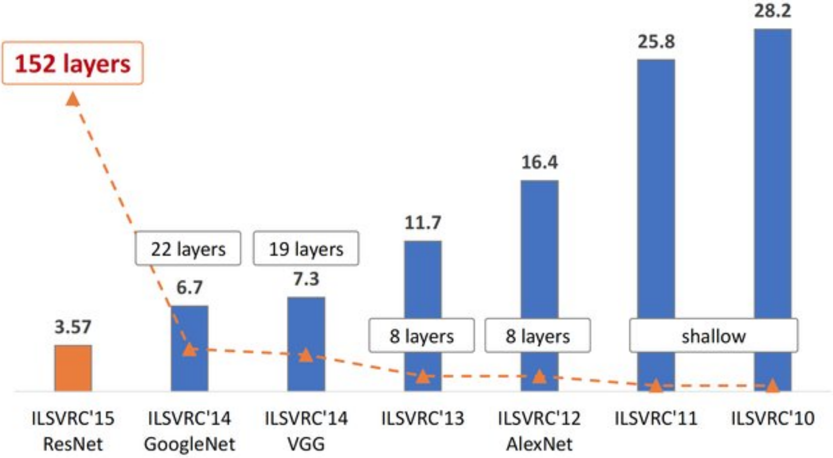
ODE as Continuous Depth Nueral Networks:
Modeling, Optimization, and Inferencing

Joint work with Bin Dong, Di He, Liwei Wang, Jianfeng Lu, Lexing Ying and et al.

Presenter:
Yiping Lu
Contact:
yplu@stanford.edu,
https://web.stanford.edu/~yplu/

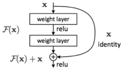
Stanford University

Deep Learning Evolution



Stanford University

ODE As Infinite Depth Neural Network



First-order ODE ⟹ Numerical solver: Euler's method ⟹ ResNet

First-order ODE

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

$$x(0) = x_0,$$

Numerical solver: Euler's method

$$x_{l+1} = x_l + \gamma F\big(x_l, t_l\big),$$

$$x_0 \doteq x(0), x_l = x\big(\gamma l\big), \ldots \gamma - step\ size$$

ResNet

[He et al. 2015]

$$x_{l+1} = x_l + F\big(x_l, t_l\big),$$

[E. 2017] [Haber et al. 2017] [Lu et al. 2017] [Sho et al 2017] [Chen et al. 2018]

Figure: ResNet can be seen as the Euler discretization of a time evolving ODE

Outline

Modeling

Optimization

MIN L(

)

Inferencing

Stanford University

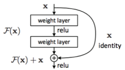
Outline Of The Talk

1 **Modeling**

2 **Optimization**
- Algorithm Design
- Theory

3 **Inferencing**

Numerical Scheme As Architecture



First-order ODE ⟹ Numerical solver: Euler's method ⟹ ResNet

[He et al. 2015]

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

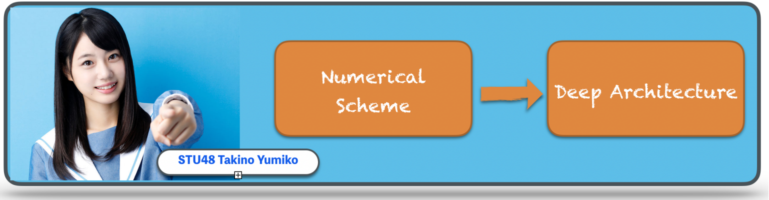
$$x(0) = x_0,$$

$$x_{l+1} = x_l + \gamma F(x_l, t_l),$$

$$x_0 \doteq x(0), x_l = x(\gamma l), \ldots \gamma - step\ size$$

$$x_{l+1} = x_l + F(x_l, t_l),$$

[E. 2017] [Haber et al. 2017] [Lu et al. 2017] [Sho et al 2017] [Chen et al. 2018]



Numerical Scheme: Skip Connection

Observation:

- ResNe(X)t = Euler Scheme
- PolyNet = An Approximation Of Implicit Scheme
- FractalNet=Runge-Kutta Schmeme ....

Numerical scheme can be used to desgin principled skip connection

All exisisting scheme are single step scheme.
Our paper[1] introduced a linear multi-step scheme to ResNet, and explained why it works.

[1] **Yiping Lu**, Aoxiao Zhong, Quanzheng Li, Bin Dong. "Beyond Finite Layer Neural Network:Bridging Deep Architects and Numerical Differential Equations" Thirty-fifth International Conference on Machine Learning (ICML), 2018

Dropout: Stochastic Differential Equation

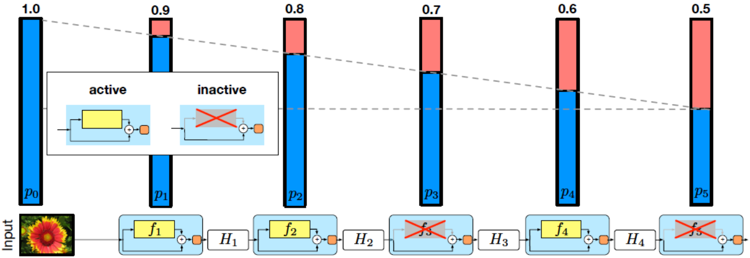


Figure: Stochastic Depth

Convergeto SDE

$$dX_t = p(t)f(X)dt + \sqrt{p(t)(1 - p(t))}f(X_t) \odot [\mathbb{1}_{0\times 1}, 0_{0\times 1}]dB_t$$

Convergence Requirement meets parameter selection.

Stanford University

Modeling Seq2Seq: Transformer

Understand Transformer as a **multi-particle system.**

$$\frac{\mathrm{d}x_i(t)}{\mathrm{d}t} = \underbrace{F(x_i(t), [x_1(t), \cdots, x_n(t)], t)}_{\text{Attention Layer}} + \underbrace{G(x_i(t), t)}_{\text{FFN Layer}},$$

$$x_i(t_0) = w_i, \quad i = 1, \ldots, n.\text{(Every words in a sentence)} \tag{1}$$

Transformer is a **splitting scheme**, splitting F and G.

Stanford University

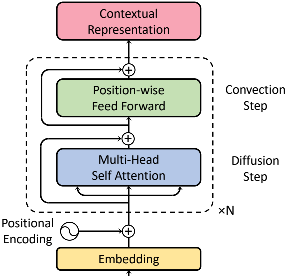
Modeling Seq2Seq: Transformer

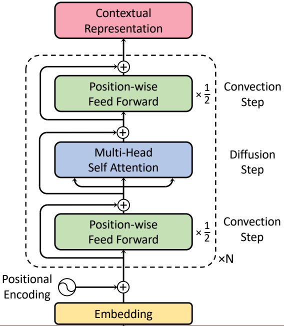
Understand Transformer as a **multi-particle system.**

$$\frac{\mathrm{d}x_i(t)}{\mathrm{d}t} = \underbrace{F(x_i(t), [x_1(t), \cdots, x_n(t)], t)}_{\text{Attention Layer}} + \underbrace{G(x_i(t), t)}_{\text{FFN Layer}},$$

$$x_i(t_0) = w_i, \quad i = 1, \ldots, n.(\text{Every words in a sentence}) \tag{1}$$

Transformer is a **splitting scheme**, splitting F and G.
Applying an higher order splitting scheme?





Stanford University

Results

Table: Translation performance (BLEU) on IWSLT14 De-En and WMT14 En-De testsets.

	IWSLT14 De-En	**WMT14 En-De**	
Method	`small`	`base`	`big`
Transformer	34.4	27.3	28.4
Weighted Transformer	/	28.4	28.9
Relative Transformer	/	26.8	29.2
Universal Transformer	/	**28.9**	/
Scaling NMT	/	/	29.3
Dynamic Conv	35.2	/	29.7
Macaron Net	**35.4**	**28.9**	**30.2**

Stanford University

Results

Table: Test results on the GLUE benchmark (except WNLI).

Method	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE	**GLUE**
Existing systems									
ELMo	33.6	90.4	84.4/78.0	74.2/72.3	63.1/84.3	74.1/74.5	79.8	58.9	70.0
OpenAI GPT	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.7/80.6	87.2	69.1	76.9
BERT `base`	52.1	93.5	**88.9**/**84.8**	87.1/85.8	**71.2**/**89.2**	84.6/83.4	90.5	66.4	78.3
Our systems									
BERT `base` (ours)	52.8	92.8	87.3/83.0	81.2/80.0	70.2/88.4	84.4/83.7	90.4	64.9	77.4
Macaron Net `base`	**57.6**	**94.0**	88.4/84.4	**87.5**/**86.3**	70.8/89.0	**85.4**/**84.5**	**91.6**	**70.5**	**79.7**

Stanford University

Neural ODE: Enforcing Constraint

- **Implicit Scheme:** Stability. *(Behrmann J, et al. Invertible residual networks. ICML2019.)*
- **Symplectic Scheme:** Energy Conservation.*(Chen Z, et al. Symplectic Recurrent Neural Networks. ICLR2020)*
- **Approximation To Optimal Transport map:***(Finlay C, Jacobsen J H, Nurbekyan L, et al. How to train your neural ODE. arXiv preprint arXiv:2002.02798, 2020.)*
- **Adversarial Examples:** *(Zhang J, Han B, Wynter L, et al. Towards robust resnet: A small step but a giant leap. IJCAI2019.)*

ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations: http://iclr2020deepdiffeq.rice.edu/

(Invited Talk 3: Subtleties of Neural ODEs: Learning with Constraints By Ricky Chen.)

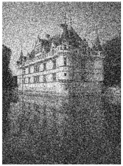
Our Example: DURR

Train

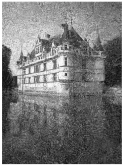
10dB Noise | 30dB Noise | 50dB Noise

Test

70dB Noise



BM3D



DnCNN

Traditional Method vs Deep Learning

How can we encode the physic of task?

Our Example: DURR

Train

10dB Noise	30dB Noise	50dB Noise

Test

70dB Noise

BM3D — DnCNN

Traditional Method VS Deep Learning

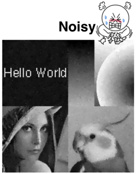
How can we encode the physic of task?

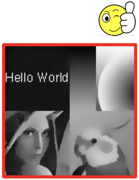
input

$$
\begin{cases}
\dfrac{\partial u}{\partial t} = \operatorname{div}\left(c(|\nabla u|^2)\, \nabla u \right) & \text{in } \Omega \times (0,\mathrm{T}), \\
\dfrac{\partial u}{\partial N} = 0 & \text{on } \partial\Omega \times (0,T), \\
u(0,x) = u_0(x) & \text{in } \Omega,
\end{cases}
$$

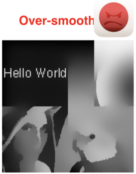
Perona-Malik Equation

Noisy





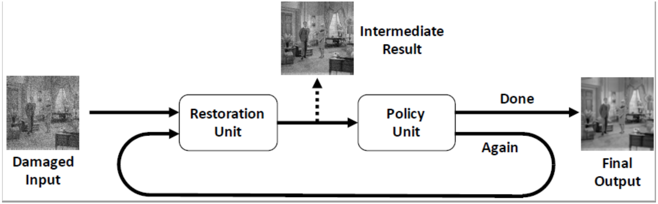
Over-smooth



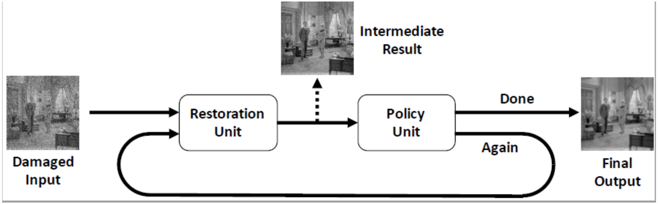
processing

Stanford University

Our Example: DURR



Our Example: DURR



	BM3D	WNNM	DnCNN-B	UNLNet_5	DURR
$\sigma = 25$	28.55	28.73	29.16	28.96	29.16
$\sigma = 35$	27.07	27.28	27.66	27.50	27.72
$\sigma = 45$	25.99	26.26	26.62	26.48	26.71
$\sigma = 55$	25.26	25.49	25.80	25.64	25.91
$\sigma = 65$	24.69	24.51	23.40*	-	25.26*
$\sigma = 75$	22.63	22.71	18.73*	-	24.71*

Stanford University

Modling The Physics

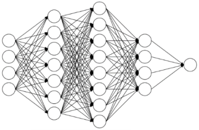
Tycho Brahe

Johannes Kepler

Isaac Newton

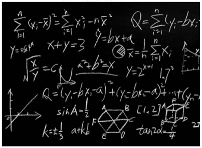
phenomenon

discipline



Our Work

Law



PDE-Net: Modeling The Physics

Consider a finite difference scheme for PDE:

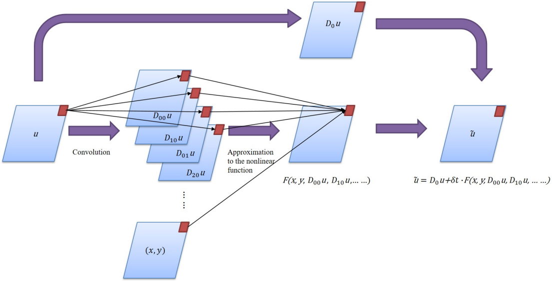
$$u_{i\pm1} = \left[u \pm h\partial_x u + \frac{h^2}{2}\partial_x^2 u \pm \frac{h^3}{6}\partial_x^3 u + \frac{h^4}{24}\partial_x^4 u \pm \frac{h^5}{120}\partial_x^5 u + \cdots \right]_i$$

Thus

$$\left|\frac{u_{i+1} - 2u_i + u_{i-1}}{2} - u''\right| = \left|\frac{1}{12}h^2(\partial_x^4 u)_i + O(h^4)\right|$$

	1	
1	-4	1
	1	

$$\Delta u = u_{xx} + u_{yy}$$



University

Convolution Operator As Differential Operator

Definition (Order of Sum Rules)

For a filter q, we say q to have sum rules of order $\alpha = (\alpha_1, \alpha_2)$, where $\alpha \in \mathbb{Z}_+^2$, provided that

$$\sum_{k \in \mathbb{Z}^2} k^{\beta} q[k] = 0 \tag{2}$$

for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| < |\alpha|$ and for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| = |\alpha|$ but $\beta \neq \alpha$. If (2) holds for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| < K$ except for $\beta \neq \beta_0$ with certain $\beta_0 \in \mathbb{Z}_+^2$ and $|\beta_0| = J < K$, then we say q to have total sum rules of order $K \backslash \{J+1\}$.

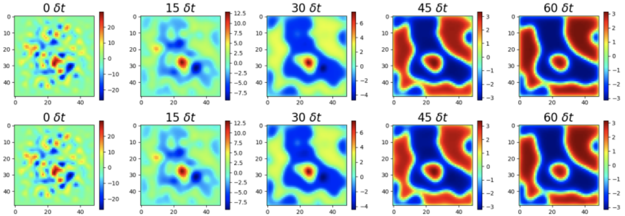
Theorem

Let q be a filter with sum rules of order $\alpha \in \mathbb{Z}_+^2$. Then for a smooth function $F(x)$ on $\mathbb{R}^2$, we have

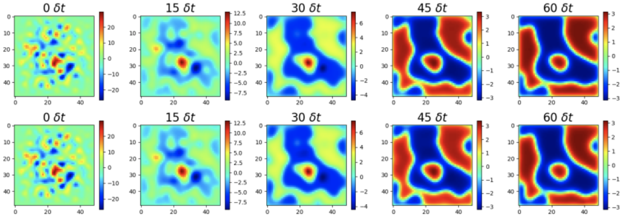
$$\frac{1}{\varepsilon^{|\alpha|}} \sum_{k \in \mathbb{Z}^2} q[k] F(x + \varepsilon k) = C_\alpha \frac{\partial^\alpha}{\partial x^\alpha} F(x) + O(\varepsilon), \text{ as } \varepsilon \to 0, \tag{3}$$

where C_α is the constant defined by $C_\alpha = \frac{1}{\alpha!} \sum_{k \in \mathbb{Z}^2} k^\alpha q[k]$.

PDE-Net: Recovering Coefficients



PDE-Net: Recovering Coefficients



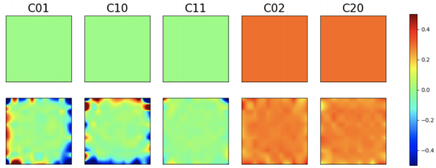
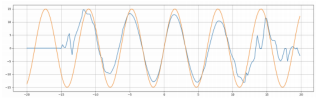
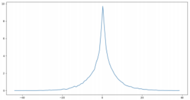


Figure 15: First row: the true coefficients $\{f_{ij} : 1 \le i + j \le 2\}$ of the equation. Second row: the learned coefficients $\{c_{ij} : 1 \le i + j \le 2\}$ by the PDE-Net with 3 δt-blocks and 7×7 filters.





Stanford University

Outline Of The Talk

1 Modeling

2 **Optimization**
- Algorithm Design
- Theory

3 Inferencing

Stanford University

An Optimal Control View of Deep Learning

Deep learning:

$$\min_{\theta} J(\theta) = \ell(x_T) + \sum_{t=0}^{T-1} R_t(x_t; \theta_t) \tag{4}$$
$$s.t. \quad x_{t+1} = f_t(x_t, \theta_t), t = 1, 2, \cdots, T-1$$

An Optimal Control View of Deep Learning

Deep learning:

$$\min_{\theta} J(\theta) = \ell(x_T) + \sum_{t=0}^{T-1} R_t(x_t;\theta_t)$$
$$s.t. \quad x_{t+1} = f_t(x_t,\theta_t), t = 1,2,\cdots,T-1 \tag{4}$$

Optimal Control:

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(\mathbf{x}(T)) + \int_0^T R(\mathbf{x}(t),\theta(t))dt$$
$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\theta(t)) \tag{5}$$

$\theta(\cdot)$ is called a **control**

Stanford University

An Optimal Control View of Deep Learning

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(\mathbf{x}(T)) + \int_0^T R(\mathbf{x}(t), \theta(t))dt \qquad (6)$$
$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta(t))$$

Stanford University

An Optimal Control View of Deep Learning

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(\mathbf{x}(T)) + \int_0^T R(\mathbf{x}(t), \theta(t))dt \qquad (6)$$
$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta(t))$$

Gradient Based Training: Adjoint Equation

$$\dot{p}(t) = -\nabla_x H(x(t), p(t), \theta(t))$$

A New method?

An Optimal Control View of Deep Learning

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(\mathbf{x}(T)) + \int_0^T R(\mathbf{x}(t), \theta(t))dt \tag{6}$$
$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta(t))$$

Gradient Based Training: Adjoint Equation

$$\dot{p}(t) = -\nabla_x H(x(t), p(t), \theta(t))$$

A New method? **NO!**.

$\boxed{\textbf{Adjoint Equation = Back Propagation!}}$

An Optimal Control View of Deep Learning

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(\mathbf{x}(T)) + \int_0^T R(\mathbf{x}(t), \theta(t))dt \qquad (6)$$
$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta(t))$$

Gradient Based Training: Adjoint Equation

$$\dot{p}(t) = -\nabla_x H(x(t), p(t), \theta(t))$$

A New method? **NO!**.

$\boxed{\textbf{Adjoint Equation = Back Propagation!}}$

Benefit:

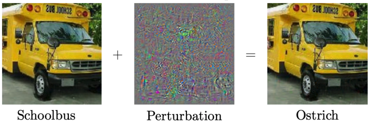
- **Invertible:** Neural Ordinary Differential Equation Neruips2018.
- **Find out structure!** $\boxed{\text{(Our work)}}$

Stanford University

Outline Of The Talk

1. **Modeling**
2. **Optimization**
   - Algorithm Design
   - Theory
3. **Inferencing**

Adversarial Training

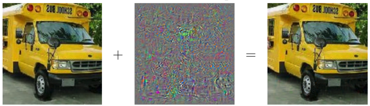


(Szegedy et al, 2013)

Robust Optimization

$$min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{||\eta||\leq\epsilon} \ell(\theta; x+\eta, y),$$

Adversarial Training



(rescaled for visualization)
(Szegedy et al, 2013)

Robust Optimization

$$min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{\|\eta\|\leq\epsilon} \ell(\theta; x+\eta, y),$$

PGD Method

- Gradient ascent on x.

$$x^{t+1} = \Pi_{x+\mathcal{S}}\left(x^t + \alpha\,\mathrm{sign}\left(\nabla_x \ell\right)\right)$$

$\boxed{\text{for } r \text{ times.}}$

- Gradient Descent On θ.

$$\theta = \theta - \nabla_\theta \ell$$

$\boxed{\text{for } 1 \text{ times.}}$

Stanford University

Our Intuition: Spilitting The Gradient

YOPO(**Y**ou **O**nly **P**ropogate **O**nce)

initialize perturbation η
for $\boxed{k = 1 \text{ to } m}$ **do**

m times full backprop.

    $p \leftarrow \nabla_{f_0} \ell(x + \eta)$
    for $\boxed{i = 1 \text{ to } n}$ **do**

Focus on first layer.

splitting

        $\eta \leftarrow \eta + \alpha \cdot p \cdot \nabla_x f_0(x + \eta)$
    end for
    accumulate gradient $U \leftarrow U + \nabla_\theta \ell(x + \eta)$

use intermediate adversarial examples

end for
Use U tp perform SGD / momentum SGD

Stanford University

A Differential Game View of Adversarial Training

Adversarial Training:

$$\min_{\theta} \max_{\|\eta\|\leq\epsilon} J(\theta,\eta) = \ell(x_T) + \sum_{t=0}^{T-1} R_t(x_t;\theta_t,\eta_t) \tag{7}$$

$$s.t. \quad x_1 = f_0(x_0+\eta,\theta_0), x_{t+1} = f_t(x_t,\theta_t), t = 1,2,\cdots,T-1$$

Differential Game:

$$\min_{\theta(\cdot)} \max_{\eta(\cdot)} J[\theta(\cdot),\eta(\boldsymbol{\cdot})] = \ell(\mathbf{x}(T)) + \int_0^T R(\mathbf{x}(t),\theta(t),\eta(\boldsymbol{t}))dt \tag{8}$$

$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\theta(t),\eta(\boldsymbol{t}))$$

Differential game is optimal control with 2 controls, **each having opposite target**.

Stanford University

YOPO: An Optimal Control View

- Pontryagin's Maximal Principle (PMP) is a neccesary condition for optimal control problem (Stronger than KKT.)
- We'll show that **YOPO is actually a discretion of PMP**

YOPO: An Optimal Control View

- Pontryagin's Maximal Principle (PMP) is a neccesary condition for optimal control problem (Stronger than KKT.)
- We'll show that **YOPO is actually a discretion of PMP**

Define *Hamiltonian*

$$H(x, p, \theta, \eta) := p \cdot \mathbf{f}(x, \theta, \eta) + r(x, \theta, \eta)$$

PMP for differential game tells us there exists an **adjoint dynamic $\mathbf{p}(\cdot)$** satisfying :

$$\begin{aligned}
\dot{\mathbf{x}}^*(t) &= \nabla_p H\left(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta^*(t)\right) \\
\dot{\mathbf{p}}^*(t) &= -\nabla_x H\left(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta^*(t)\right) \\
H\left(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta\right) &\geq H\left(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta^*(t)\right) \\
&\geq H\left(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, \eta^*(t)\right), \quad \forall t, \eta, \theta
\end{aligned}$$

Stanford University

YOPO: An Optimal Control View

- Pontryagin's Maximal Principle (PMP) is a neccesary condition for optimal control problem
- We'll show that **YOPO is actually a discretization of PMP**

$$\dot{\mathbf{x}}^*(t) = \nabla_p H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta^*(t))$$

The same as the forward equation $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta(t), \eta(\boldsymbol{t}))$.

$$\dot{\mathbf{p}}^*(t) = -\nabla_x H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta^*(t))$$

Known as **Adjoint Equation**, the same as back propagation on feature map $\mathbf{x}(t)$. *i.e.* $\mathbf{p}(t) = \frac{\partial J}{\partial \mathbf{x}(t)}$

$$\begin{aligned} H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta) &\geq H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), \eta^*(t)) \\ &\geq H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, \eta^*(t)), \quad \forall t, \eta, \theta \end{aligned}$$

Parameter θ, η should optimize the Hamiltonian. $\eta(0)$ only coupled with the first layer.

Stanford University

Decoupled Training

Back propagation is a sequential process, how can we parallelize it?

- *ADMM*: Taylor G, Burmeister R, Xu Z, et al. Training neural networks without gradients: A scalable admm approach ICML2016.
- *Coordinate Descent*:Zeng J, Lau T T K, Lin S, et al. Global convergence of block coordinate descent in deep learning ICML2018.
- *Lifted machines*:Li J, Fang C, Lin Z. Lifted proximal operator machines AAAI 2019.
- **ODE Based Methods**:Gunther S, Ruthotto L, Schroder J B, et al. Layer-parallel training of deep residual neural networks. SIMDOS

CIFAR10 WideResNet34 Results

Training Methods	Clean Data	PGD-20 Attack	Training Time (mins)
Natural train	95.03%	0.00%	233
PGD-3	90.07%	39.18%	1134
PGD-5	89.65%	43.85%	1574
PGD-10	87.30%	47.04%	2713
Free-8 [1]	86.29%	47.00%	667
YOPO-3-5 (Ours)	87.27%	43.04%	299
YOPO-5-3 (Ours)	86.70%	47.98%	**476**

Table: Results of Wide ResNet34 for CIFAR10.

Take Home Message

Bridging

- Adversrial Training
- Differential Game.

Take Home Message

Bridging

- Adversrial Training
- Differential Game.

YOPO(**Y**ou **O**nly **P**ropogate **O**nce)

1. Split the network
   Assuming p unchanged in inner iteration, YOPO increase update iteration number with slightly more computation

Stanford University

Take Home Message

Bridging

- Adversrial Training
- Differential Game.

YOPO(**Y**ou **O**nly **P**ropogate **O**nce)

1. Split the network
   Assuming p unchanged in inner iteration, YOPO increase update iteration number with slightly more computation
2. Use intermediate perturbation to update weights θ

Stanford University

Take Home Message

Bridging

- Adversrial Training
- Differential Game.

YOPO(**Y**ou **O**nly **P**ropogate **O**nce)

1. Split the network
   Assuming p unchanged in inner iteration, YOPO increase update iteration number with slightly more computation
2. Use intermediate perturbation to update weights θ

YOPO can be understood as

$$\boxed{\textbf{discretization way solving PMP}}$$

Stanford University

Outline Of The Talk

1. **Modeling**
2. **Optimization**
   - Algorithm Design
   - Theory
3. **Inferencing**

Stanford University

Global Convergence Proof Of NN

- Neural Tangent Kernel([Jacot et al.2019]):Linearize the model
  $f_{\text{NN}}(\theta) = f_{\text{NN}}(\theta_{\text{init}}) + < \nabla_\theta f_{\text{NN}}(\theta_{\text{init}}), \theta - \theta_{\text{init}} >$

Global Convergence Proof Of NN

- Neural Tangent Kernel([Jacot et al.2019]):Linearize the model $f_{\text{NN}}(\theta) = f_{\text{NN}}(\theta_{\text{init}}) + <\nabla_\theta f_{\text{NN}}(\theta_{\text{init}}), \theta - \theta_{\text{init}} >$
  - **Pro:** can provide proof of convergence for any structure of NN. ([Li et al. 2019])
  - **Con:** Feature is lazy learned, *i.e.* not data dependent. ([Chizat and Bach 2019.][Ghorbani et al.2019])

Global Convergence Proof Of NN

- Neural Tangent Kernel([Jacot et al.2019]):Linearize the model $f_{\text{NN}}(\theta) = f_{\text{NN}}(\theta_{\text{init}}) + <\nabla_\theta f_{\text{NN}}(\theta_{\text{init}}), \theta - \theta_{\text{init}} >$
  - **Pro:** can provide proof of convergence for any structure of NN. ([Li et al. 2019])
  - **Con:** Feature is lazy learned, *i.e.* not data dependent. ([Chizat and Bach 2019.][Ghorbani et al.2019])

- Mean Field Regime([Bengio et al.2006][Bach et al.2014][Suzuki et al.2015]): We consider properties of the loss landscape with respect to the distribution of weights $L(\rho) = \|\mathbb{E}_{\theta\sim\rho} g(\theta, x) - f(x)\|_2^2$, the objective is a convex function

Global Convergence Proof Of NN

- Neural Tangent Kernel([Jacot et al.2019]):Linearize the model $f_{\mathrm{NN}}(\theta) = f_{\mathrm{NN}}(\theta_{\mathrm{init}}) + < \nabla_\theta f_{\mathrm{NN}}(\theta_{\mathrm{init}}), \theta - \theta_{\mathrm{init}} >$
  - **Pro:** can provide proof of convergence for any structure of NN. ([Li et al. 2019])
  - **Con:** Feature is lazy learned, *i.e.* not data dependent. ([Chizat and Bach 2019.][Ghorbani et al.2019])

- Mean Field Regime([Bengio et al.2006][Bach et al.2014][Suzuki et al.2015]): We consider properties of the loss landscape with respect to the distribution of weights $L(\rho) = \|\mathbb{E}_{\theta\sim\rho} g(\theta, x) - f(x)\|_2^2$, the objective is a convex function
  - **Pro:** SGD = Wasserstein Gradient Flow ([Mei et al.2018][Chizat et al.2018][Rotskoff et al.2018])
  - **Con:** Hard to generalize beyond two layer

Stanford University

Mean Field ResNet

Naive ODE analogy does not directly provide guarantees of global convergence even in the continum limit.

Our Aim: Provide a **new** continuous limit for ResNet with good limiting landscape.

Idea:We consider properties of the loss landscape with respect to the distribution of weights.

Mean Field ResNet

Naive ODE analogy does not directly provide guarantees of global convergence even in the continum limit.

Our Aim: Provide a **new** continuous limit for ResNet with good limiting landscape.

Idea:We consider properties of the loss landscape with respect to the distribution of weights.

$$\dot{X}_\rho(x,t) = \int_\theta f(X_\rho(x,t),\theta)\boxed{\rho(\theta,t)}d\theta$$

Residual block sample from ρ

Here:

- Input data is the initial condition $X_\rho(x,0) = \langle w_2, x\rangle$
- X is the feature, t represents the depth.
- Loss function:$E(\rho) = \mathbb{E}_{x\sim\mu}\left[\frac{1}{2}\left(\left\langle w_1, X_\rho(x,1)\right\rangle - y(x)\right)^2\right].$

Adjoint Equation

To optimize the Mean Field model, we calculate the gradient $\frac{\delta E}{\delta \rho}$ via the *adjoint sensitivity method*.

Model

The loss function can be written as

$$\mathbb{E}_{x\sim\mu} E(x;\rho) := \mathbb{E}_{x\sim\mu} \frac{1}{2} \left| \langle w_1, X_\rho(x,1) \rangle - y(x) \right|^2 \tag{9}$$

where X_ρ satisfies the equation $\dot{X}_\rho(x,t) = \int_\theta f(X_\rho(x,t),\theta)\rho(\theta,t)d\theta$,

Adjoint Equation

To optimize the Mean Field model, we calculate the gradient $\frac{\delta E}{\delta \rho}$ via the *adjoint sensitivity method*.

Model

The loss function can be written as

$$\mathbb{E}_{x\sim\mu}E(x;\rho) := \mathbb{E}_{x\sim\mu}\frac{1}{2}\left|\langle w_1, X_\rho(x,1)\rangle - y(x)\right|^2 \tag{9}$$

where X_ρ satisfies the equation $\dot{X}_\rho(x,t) = \int_\theta f(X_\rho(x,t),\theta)\rho(\theta,t)d\theta$,

Adjoint Equation. The gradient can be represented as a second backwards-in-time augmented ODE.

$$\begin{aligned}\dot{p}_\rho(x,t) &= -\delta_X H_\rho(p_\rho, x, t)\\ &= -p_\rho(x,t)\int \nabla_X f(X_\rho(x,t),\theta)\rho(\theta,t)d\theta,\end{aligned}$$

Here the Hamiltonian is defined as $H_\rho(p,x,t) = p(x,t)\cdot\int f(x,\theta)\rho(\theta,t)d\theta$.

Stanford University

Adjoint Equation

Theorem

For $\rho \in \mathcal{P}^2$ let $\frac{\delta E}{\delta \rho}(\theta, t) = \mathbb{E}_{x\sim\mu} f(X_\rho(x,t),\theta)) p_\rho(x,t)$, where p_ρ is the solution to the backward equation $\dot{p}_\rho(x,t) = -p_\rho(x,t) \int \nabla_X f(X_\rho(x,t),\theta)\rho(\theta,t)d\theta$. Then for every $\nu \in \mathcal{P}^2$, we have

$$E(\rho + \lambda(\nu - \rho)) = E(\rho) + \lambda \left\langle \frac{\delta E}{\delta \rho}, (\nu - \rho) \right\rangle + o(\lambda)$$

for the convex combination $(1-\lambda)\rho + \lambda\nu \in \mathcal{P}^2$ with $\lambda \in [0,1]$.

Adjoint equation is equivalent to the back propagation

Li Q, Chen L, Tai C, et al. Maximum principle based algorithms for deep learning. JMLR 2019
Zhang D, Zhang T, Lu Y, et al. You only propagate once: Painless adversarial training using maximal principle Neurips2019

Stanford University

Deep Residual Network Behaves Like an Ensemble Of Shallow Models

$$X^1 = X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0.$$

Deep Residual Network Behaves Like an Ensemble Of Shallow Models

$$X^1 = X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0.$$

$$X^2 = X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0 + \int_{\theta^1} \sigma(\theta^1(X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0))\rho^1(\theta^1)d\theta^1$$

$$= X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0 + \frac{1}{L}\int_{\theta^1} \sigma(\theta^1 X^0)\rho^1(\theta^1)d\theta^1 + \frac{1}{L^2}\int_{\theta_1} \nabla\sigma(\theta^1 X^0)\theta^1(\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0)\rho^1(\theta^1)d\theta^1$$

$$+ h.o.t.$$

Stanford University

Deep Residual Network Behaves Like an Ensemble Of Shallow Models

$$X^1 = X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0.$$

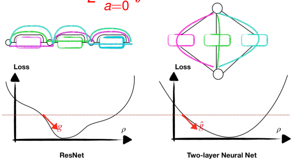
$$X^2 = X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0 + \int_{\theta^1} \sigma(\theta^1(X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0))\rho^1(\theta^1)d\theta^1$$

$$= X^0 + \frac{1}{L}\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0 + \frac{1}{L}\int_{\theta^1} \sigma(\theta^1 X^0)\rho^1(\theta^1)d\theta^1 + \frac{1}{L^2}\int_{\theta_1} \nabla\sigma(\theta^1 X^0)\theta^1(\int_{\theta^0} \sigma(\theta^0 X^0)\rho^0(\theta^0)d\theta^0)\rho^1(\theta^1)d\theta^1$$

$$+ h.o.t.$$

Iterating this expansion gives rise to

$$X^L \approx X^0 + \frac{1}{L}\sum_{a=0}^{L-1}\int \sigma(\theta X^0)\rho^a(\theta)d\theta + \frac{1}{L^2}\sum_{b>a}\int\int \nabla\sigma(\theta^b X^0)\theta^b\sigma(\theta^a X^0)\rho^b(\theta^b)\rho^a(\theta^a)d\theta^b\theta^a + h.o.t.$$



Veit A, Wilber M J, Belongie S. **Residual networks behave like ensembles of relatively shallow networks.** Advances in neural information processing systems. 2016: 550-558.

Stanford University

Deep Residual Network Behaves Like an Ensemble Of Shallow Models

Difference of back propagation process of two-layer net and ResNet.

Two-layer Network

$$\frac{\delta E}{\delta \rho}(\theta, t) = \mathbb{E}_{x\sim\mu} f(x, \theta))(X_\rho - y(x))$$

ResNet

$$\frac{\delta E}{\delta \rho}(\theta, t) = \mathbb{E}_{x\sim\mu} f(X_\rho(x, t), \theta))p_\rho(x, t)$$

We aim to show that the two gradient are similar.

Deep Residual Network Behaves Like an Ensemble Of Shallow Models

Difference of back propagation process of two-layer net and ResNet.

Two-layer Network

$$\frac{\delta E}{\delta \rho}(\theta, t) = \mathbb{E}_{x\sim\mu} f(x, \theta))(X_\rho - y(x))$$

ResNet

$$\frac{\delta E}{\delta \rho}(\theta, t) = \mathbb{E}_{x\sim\mu} f(X_\rho(x, t), \theta))p_\rho(x, t)$$

We aim to show that the two gradient are similar.

Lemma

The norm of the solution to the adjoint equation can be bounded by the loss

$$\|p_\rho(\cdot, t)\|_\mu \geq e^{-(C_1+C_2 r)} E(\rho), \forall t \in [0, 1]$$

,

Stanford University

Local = Global

Theorem

If $E(\rho) > 0$ for distribution $\rho \in \mathcal{P}^2$ that is supported on one of the nested sets Q_r, we can always construct a descend direction $\nu \in \mathcal{P}^2$, i.e.

$$\inf_{\nu \in \mathcal{P}^2} \left\langle \frac{\delta E}{\delta \rho}, (\nu - \rho) \right\rangle < 0$$

Local = Global

Theorem

If $E(\rho) > 0$ for distribution $\rho \in \mathcal{P}^2$ that is supported on one of the nested sets Q_r, we can always construct a descend direction $\nu \in \mathcal{P}^2$, i.e.

$$\inf_{\nu \in \mathcal{P}^2} \left\langle \frac{\delta E}{\delta \rho}, (\nu - \rho) \right\rangle < 0$$

Corollary

Consider a stationary solution to the Wasserstein gradient flow which is full support(informal), then it's a global minimizer.

Numerical Scheme

We may consider using a parametrization of ρ with n particles as

$$\rho_n(\theta, t) = \sum_{i=1}^{n} \delta_{\theta_i}(\theta) \mathbb{1}_{[\tau_i, \tau_i']}(t).$$

The characteristic function $\mathbb{1}_{[\tau_i, \tau_i']}$ can be viewed as a relaxation of the Dirac delta mass $\delta_{\tau_i}(t)$.

Given: A collection of residual blocks $(\theta_i, \tau_i)_{i=1}^n$
while training **do**
  Sort (θ_i, τ_i) based on τ_i to be (θ^i, τ^i) where $\tau^0 \leq \cdots \leq \tau^n$.
  Define the ResNet as $X^{\ell+1} = X^\ell + (\tau^\ell - \tau^{\ell-1})\sigma(^\ell X^\ell)$ for $0 \leq \ell < n$.
  Use gradient descent to update both θ^i and τ^i.
end while

Numerical Results

	Vanilla	mean-field	Dataset
ResNet20	8.75	8.19	CIFAR10
ResNet32	7.51	7.15	CIFAR10
ResNet44	7.17	6.91	CIFAR10
ResNet56	6.97	6.72	CIFAR10
ResNet110	6.37	6.10	CIFAR10
ResNet164	5.46	5.19	CIFAR10
ResNeXt29(864d)	17.92	17.53	CIFAR100
ResNeXt29(1664d)	17.65	16.81	CIFAR100

Table: Comparison of the stochastic gradient descent and mean-field training (Algorithm 1.) of ResNet On CIFAR Dataset. Results indicate that our method our performs the Vanilla SGD consistently.

Stanford University

Take Home Message

- We propose a new continuous limit for deep resnet

$$\dot{X}_\rho(x,t) = \int_\theta f(X_\rho(x,t),\theta)\rho(\theta,t)d\theta,$$

with initial $X_\rho(x,0) = \langle w_2, x\rangle$

- Local minimizer is global in ℓ_2 space.
- A potential scheme to approximate.

Take Home Message

- We propose a new continuous limit for deep resnet

$$\dot{X}_\rho(x,t) = \int_\theta f(X_\rho(x,t),\theta)\rho(\theta,t)d\theta,$$

with initial $X_\rho(x,0) = \langle w_2, x\rangle$

- Local minimizer is global in ℓ_2 space.
- A potential scheme to approximate.

TO DO List.

- Analysis of Wasserstein gradient flow. (Global Existence)
- Refined analysis of numerical scheme
- h.o.t in the expansion from ResNet to ensemble of small networks.

Stanford University

Outline Of The Talk

1 **Modeling**

2 **Optimization**
- Algorithm Design
- Theory

3 **Inferencing**

Inference

On Going

Thanks

Reference: Based on papers published at ICML2018, ICLR209, Neurips 2019 and ICML 2020.

Contact:yplu@stanford.edu