Dynamic System and Deep Learning[1]

Shihua Zhang

December 8, 2021

---

[1] Qianxiao Li. *Dynamical Systems and Machine Learning*. Summer School, Peking University, 2020.

Outline

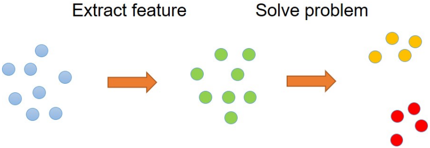
1. Introduction
2. Optimal control and deep learning
3. Control inspired learning algorithms
4. Control inspired architectures

Machine learning and deep learning

- Task: explore the information in data

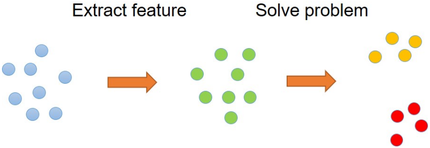
Machine learning and deep learning

- Task: explore the information in data
- A general scheme of machine learning: extract features from data and solve problem with features

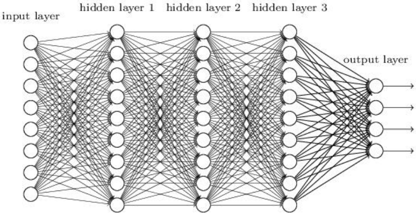


Machine learning and deep learning

- Task: explore the information in data
- A general scheme of machine learning: extract features from data and solve problem with features



- Deep learning also follow the scheme, however in a deep way



Is deep learning explainable?

- Traditional machine learning models are explainable
    - meaningful feature
    - less parameter
    - designable achitecture

Is deep learning explainable?

- Traditional machine learning models are explainable
    - meaningful feature
    - less parameter
    - designable achitecture
- Deep learning is hard to explain
    - meaningless feature
    - huge number of parameter
    - achitecture hard to design

Is deep learning explainable?

- Traditional machine learning models are explainable
  - meaningful feature
  - less parameter
  - designable achitecture
- Deep learning is hard to explain
  - meaningless feature
  - huge number of parameter
  - achitecture hard to design

We need an explainable model to model deep learning models!

ResNet is a typical deep learning model

- ResNet[2] is a milestone in deep learing

- A simple change on each block (layer) of DNN

$$x_{k+1} = v(x_k, \theta_k) \to x_{k+1} = x_k + v(x_k, \theta_k)$$

- ResNet can be really deep (more than 1000 layers) and is widely used in practice as a base model

---

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

The intuition to use dynamic system

The “model” we use is dynamic system

- Consider an ODE

$$\frac{dx}{dt} = F(x, t) \tag{1}$$

- Forward Euler method:

$$x_{t+h} = x_t + hF(x_t, t) \tag{2}$$

---

[3] E Weinan. “A proposal on machine learning via dynamical systems”. In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11.

The intuition to use dynamic system

The “model” we use is dynamic system

- Consider an ODE

$$\frac{dx}{dt} = F(x, t) \tag{1}$$

- Forward Euler method:

$$x_{t+h} = x_t + hF(x_t, t) \tag{2}$$

- Analogous to ResNet block

$$x_{k+1} = x_k + v(x_k, \theta_k) \tag{3}$$

- ResNet can be regarded as a numerical solution of an ODE by disretizing time[3]

---

[3] E Weinan. “A proposal on machine learning via dynamical systems”. In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11.

Connection to dynamic system

What can we do based on the connection of ResNet and dynamic system?

- Formulize deep learning to optimal control problem

- Inspire new learning algorithms

- Inspire new architectures

Outline

1. Introduction

2. Optimal control and deep learning

3. Control inspired learning algorithms

4. Control inspired architectures

The optimal control

Definition

Optimal control theory is a branch of mathematical optimization that deals with finding a control for a dynamical system over a period of time such that an objective function is optimized

- The dynamics

$$\dot{x}(t) = f(x(t), \theta(t)), \quad t \in [t_0, t_1], \quad x(t_0) = x_0$$

The optimal control

Definition

Optimal control theory is a branch of mathematical optimization that deals with finding a control for a dynamical system over a period of time such that an objective function is optimized

- The dynamics

$$\dot{x}(t) = f(x(t), \theta(t)), \quad t \in [t_0, t_1], \quad x(t_0) = x_0$$

- The cost function

$$J[\theta] = \int_{t_0}^{t_1} L(t, x(t), \theta(t))dt + \Phi\left(t_1, x\left(t_1\right)\right)$$

- L is called running cost and Φ is called terminal cost

Optimal control formulation

- The formulation of the optimal control problem is

$$\begin{aligned} &\inf_{\theta} J[\theta]=\int_{t_0}^{t_1} L(t, x(t), \theta(t)) dt+\Phi\left(t_1, x\left(t_1\right)\right) \\ &\text{subject to} \\ &\dot{x}(t)=f(t, x(t), \theta(t)), \quad t \in\left[t_0, t_1\right], \quad x\left(t_0\right)=x_0 \end{aligned} \tag{4}$$

- Based on the connection between dynamic system and deep learining, we can study the optimization of deep learning from the optimal control view

A mean-field optimal control formulation of deep learning

- Consider supervised learning with ResNet with K blocks (layers)
  - (x, y) is sampled from distribution μ
  - Let $\Phi(x_K, y)$ denote the loss function on data x and its label y
  - Let $L(x_k, \theta_k)$ denotes the regularizer of the k-th block (e.g., L_2 regularizer)

A mean-field optimal control formulation of deep learning

- Consider supervised learning with ResNet with K blocks (layers)
  - (x, y) is sampled from distribution μ
  - Let $\Phi(x_K, y)$ denote the loss function on data x and its label y
  - Let $L(x_k, \theta_k)$ denotes the regularizer of the k-th block (e.g., L_2 regularizer)
- The supervised learning problem can be formulated as

$$\inf_{(\theta(0),\cdots,\theta(K-1))\in\Theta^K} \mathbb{E}_{(x,y)\sim\mu}\left[\Phi(x(K), y) + \sum_{k=0}^{K-1} L(x(k), \theta(k))\right]$$

subject to

$$x(k+1) = x(k) + f(x(k), \theta(k)), \quad k = 0, \cdots, K-1, \quad x(0) = x$$

A mean-field optimal control formulation of deep learning

- The continuous-time version of the supervised learning problem is

$$\begin{aligned} &\inf_{\theta \in L^{\infty}([0,T],\Theta)} \mathbb{E}_{(x,y)\sim\mu} \left[\Phi(x(T),y) + \int_0^T L(x(t),\theta(t))dt \right] \\ &\text{subject to} \\ &\dot{x}(t) = f(x(t),\theta(t)), \quad t \in [0,T], \quad x(0) = x \end{aligned} \tag{5}$$

- Mean-field: emphasize the fact that we seek an optimal control that is shared with all input-label pairs (x, y).
- They are jointly distributed according to μ

Optimality conditions

- Pontryagin's maximum principle (PMP) gives necessary condition for optimality for our problem[4]

---

[4] Qianxiao Li, Long Chen, Cheng Tai, et al. "Maximum principle based algorithms for deep learning". In: *arXiv preprint arXiv:1710.09513* (2017).

Optimality conditions

- Pontryagin's maximum principle (PMP) gives necessary condition for optimality for our problem[4]
- We first define the Hamiltonian

$$
\begin{aligned}
&H : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^d \times \Theta \to \mathbb{R} \\
&H(t, x, p, \theta) = p^\top f(t, x, \theta) - L(t, x, \theta)
\end{aligned}
\tag{6}
$$

---

[4] Qianxiao Li, Long Chen, Cheng Tai, et al. "Maximum principle based algorithms for deep learning". In: *arXiv preprint arXiv:1710.09513* (2017).

Optimality conditions

- Pontryagin's maximum principle (PMP) gives necessary condition for optimality for our problem[4]
- We first define the Hamiltonian

$$\begin{aligned} &H : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^d \times \Theta \to \mathbb{R} \\ &H(t, x, p, \theta) = p^\top f(t, x, \theta) - L(t, x, \theta) \end{aligned} \tag{6}$$

Pontryagin's maximum principle

If θ^* is optimal and x^* is the corresponding state trajectory, then there exists an absolutely continuous process p

$$\begin{aligned} &\dot{x}^*(t) = \nabla_p H\left(t, x^*(t), p^*(t), \theta^*(t)\right), \quad x^*\left(t_0\right) = x_0 \\ &\dot{p}^*(t) = -\nabla_x H\left(t, x^*(t), p^*(t), \theta^*(t)\right), \quad p^*\left(t_1\right) = -\nabla_x \Phi\left(x^*\left(t_1\right), y\right) \\ &H\left(t, x^*(t), p^*(t), \theta^*(t)\right) \geqslant H\left(t, x^*(t), p^*(t), \theta\right) \\ &\forall \theta \in \Theta \text{ and a.e. } t \in [t_0, t_1] \end{aligned} \tag{7}$$

---

[4] Qianxiao Li, Long Chen, Cheng Tai, et al. “Maximum principle based algorithms for deep learning”. In: *arXiv preprint arXiv:1710.09513* (2017).

Understanding PMP from nonlinear program

- We can view the optimal control problem as a nonlinear program where the constraint is the ODE
- Co-state process p^* plays the role of a continuous-time analogue of Lagrange multipliers.
- The key difference between the PMP and the KKT conditions is the Hamiltonian maximization condition, which is stronger than a typical first-order condition that assumes smoothness with respect to θ
- In particular, the PMP says that H is not only stationary, but globally maximized at an optimal control

Outline

1. Introduction
2. Optimal control and deep learning
3. Control inspired learning algorithms
4. Control inspired architectures

Control inspired learning algorithm

Method of successive approximations (MSA)

start with an initial guess θ^0, at the n^{th} iteration we solve

$$\begin{aligned} \dot{x}^n(t) &= f\left(x^n(t), \theta^n(t)\right) & x^n(0) &= x \\ \dot{p}^n(t) &= -\nabla_x H\left(x^n(t), p^n(t), \theta^n(t)\right) & p^n(T) &= -\nabla_x \Phi\left(x^n(T), y\right) \\ \theta^{n+1}(t) &= \arg\max_{\theta \in \Theta} H\left(x^n(t), p^n(t), \theta\right). \end{aligned} \tag{8}$$

- if (x^n, p^n, θ^n) converges, then the limit must be a solution of the PMP

Connection between BP and MSA

- Recall

$$H(t,x,p,\theta) = p^{\top} f(t,x,\theta) - L(t,\theta) \tag{9}$$

Since

$$\dot{p}^n(t) = -\nabla_x H\left(x^n(t), p^n(t), \theta^n(t)\right), \quad p^n(T) = -\nabla_x \Phi\left(x^n(T), y\right) \tag{10}$$

It is easy to see

$$p^n(t) = -\nabla_{x^n(t)} \Phi\left(x^n(T), y\right) \tag{11}$$

Connection between BP and MSA

- Recall

$$H(t, x, p, \theta) = p^\top f(t, x, \theta) - L(t, \theta) \tag{9}$$

Since

$$\dot{p}^n(t) = -\nabla_x H\left(x^n(t), p^n(t), \theta^n(t)\right), \quad p^n(T) = -\nabla_x \Phi\left(x^n(T), y\right) \tag{10}$$

It is easy to see

$$p^n(t) = -\nabla_{x^n(t)} \Phi\left(x^n(T), y\right) \tag{11}$$

- Then we have

$$\begin{aligned}\nabla_{\theta(t)} J(\theta) &= \nabla_{x(t)} \Phi\left(x(T), y\right) \nabla_{\theta(t)} x(t) + \nabla_{\theta(t)} L(t, \theta) \\ &= -p(t) \nabla_{\theta(t)} f(x(t), \theta(t)) + \nabla_{\theta(t)} L(t, \theta) \\ &= \nabla_{\theta(t)} H\end{aligned} \tag{12}$$

Connection between BP and MSA

- For BP

$$\theta^{n+1}(t) = \theta^{n}(t) + \eta\nabla_{\theta} H\left(x^{n}(t), p^{n}(t), \theta\right) \tag{13}$$

- For MSA

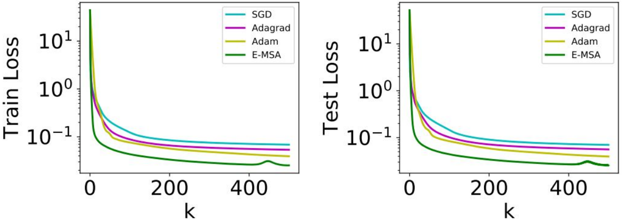
$$\theta^{n+1}(t) = \arg\max_{\theta\in\Theta} H\left(x^{n}(t), p^{n}(t), \theta\right) \tag{14}$$

- MSA is a generalization of the back-propagation algorithm

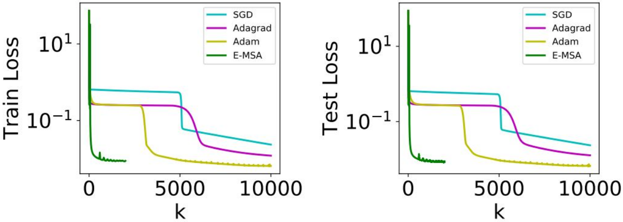
Experiments of MSA

The target function is $F(x) = \sin(x)$

Line 1: good initialization; line 2: bad initialization



(a)



Layer parallel training algorithm[5]

- Note that the equation for x has a initial condition, the equation for p has a terminal condition

$$x(0) = x, \quad p(T) = -\nabla_x \Phi\left(x(T), y\right)$$

We can break it down to two sub-problems

---

[5] Panos Parpas and Corey Muir. "Predict globally, correct locally: Parallel-in-time optimal control of neural networks". In: *arXiv preprint arXiv:1902.02542* (2019).

Layer parallel training algorithm[5]

- Note that the equation for x has a initial condition, the equation for p has a terminal condition

$$x(0) = x, \quad p(T) = -\nabla_x \Phi\left(x(T), y\right)$$

We can break it down to two sub-problems

- Let $S = T/2$,
  - P1: $(t \in [0, S])$

$$\begin{aligned} \dot{p}^n(t) &= -\nabla_x H\left(x^{n-1}(t), p^n(t), \theta^n(t)\right) \quad p^n(S) = p^{n-1}(S) \\ \dot{x}^n(t) &= f\left(x^n(t), \theta^n(t)\right) \quad x^n(0) = x \end{aligned} \tag{15}$$

  - P2: $(t \in [S, T])$

$$\begin{aligned} \dot{x}^n(t) &= f\left(x^n(t), \theta^n(t)\right) \quad x^n(S) = x^{n-1}(S) \\ \dot{p}^n(t) &= -\nabla_x H\left(x^n(t), p^n(t), \theta^n(t)\right) \quad p^n(T) = -\nabla_x \Phi\left(x^n(T), y\right) \end{aligned} \tag{16}$$

---

[5] Panos Parpas and Corey Muir. "Predict globally, correct locally: Parallel-in-time optimal control of neural networks". In: *arXiv preprint arXiv:1902.02542* (2019).

Layer parallel training algorithm[5]

- Note that the equation for x has a initial condition, the equation for p has a terminal condition

$$x(0) = x, \quad p(T) = -\nabla_x \Phi\left(x(T), y\right)$$

We can break it down to two sub-problems

- Let $S = T/2$,
  - P1: $(t \in [0, S])$

$$\begin{aligned} \dot{p}^n(t) &= -\nabla_x H\left(x^{n-1}(t), p^n(t), \theta^n(t)\right) \quad p^n(S) = p^{n-1}(S) \\ \dot{x}^n(t) &= f\left(x^n(t), \theta^n(t)\right) \quad x^n(0) = x \end{aligned} \tag{15}$$

  - P2: $(t \in [S, T])$

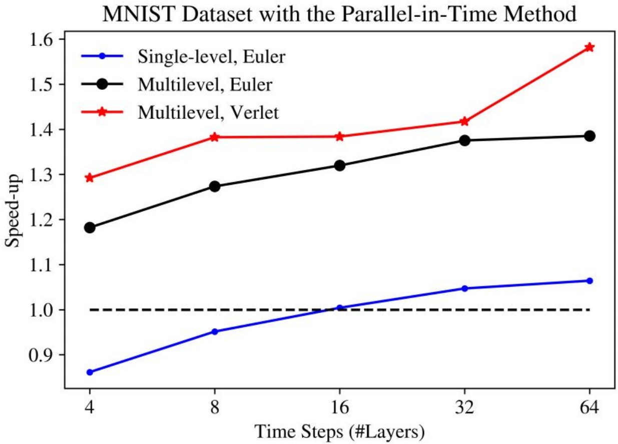
$$\begin{aligned} \dot{x}^n(t) &= f\left(x^n(t), \theta^n(t)\right) \quad x^n(S) = x^{n-1}(S) \\ \dot{p}^n(t) &= -\nabla_x H\left(x^n(t), p^n(t), \theta^n(t)\right) \quad p^n(T) = -\nabla_x \Phi\left(x^n(T), y\right) \end{aligned} \tag{16}$$

- P1 and P2 can be run in parallel

---

[5] Panos Parpas and Corey Muir. "Predict globally, correct locally: Parallel-in-time optimal control of neural networks". In: *arXiv preprint arXiv:1902.02542* (2019).

Layer-parallel vs data-parallel



Layer-parallel implementation has more merit when network is deep

Outline

1. Introduction
2. Optimal control and deep learning
3. Control inspired learning algorithms
4. **Control inspired architectures**

Control inspired architecture

Stability of linear system

A simple linear ODE $\dot{x}(t) = Ax(t), \quad x(0) = x_0 \Rightarrow x(t) = e^{tA}x_0$
$\lambda_1, \cdots, \lambda_d \in \mathbb{C}$ are the eigenvalues of A. x_ϵ is a small perturbation of x.
Then 1) if $\mathcal{R}(\lambda_i) \leqslant 0 \Rightarrow \|x(t) - x_\epsilon(t)\|$ is bounded; 2) if $\mathcal{R}(\lambda_i) > 0$ for some $i \Rightarrow \|x(t) - x_\epsilon(t)\| \to \infty$ as $t \to \infty$

---

[6] Eldad Haber and Lars Ruthotto. “Stable architectures for deep neural networks”. In: *Inverse problems* 34.1 (2017), p. 014004.

Control inspired architecture

Stability of linear system

A simple linear ODE $\dot{x}(t) = Ax(t), \quad x(0) = x_0 \Rightarrow x(t) = e^{tA}x_0$
$\lambda_1, \cdots, \lambda_d \in \mathbb{C}$ are the eigenvalues of A. x_ϵ is a small perturbation of x. Then 1) if $\mathcal{R}(\lambda_i) \leqslant 0 \Rightarrow \|x(t) - x_\epsilon(t)\|$ is bounded; 2) if $\mathcal{R}(\lambda_i) > 0$ for some $i \Rightarrow \|x(t) - x_\epsilon(t)\| \to \infty$ as $t \to \infty$

- If A is anti symmetric,i.e., $A^T = -A$ then $\mathcal{R}(\lambda_i) = 0$
- Anti symmetric can be constructed by $A = B - B^T$
- One can reparameterize the neural network in a similar way[6]

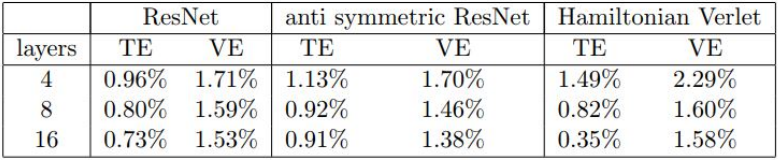
$$\dot{x}(t) = \sigma(W(t)x(t)+b(t)) \to \quad \dot{x}(t) = \sigma\left(\left[V(t) - V(t)^\top\right] x(t) + b(t)\right) \tag{17}$$

---

[6] Eldad Haber and Lars Ruthotto. “Stable architectures for deep neural networks”. In: *Inverse problems* 34.1 (2017), p. 014004.

Experiments

Comparison of training error (TE) and validation error (VE)



	ResNet		anti symmetric ResNet		Hamiltonian Verlet	
layers	TE	VE	TE	VE	TE	VE
4	0.96%	1.71%	1.13%	1.70%	1.49%	2.29%
8	0.80%	1.59%	0.92%	1.46%	0.82%	1.60%
16	0.73%	1.53%	0.91%	1.38%	0.35%	1.58%

The antisymmetric ResNet giving slightly lower validation errors at each level

Backward Euler discretization[7]

- ResNet corresponds to the forward Euler discretization

---

[7] Yiping Lu et al. “Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3276–3285.

Backward Euler discretization[7]

- ResNet corresponds to the forward Euler discretization
- The backward Euler is

$$\begin{aligned} \widehat{x}(k+1) &= \widehat{x}(k) + \Delta t f((k+1)\Delta t, \widehat{x}(k+1)) \\ &\Downarrow \\ \widehat{x}(k+1) &= (I - \Delta t f((k+1)\Delta t, \cdot))^{-1}(\widehat{x}(k)) \end{aligned} \tag{18}$$

- For linear f one can expand as

$$\widehat{x}(k+1) \approx \widehat{x}(k) + \Delta t f((k+1)\Delta t, \widehat{x}(k)) + \Delta t^2 [f((k+\ 1)\ \ \Delta t, \cdot)^2] \, (\widehat{x}(k)) \tag{19}$$

---

[7] Yiping Lu et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3276–3285.

Backward Euler discretization[7]

- ResNet corresponds to the forward Euler discretization
- The backward Euler is

$$\begin{gathered}\widehat{x}(k+1)=\widehat{x}(k)+\Delta t f((k+1)\Delta t,\widehat{x}(k+1))\\ \Downarrow\\ \widehat{x}(k+1)=(I-\Delta t f((k+1)\Delta t,\cdot))^{-1}(\widehat{x}(k))\end{gathered}\tag{18}$$

- For linear f one can expand as

$$\widehat{x}(k+1)\approx\widehat{x}(k)+\Delta t f((k+1)\Delta t,\widehat{x}(k))+\Delta t^2\left[f((k+1)\ \ \Delta t,\cdot)^2\right](\widehat{x}(k))\tag{19}$$

- In numerical analysis, backward Euler enjoy better stability

---

[7] Yiping Lu et al. “Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3276–3285.

Linear multi-step discretization

- Another family of networks based on linear multi-step discretization

$$\widehat{x}(k+1) = (1-\alpha_k)\,\widehat{x}(k) + \alpha_k \widehat{x}(k-1) + \Delta t f(k\Delta t, \widehat{x}(k)) \tag{20}$$

- The weight α_k as a trainable parameter

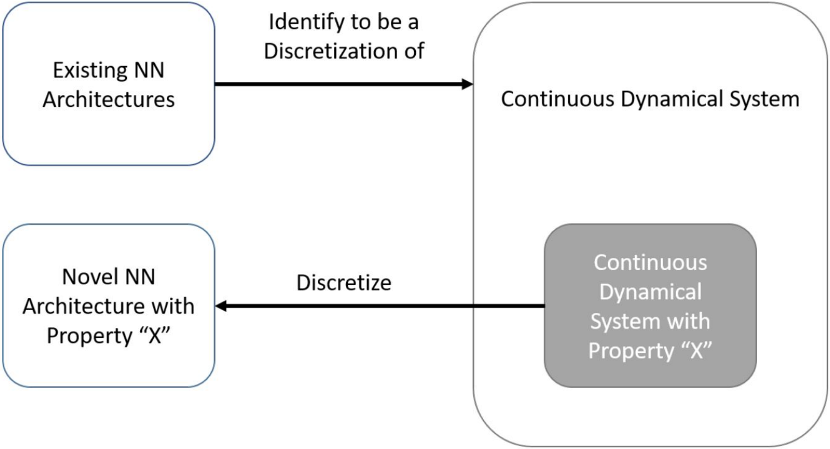
- DenseNet[8] can be thought of as an extreme case of such a multi-step method

---

[8] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

Architecture from PDE Theory

- We derive new models with better robustness or other good properties from PDE Theory



Existing NN Architectures

Identify to be a Discretization of

Continuous Dynamical System

Continuous Dynamical System with Property “X”

Discretize

Novel NN Architecture with Property “X”

Connection to transport equation (TE)[9]

- For an ODE,

$$\frac{dx}{dt} = F(A(t), x), \;\; x(0) = x_0$$

  let $u(x, t)$ be a function that is constant along the trajectory defined by the ODE

- Then

$$\frac{du(x(t), t)}{dt} = \frac{\partial u(x, t)}{\partial t} + F(A(t), x)\frac{\partial u}{\partial x} = 0$$

- Enforce $u(x(1), 1) = f(x(0))$ and $u(x(0), 0) = y$

- Then training ResNet is to find u and $u(x(0), 0)$ is the classifier

---

[9] Zhen Li and Zuoqiang Shi. “Deep residual learning and pdes on manifold”. In: *arXiv preprint arXiv:1708.05115* (2017).

TE view of ResNet

- The formulization is

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} + F(A(t),x)\dfrac{\partial u}{\partial x} = 0 \\ u(x(1),1) = f(x(0)) \\ u(x(0),0) = y \end{cases} \tag{21}$$

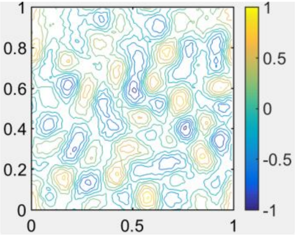
- When F is complex, $u(x,0)$ may be highly irregular
- TE more regular → ResNet more robust
- Solve the convection diffusion equation

$$\frac{\partial u(x,t)}{\partial t} + F(A(t),x)\frac{\partial u}{\partial x} + \frac{1}{2}\sigma^2\Delta u(x,t) = 0 \tag{22}$$

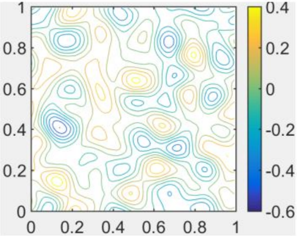
instead

Regularity of convection diffusion equation

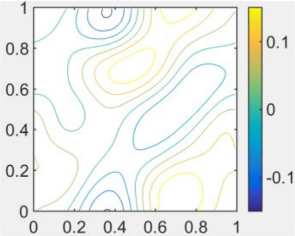
Terminal condition of convection diffusion equation with different σ



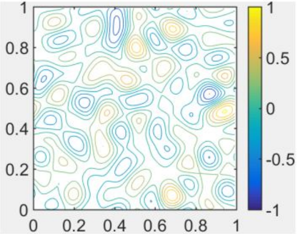
(a) $\sigma = 0$



(b) $\sigma = 0.01$



(c) $\sigma = 0.1$



(d) $u(\mathbf{x}, 1)$

ResNets Ensemble via the Feynman-Kac Formula

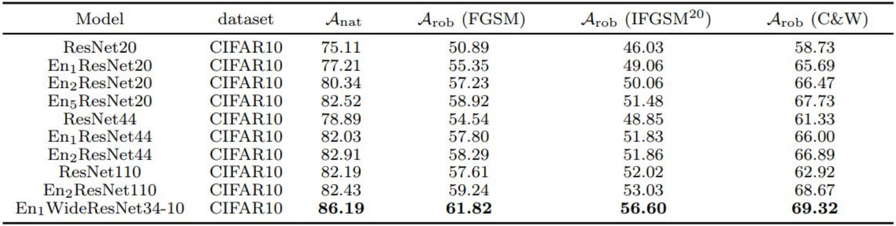
- The convection-diffusion equation can be solved using the Feynman-Kac formula in high dimensional space

$$dx(t) = F(A(t), x)dt + \sigma dB_t \tag{23}$$

- The term σdB_t in the Itô process that can be approximated by adding a specially designed Gaussian noise, $\sigma N(0, I)$

- We approximate the Feynman-Kac formula by an ensemble of modified ResNets

Results of experiment



Model	dataset	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{rob}}$ (FGSM)	$\mathcal{A}_{\text{rob}}$ (IFGSM^{20})	$\mathcal{A}_{\text{rob}}$ (C&W)
ResNet20	CIFAR10	75.11	50.89	46.03	58.73
En_1ResNet20	CIFAR10	77.21	55.35	49.06	65.69
En_2ResNet20	CIFAR10	80.34	57.23	50.06	66.47
En_5ResNet20	CIFAR10	82.52	58.92	51.48	67.73
ResNet44	CIFAR10	78.89	54.54	48.85	61.33
En_1ResNet44	CIFAR10	82.03	57.80	51.83	66.00
En_2ResNet44	CIFAR10	82.91	58.29	51.86	66.89
ResNet110	CIFAR10	82.19	57.61	52.02	62.92
En_2ResNet110	CIFAR10	82.43	59.24	53.03	68.67
En_1WideResNet34-10	CIFAR10	**86.19**	**61.82**	**56.60**	**69.32**

Ensemble of ResNets is more robust to different attacks (FGSM, IFGSM and C&W)