 2020

When is Φ_θ stable?

Recall $\Phi_\theta(u) = u(T)$ with u solving $\dot{u}(t) = f(t, u(t)), t \in [0, T]$

- Arguments based on **Lipschitz continuity** of f
  - If $f(t, \cdot)$ being L-Lipschitz, e.g. $f(t, u) = \sigma(A(t)u + b(t))$ with σ being S-Lipschitz and A continuous, then Def 1 holds with

$$C = \exp(T \cdot L) \quad (= \exp(T \cdot S \max_t \|A(t)\|)).$$

Can't satisfy Def 2 since $C > 1$ for all non-trivial cases.

- Arguments based on **"one-sided" Lipschitz continuity** of f

$$\langle f(t, u_1) - f(t, u_2), u_1 - u_2 \rangle \leq \nu \|u_1 - u_2\|^2$$

  - If f is L-Lipschitz, then f is "one-sided" Lipschitz with $\nu = L$

$$\begin{aligned} \langle f(t, u_1) - f(t, u_2), u_1 - u_2 \rangle &\leq \|f(t, u_1) - f(t, u_2)\| \|u_1 - u_2\| \\ &\leq L \|u_1 - u_2\|^2 \end{aligned}$$

  - Then Ψ_θ is Lipschitz with $C = \exp(T \cdot \nu)$.
    Catch: ν can be non-positive, $\nu \leq 0$. Thus may satisfy Def 2

Celledoni et al. 2020, Zhang and Schaeffer 2020

Sufficient Conditions for Stability

Recall, $\Phi_\theta(u) = u(T)$ with u solving $\dot{u}(t) = f(t, u(t)), t \in [0, T]$

"one-sided" Lipschitz continuity of f

$$\langle f(t, u_1) - f(t, u_2), u_1 - u_2 \rangle \leq \nu \|u_1 - u_2\|^2 \tag{1}$$

Theorem

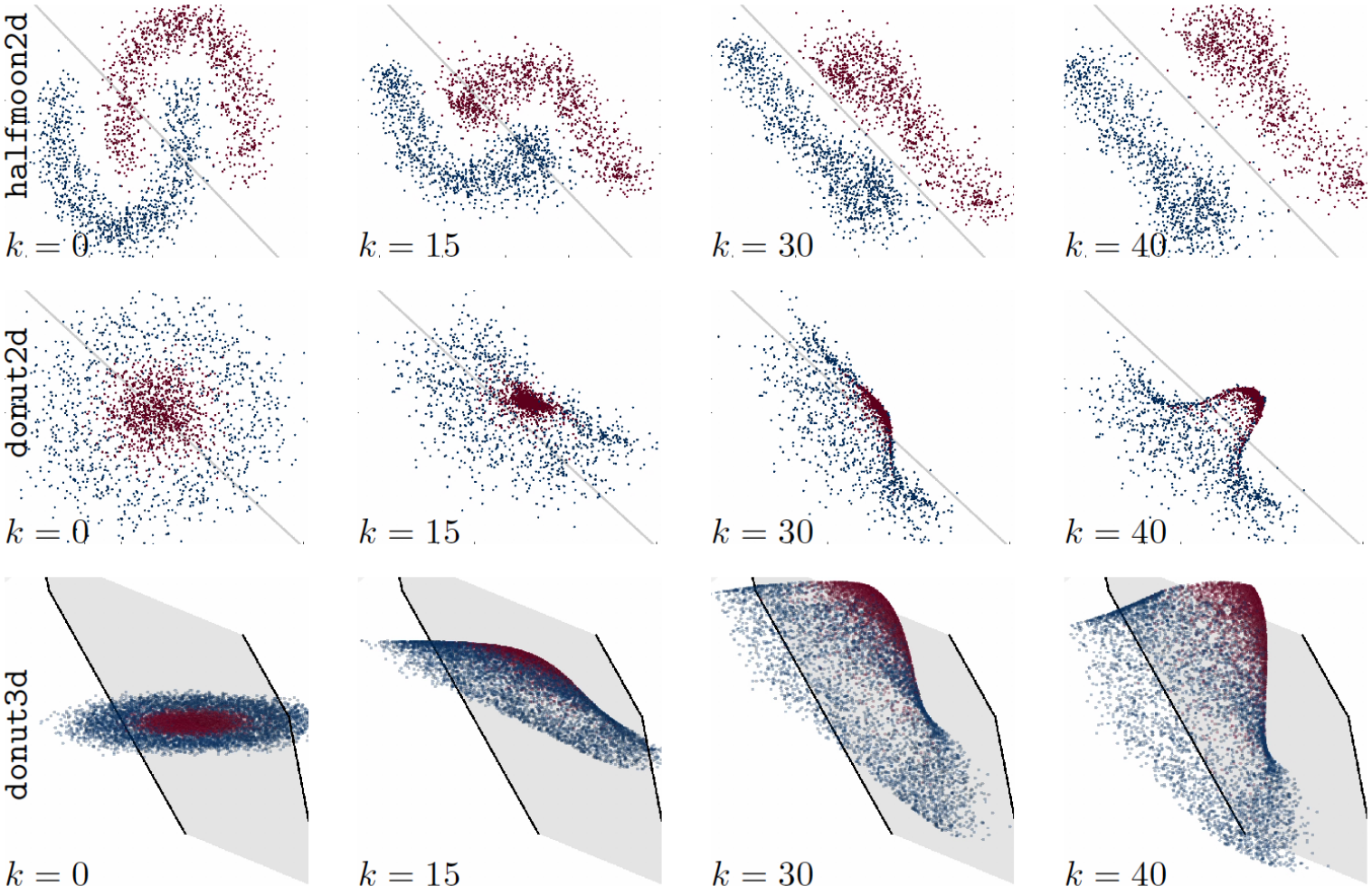
- *Let $V_t(u)$ be twice differentiable and convex. Then $f(t, u) = -\nabla V_t(u)$ satisfies the one-sided Lipschitz condition for some $\nu \leq 0$.*
- *Let $0 \leq \sigma' \leq 1$ almost everywhere. Then*

$$f(t, u) = -A^*(t)\sigma(A(t)u + b(t))$$

satisfies the one-sided Lipschitz condition with $-\mu_*^2 \leq \nu \leq 0$ where $\mu_* := \inf_t \mu(t)$ and $\mu(t)$ is the smallest singular value of $A(t)$.

Celledoni et al. 2020, Zhang and Schaeffer 2020

Examples: Forward propagation with ResNet



videos?

Celledoni et al. 2020

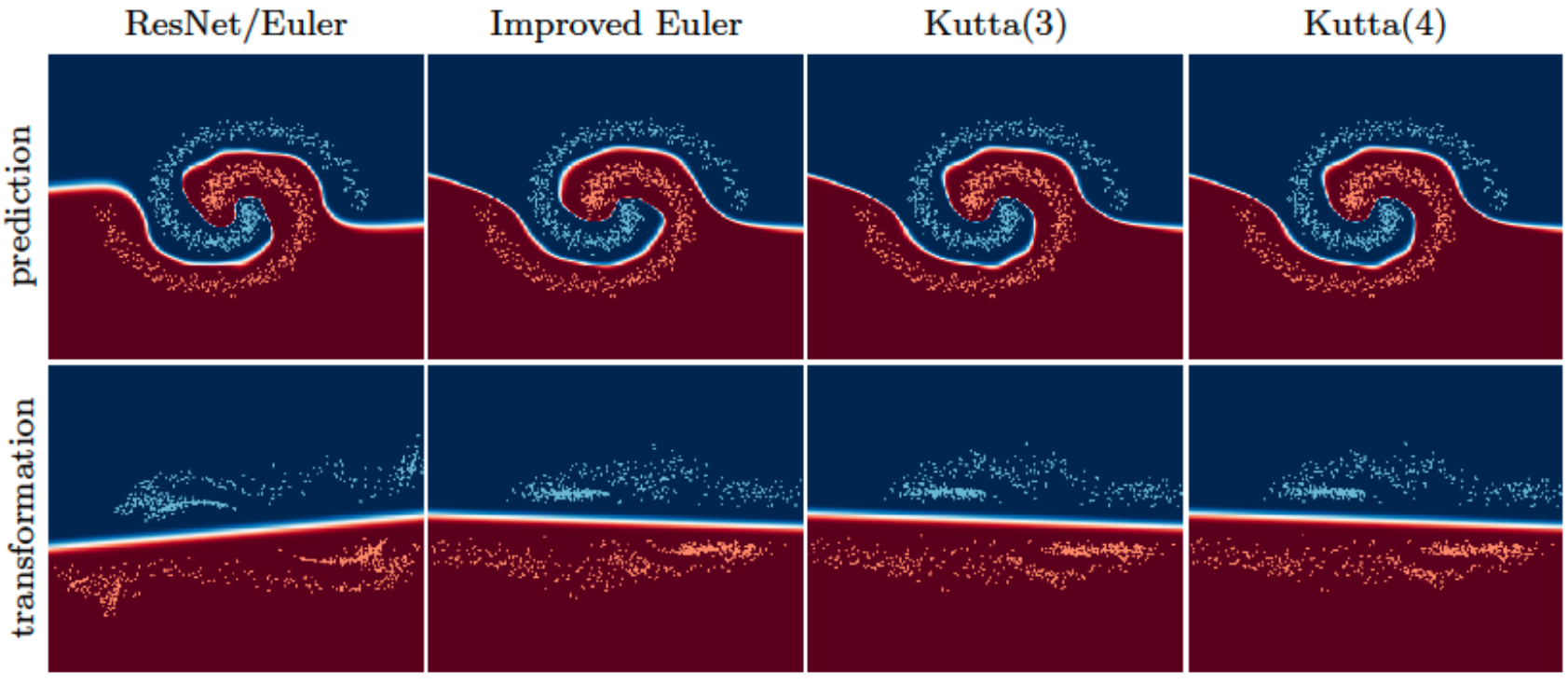
Examples: Different Runge-Kutta methods

$$\begin{array}{c|c} 0 & \\ 1 & 1 \\ \hline & \frac{1}{2} \quad \frac{1}{2} \end{array}$$

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ 1 & -1 & 2 \\ \hline & \frac{1}{6} & \frac{2}{3} \quad \frac{1}{6} \end{array}$$

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & 0 & \frac{1}{2} & \\ 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} \quad \frac{1}{6} \end{array}$$

TABLE 1. Four explicit Runge–Kutta methods: ResNet/Euler, Improved Euler, Kutta(3) and Kutta(4).



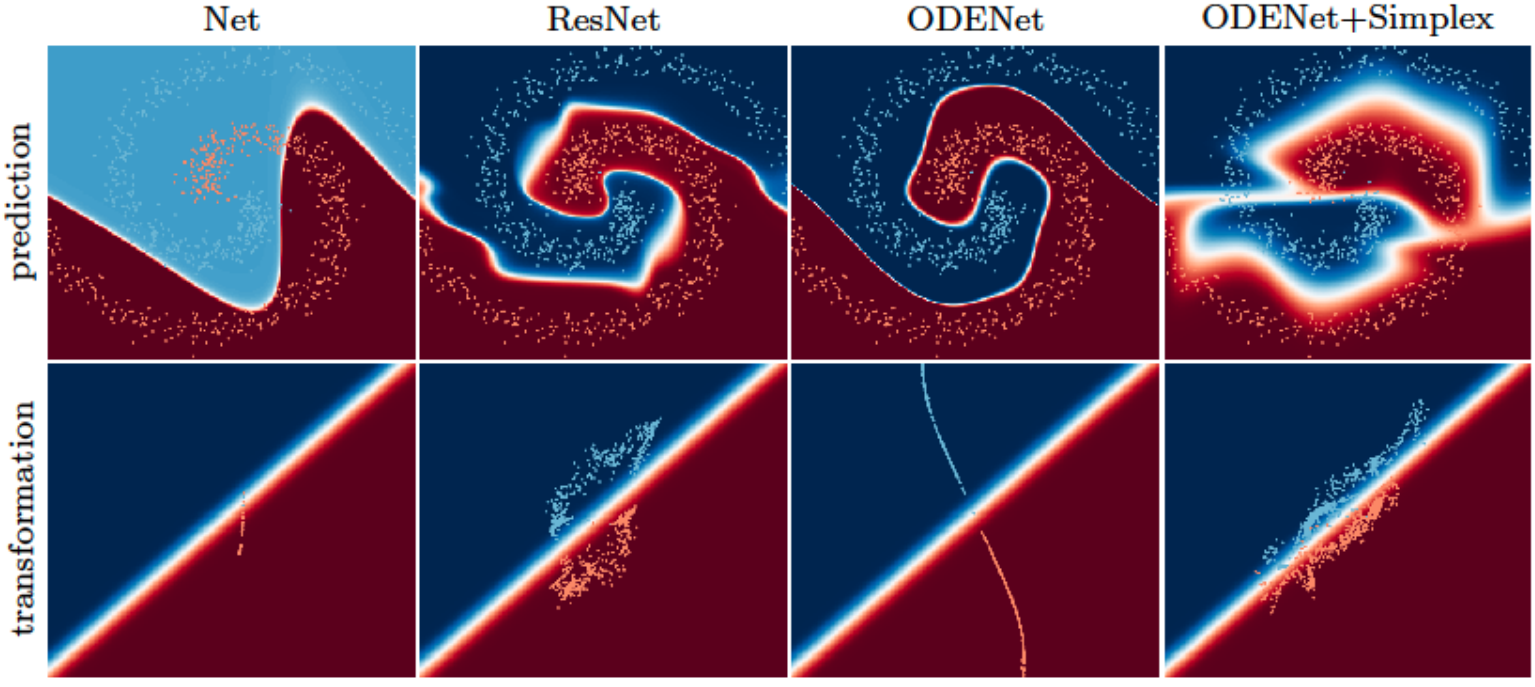
Benning et al. 2019

Examples: Learn time steps

$x^{k+1} = x^k + \Delta t^k \sigma(A^k x^k + b^k)$

ODENet: Estimate $(\Delta t^k, A^k, b^k)$

Simplex constraint: $\Delta t^k \geq 0, \sum_k \Delta t^k = T$



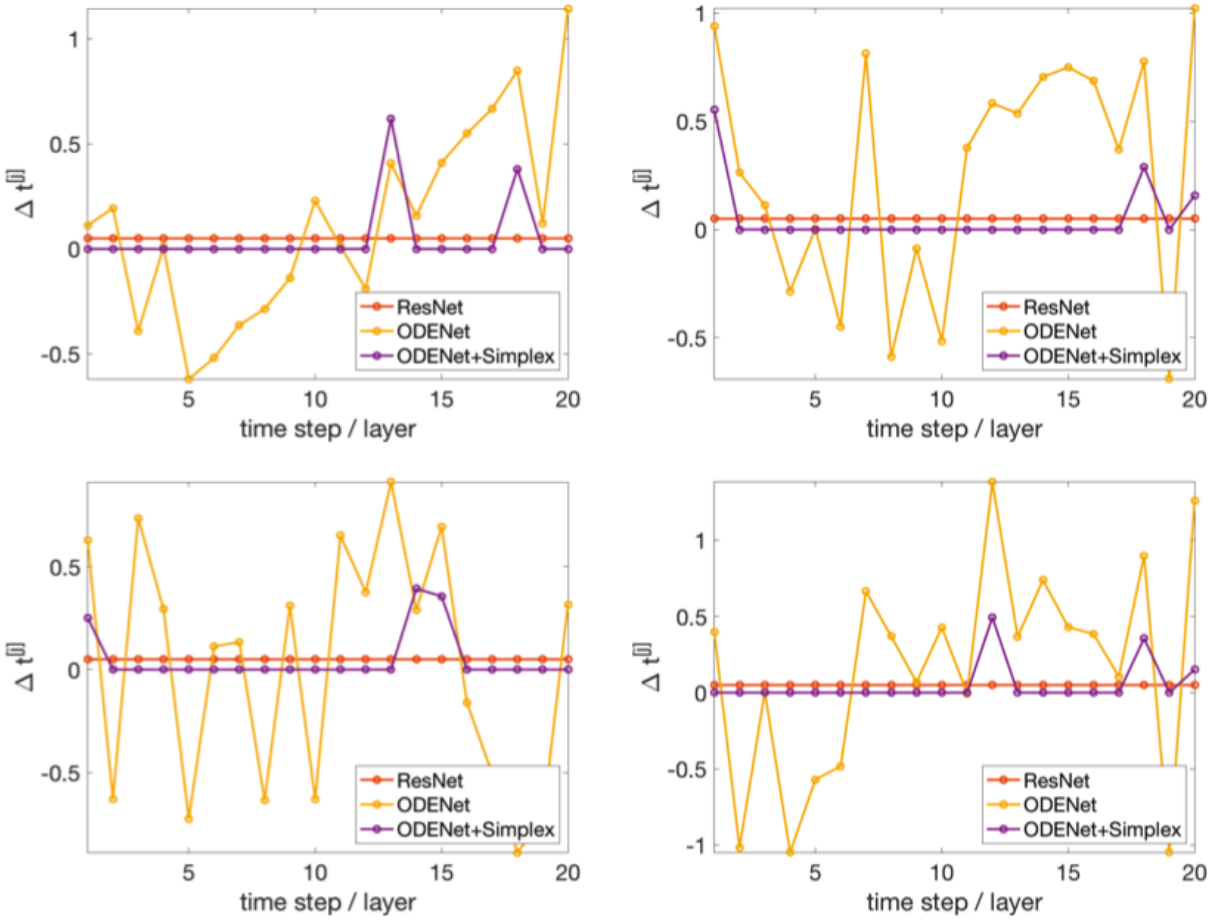
Benning et al. 2019

Examples: Learn time steps

$x^{k+1} = x^k + \Delta t^k \sigma(A^k x^k + b^k)$

ODENet: Estimate $(\Delta t^k, A^k, b^k)$

Simplex constraint: $\Delta t^k \geq 0, \sum_k \Delta t^k = T$



Benning et al. 2019

Deep Limits: Number of Layers $K \to \infty$ Thorpe and van Gennip 2018

Interpret discrete parameters as piecewise constant functions on $[0, T]$

Discrete Learning Problem

Find $\theta^K := (A^k, b^k)_{k=1,\dots,K}$ which minimize $E^K : L^2(\mu^K) \to \mathbb{R}$, $E^K(\theta) = \sum_{k=1}^K \|x_n^K - y_n\|^2 + R^K(\theta)$ such that

$$x_n^{k+1} = x_n^k + \frac{T}{K}\sigma(A^k x_n^k + b^k), \quad x_n^0 = x_n$$

Continuous Learning Problem

Find $\theta^\infty := (A : [0, T] \to \mathbb{R}^{M\times M}, b : [0, T] \to \mathbb{R}^M)$ which minimize $E^\infty : H^1 \to \mathbb{R}$, $E(\theta) = \sum_{k=1}^K \|x_n(T) - y_n\|^2 + R(\theta)$ such that

$$\dot{x}_n = \sigma(A x_n + b), \quad x_n(0) = x_n$$

Deep Limits: Number of Layers $K \to \infty$ Thorpe and van Gennip 2018

Interpret discrete parameters as piecewise constant functions on $[0, T]$

Discrete Learning Problem

Find $\theta^K := (A^k, b^k)_{k=1,\dots,K}$ which minimize $E^K : L^2(\mu^K) \to \mathbb{R}$, $E^K(\theta) = \sum_{k=1}^K \|x_n^K - y_n\|^2 + R^K(\theta)$ such that

$$x_n^{k+1} = x_n^k + \frac{T}{K}\sigma(A^k x_n^k + b^k), \quad x_n^0 = x_n$$

Continuous Learning Problem

Find $\theta^\infty := (A : [0, T] \to \mathbb{R}^{M\times M}, b : [0, T] \to \mathbb{R}^M)$ which minimize $E^\infty : H^1 \to \mathbb{R}$, $E(\theta) = \sum_{k=1}^K \|x_n(T) - y_n\|^2 + R(\theta)$ such that

$$\dot{x}_n = \sigma(Ax_n + b), \quad x_n(0) = x_n$$

Theorem

R^K, R suitable regularization and σ Lipschitz with $\sigma(0) = 0$. Then 1) minimizers exist, 2) minimal values converge and 3) $\{\theta^K\}_{K\in\mathbb{N}} \subset L^2$ compact and **any limit point is a minimizer** *of E.*

Enforcing Structure in Neural Networks

Invertible Neural Networks

Consider a neural network $\Psi : X \to X, \Psi(x) = (f_K \circ ... \circ f_1)(x)$

If f_i are **invertible**, then so is Ψ and $\Psi^{-1}(x) = (f_1^{-1} \circ ... \circ f_K^{-1})(x)$

Invertible Neural Networks

Consider a neural network $\Psi : X \to X, \Psi(x) = (f_K \circ ... \circ f_1)(x)$
If f_i are **invertible**, then so is Ψ and $\Psi^{-1}(x) = (f_1^{-1} \circ ... \circ f_K^{-1})(x)$

What are invertible networks **useful** for?

- **memory efficient backpropagation** (computing derivatives): can be implemented as $\mathcal{O}(1)$ rather than $\mathcal{O}(K)$

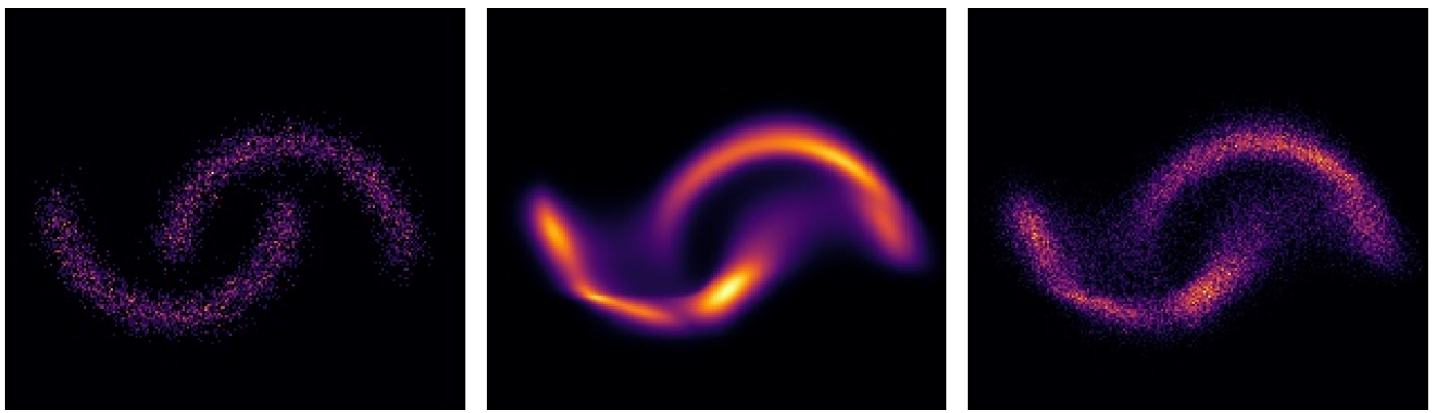
Invertible Neural Networks

Consider a neural network $\Psi : X \to X, \Psi(x) = (f_K \circ ... \circ f_1)(x)$
If f_i are **invertible**, then so is Ψ and $\Psi^{-1}(x) = (f_1^{-1} \circ ... \circ f_K^{-1})(x)$

What are invertible networks **useful** for?

- **memory efficient backpropagation** (computing derivatives): can be implemented as $\mathcal{O}(1)$ rather than $\mathcal{O}(K)$
- **generative modeling**: Usually learn Ψ_θ such that $x \sim (\Psi_\theta)_* \mu$ with PDF of μ being q. Easy to sample from $(\Psi_\theta)_* \mu$ but PDF unknown.
  If Ψ_θ is invertible, then $(\Psi_\theta)_* \mu$ has PDF

$$x \mapsto q(\Psi_\theta^{-1}(x)) |\det J\Psi_\theta^{-1}(x)|$$



Chen et al. 2019

Invertible Neural Networks

Types of invertible layers:

1) **Linear invertible layers**: LU factorization Kondor et al. 2018, pixel shuffle Dinh et al. 2014

Invertible Neural Networks

Types of invertible layers:

1) **Linear invertible layers**: LU factorization Kondor et al. 2018, pixel shuffle Dinh et al. 2014
2) **Coupling layers** Dinh et al. 2014, $x = (x_1, x_2)$

$$f(x_1, x_2) = (x_1, g_{h(x_1)}(x_2))$$

If g_a is invertible for any a , so is f with inverse

$$f^{-1}(x_1, x_2) = (x_1, g^{-1}_{h(x_1)}(x_2))$$

Simplest example: "additive coupling layer" $\gamma_a(b) = b + a$ has inverse $\gamma_a^{-1}(b) = b - a$.
See Din et al. 2016 and Durkan et al. 2019 for more examples.

Invertible Neural Networks

Types of invertible layers:

1) **Linear invertible layers**: LU factorization Kondor et al. 2018, pixel shuffle Dinh et al. 2014
2) **Coupling layers** Dinh et al. 2014, $x = (x_1, x_2)$

$$f(x_1, x_2) = (x_1, g_{h(x_1)}(x_2))$$

If g_a is invertible for any a , so is f with inverse

$$f^{-1}(x_1, x_2) = (x_1, g^{-1}_{h(x_1)}(x_2))$$

Simplest example: "additive coupling layer" $\gamma_a(b) = b + a$ has inverse $\gamma_a^{-1}(b) = b - a$.
See Din et al. 2016 and Durkan et al. 2019 for more examples.

3) **Residual layers**: $f(x) = x + g(x)$
If g is Lipschitz with constant $L < 1$, then f is invertible.
Problem: $f^{-1}(x)$ is not easy to compute, e.g. use fixed point iteration, and $L < 1$ hard to satisfy. See Behrmann et al. 2019 for more details.

Invertible Neural Networks

Types of invertible layers:

1) **Linear invertible layers**: LU factorization Kondor et al. 2018, pixel shuffle Dinh et al. 2014
2) **Coupling layers** Dinh et al. 2014, $x = (x_1, x_2)$

$$f(x_1, x_2) = (x_1, g_{h(x_1)}(x_2))$$

If g_a is invertible for any a , so is f with inverse

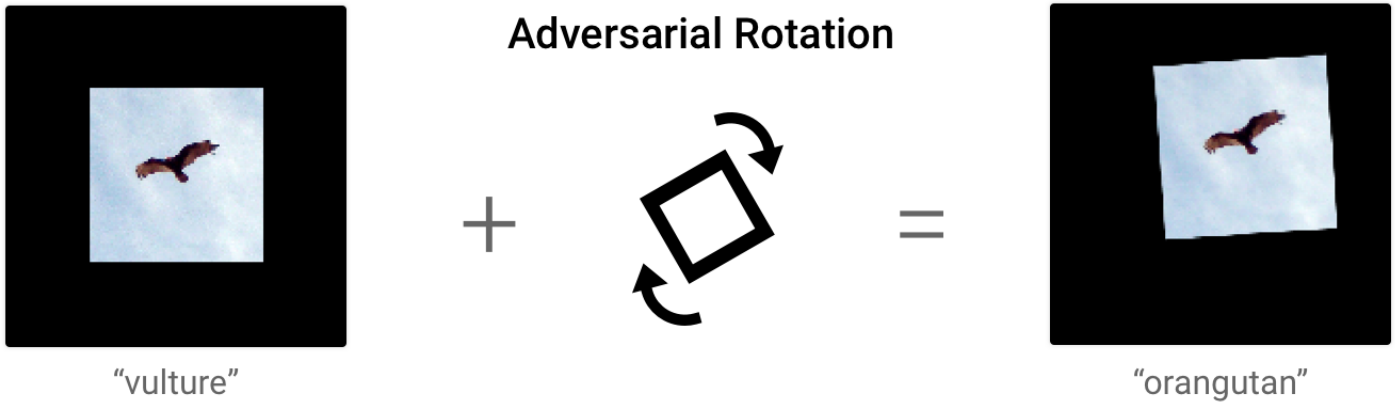
$$f^{-1}(x_1, x_2) = (x_1, g^{-1}_{h(x_1)}(x_2))$$

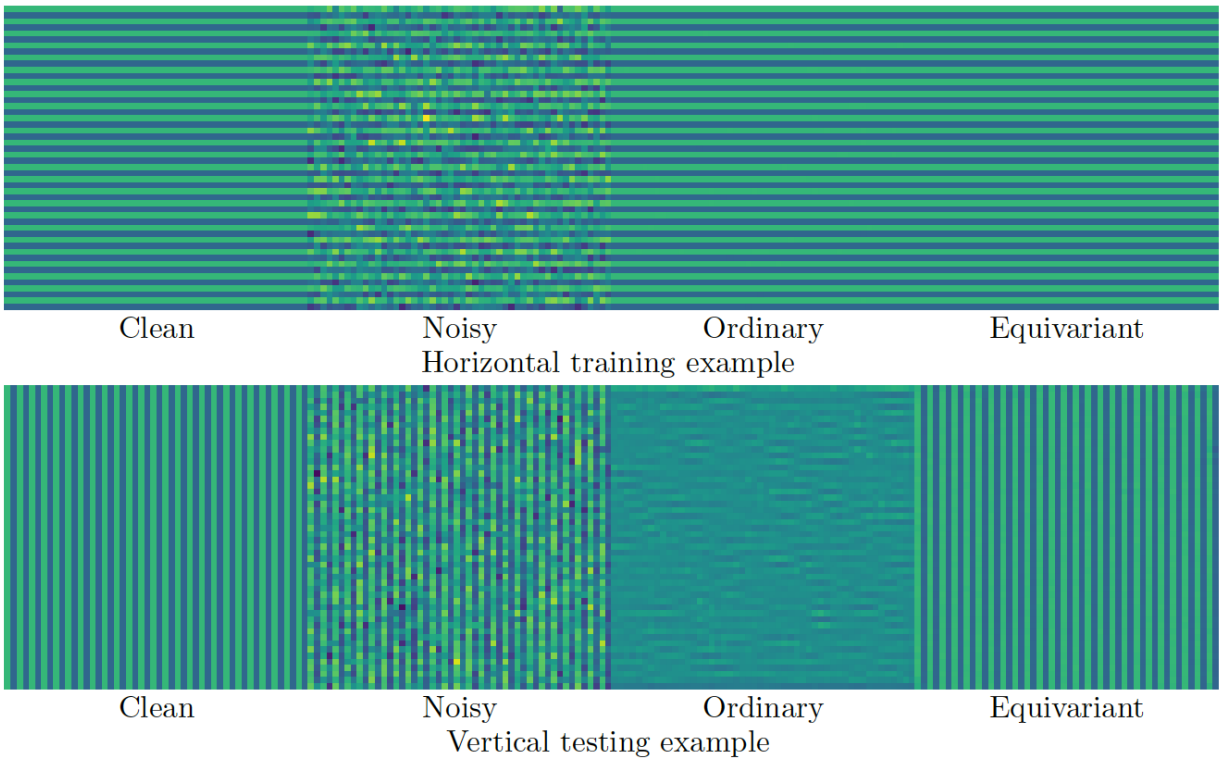
Simplest example: "additive coupling layer" $\gamma_a(b) = b + a$ has inverse $\gamma_a^{-1}(b) = b - a$.
See Din et al. 2016 and Durkan et al. 2019 for more examples.

3) **Residual layers**: $f(x) = x + g(x)$
If g is Lipschitz with constant $L < 1$, then f is invertible.
Problem: $f^{-1}(x)$ is not easy to compute, e.g. use fixed point iteration, and $L < 1$ hard to satisfy. See Behrmann et al. 2019 for more details.

Important: inverse must be "easy" to compute.

Equivariance and invariance





Sherry et al. 2021

Equivariance and invariance Cohen and Welling 2016, Cohen et al. 2019

Definition (Group equivariance and invariance)

Group G "acts" on spaces X and Y. For any $g \in G$ denote "action" g on $x \in X$ and $y \in Y$ by T_g^X and T_g^Y, respectively. We call a function $\Psi : X \to Y$ G-equivariant if for all $g \in G$ we have

$$\Psi \circ T_g^X = T_g^Y \circ \Psi$$

If Ψ is G-equivariant and G acts trivially on Y, then we call Ψ G-invariant. I.e. for all $x \in X$ and $g \in G$ $\Psi(T_g^X x) = \Psi(x)$.

Equivariance and invariance Cohen and Welling 2016, Cohen et al. 2019

Definition (Group equivariance and invariance)

Group G "acts" on spaces X and Y. For any $g \in G$ denote "action" g on $x \in X$ and $y \in Y$ by T_g^X and T_g^Y, respectively. We call a function $\Psi : X \to Y$ G-equivariant if for all $g \in G$ we have

$$\Psi \circ T_g^X = T_g^Y \circ \Psi$$

If Ψ is G-equivariant and G acts trivially on Y, then we call Ψ G-invariant. I.e. for all $x \in X$ and $g \in G$ $\Psi(T_g^X x) = \Psi(x)$.

Key property: If $\Psi_1 : X \to Y$ and $\Psi_2 : Y \to Z$ are G-equivariant (with the same action on Y), then $\Psi_2 \circ \Psi_1$ is G-equivariant, too!

Equivariance and invariance Cohen and Welling 2016, Cohen et al. 2019

Definition (Group equivariance and invariance)

Group G "acts" on spaces X and Y. For any $g \in G$ denote "action" g on $x \in X$ and $y \in Y$ by T_g^X and T_g^Y, respectively. We call a function $\Psi : X \to Y$ G-equivariant if for all $g \in G$ we have

$$\Psi \circ T_g^X = T_g^Y \circ \Psi$$

If Ψ is G-equivariant and G acts trivially on Y, then we call Ψ G-invariant. I.e. for all $x \in X$ and $g \in G$ $\Psi(T_g^X x) = \Psi(x)$.

Key property: If $\Psi_1 : X \to Y$ and $\Psi_2 : Y \to Z$ are G-equivariant (with the same action on Y), then $\Psi_2 \circ \Psi_1$ is G-equivariant, too!

Examples of interesting groups:

- translations
- rotations
- scaling

Application to inverse problems: Sherry et al. 2021 coming out soon!

Equivariance and invariance

Advantages of equivariance for neural networks:

- no need for data augmentation
- fewer parameters
- trains faster
- mathematical guarantees
- ...

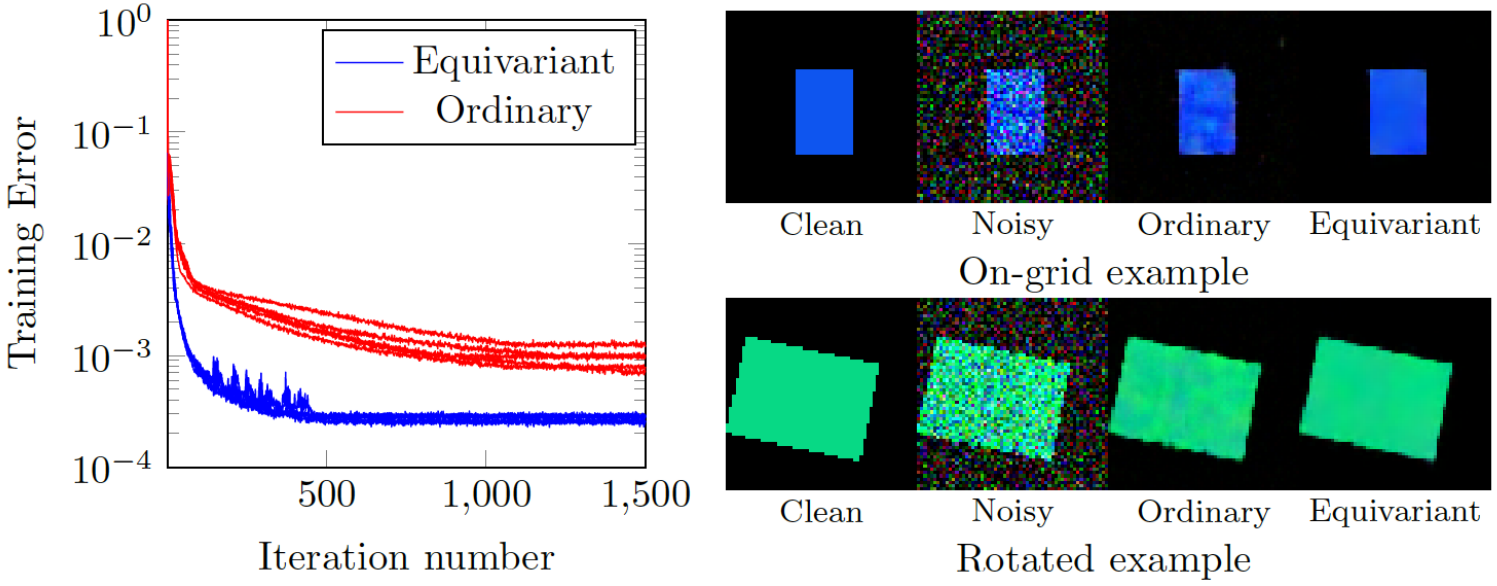


Figure: Celledoni et al. 2020

Cohen and Welling 2016, Cohen et al. 2019, Worall et al. 2017 ...

Conclusions

- **Connections** of deep learning to ODEs, optimal control, group theory ...
- **New architectures with mathematical guarantees**: stable, invertible, equivariant ...
- Direct benefit for applictions: **faster to train**, **less data**, ...

Open problems

E. Celledoni et al., "Structure preserving deep learning," arxiv:2006.03364, 2020.

- **discretization**
- **manifolds**
- **sampling complexity**
- ...