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

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

Presenter: Yipeng Lu
Contact: yplu@stanford.edu, https://web.stanford.edu/~yplu/
Deep Learning Evolution

- ILSVRC'15 ResNet: 3.57 layers
- ILSVRC'14 GoogleNet: 6.7 layers
- ILSVRC'14 VGG: 7.3 layers
- ILSVRC'13: 8 layers
- ILSVRC'12 AlexNet: 8 layers
- ILSVRC'11: 16.4 layers
- ILSVRC'10: 25.8 layers
- 152 layers

Stanford University
ODE As Infinite Depth Neural Network

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

First-order ODE
\[
\frac{dx}{dt} = F(x, t),
\]
\[
x(0) = x_0,
\]

Numerical solver: Euler’s method
\[
x_{i+1} = x_i + \gamma F(x_i, t_i),
\]
\[
x_0 \doteq x(0), x_i = x(\gamma t_i), \ldots \gamma - \text{step size}
\]

ResNet
\[
x_{i+1} = x_i + F(x_i, t_i),
\]

[He et al. 2015]
[E. 2017] [Haber et al. 2017]
[Lu et al. 2017] [Sho et al 2017] [Chen et al. 2018]
Outline

Modeling

Optimization

\[ \min L(\cdot) \]

Inferencing
Outline Of The Talk

1 Modeling

2 Optimization
   - Algorithm Design
   - Theory

3 Inferencing
Numerical Scheme As Architecture

First-order ODE

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

\[ x(0) = x_0, \]

Numerical solver: Euler’s method

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

\[ x_0 \doteq x(0), x_l = x(\gamma l), \ldots \gamma \text{ – step size} \]

ResNet

[He et al. 2015]

\[ x_{l+1} = x_l + F(x_l, t_l), \]

[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 design principled skip connection

All existing scheme are single step scheme.

Convergeto SDE

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

Convergence Requirement meets parameter selection.
Modeling Seq2Seq: Transformer

Understand Transformer as a multi-particle system.

\[
\frac{dx_i(t)}{dt} = \left( F(x_i(t), [x_1(t), \ldots, x_n(t)], t) + G(x_i(t), t), \right)
\]

Attention Layer

FFN Layer

\[\begin{align*}
    x_i(t_0) &= w_i, \quad i = 1, \ldots, n. \text{(Every words in a sentence)} \quad (1)
\end{align*}\]

Transformer is a splitting scheme, splitting \( F \) and \( G \).
Modeling Seq2Seq: Transformer

Understand Transformer as a multi-particle system.

\[
\frac{dx_i(t)}{dt} = F(x_i(t), [x_1(t), \ldots, x_n(t)], t) + G(x_i(t), t),
\]

Attention Layer

FFN Layer

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

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>IWSLT14 De-En</th>
<th>WMT14 En-De</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>small</td>
<td>base</td>
</tr>
<tr>
<td>Transformer</td>
<td>34.4</td>
<td>27.3</td>
</tr>
<tr>
<td>Weighted Transformer</td>
<td>/</td>
<td>28.4</td>
</tr>
<tr>
<td>Relative Transformer</td>
<td>/</td>
<td>26.8</td>
</tr>
<tr>
<td>Universal Transformer</td>
<td>/</td>
<td>28.9</td>
</tr>
<tr>
<td>Scaling NMT</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Dynamic Conv</td>
<td>35.2</td>
<td>/</td>
</tr>
<tr>
<td><strong>Macaron Net</strong></td>
<td>35.4</td>
<td>28.9</td>
</tr>
</tbody>
</table>
## Results

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

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


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

How can we encode the physic of task?
Our Example: DURR

How can we encode the physic of task?

\[
\begin{aligned}
\begin{cases}
    \frac{\partial u}{\partial t} &=  \text{div}(\sqrt{\varepsilon} \nabla u) \quad \text{in} \quad \Omega \times (0, T), \\
    \frac{\partial u}{\partial N} &= 0 \quad \text{on} \quad \partial \Omega \times (0, T), \\
    u(0, x) &= u_0(x) \quad \text{in} \quad \Omega,
\end{cases}
\end{aligned}
\]

Perona-Malik Equation
Our Example: DURR
Our Example: DURR

<table>
<thead>
<tr>
<th></th>
<th>BM3D</th>
<th>WNNM</th>
<th>DnCNN-B</th>
<th>UNLNet\textsubscript{5}</th>
<th>DURR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma = 25)</td>
<td>28.55</td>
<td>28.73</td>
<td>29.16</td>
<td>28.96</td>
<td>29.16</td>
</tr>
<tr>
<td>(\sigma = 35)</td>
<td>27.07</td>
<td>27.28</td>
<td>27.66</td>
<td>27.50</td>
<td>27.72</td>
</tr>
<tr>
<td>(\sigma = 55)</td>
<td>25.26</td>
<td>25.49</td>
<td>25.80</td>
<td>25.64</td>
<td>25.91</td>
</tr>
<tr>
<td>(\sigma = 65)</td>
<td>24.69</td>
<td>24.51</td>
<td>23.40*</td>
<td>-</td>
<td>25.26*</td>
</tr>
<tr>
<td>(\sigma = 75)</td>
<td>22.63</td>
<td>22.71</td>
<td>18.73*</td>
<td>-</td>
<td>24.71*</td>
</tr>
</tbody>
</table>
Modelling The Physics

Tycho Brahe: phenomenon
Johannes Kepler: discipline
Isaac Newton: Law

Our Work

[Images of scientists and a neural network diagram]

Stanford University
PDE-Net: Modeling The Physics

Consider a finite difference scheme for PDE:

\[
\begin{align*}
    u_{i\pm 1} &= \left[ u \pm h \partial_x u + \frac{h^2}{2} \partial_x^2 u \pm \frac{h^3}{6} \partial_x^3 u + \frac{h^4}{24} \partial_x^4 u \pm \frac{h^5}{120} \partial_x^5 u + \cdots \right]_i \\
\end{align*}
\]

Thus

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

\[\Delta u = u_{xx} + u_{yy}\]
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

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

(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 $K \setminus \{J + 1\}$.

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

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

(3)

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

2prime

ODE as Continuous Depth Neureral Networks 04/2020 16 / 43
PDE-Net: Recovering Coefficients
PDE-Net: Recovering Coefficients

Figure 15: First row: the true coefficients \( \{ f_{ij} : 1 \leq i + j \leq 2 \} \) of the equation. Second row: the learned coefficients \( \{ c_{ij} : 1 \leq i + j \leq 2 \} \) by the PDE-Net with 3 \( \delta t \)-blocks and \( 7 \times 7 \) filters.
Outline Of The Talk

1. Modeling

2. Optimization
   - Algorithm Design
   - Theory

3. Inferencing
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. \( x_{t+1} = f_t(x_t, \theta_t), \ t = 1, 2, \ldots, T - 1 \)
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. \hspace{1cm} x_{t+1} = f_t(x_t, \theta_t), \hspace{1cm} t = 1, 2, \ldots, T - 1 \hspace{1cm} (4)$$

Optimal Control:

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(x(T)) + \int_{0}^{T} R(x(t), \theta(t))dt$$

s.t. \hspace{1cm} \dot{x}(t) = f(x(t), \theta(t)) \hspace{1cm} (5)$$

$\theta(\cdot)$ is called a control
An Optimal Control View of Deep Learning

\[
\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(x(T)) + \int_0^T R(x(t), \theta(t)) dt
\]

s.t. \[ \dot{x}(t) = f(x(t), \theta(t)) \] (6)
An Optimal Control View of Deep Learning

$$\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(x(T)) + \int_0^T R(x(t), \theta(t))\,dt$$

s.t. $$\dot{x}(t) = f(x(t), \theta(t))$$

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

\[
\min_{\theta(\cdot)} J[\theta(\cdot)] = \ell(x(T)) + \int_0^T R(x(t), \theta(t)) dt
\]

s.t. \[ \dot{x}(t) = f(x(t), \theta(t)) \]

Gradient Based Training: Adjoint Equation

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

A New method? NO!

Adjoint Equation = Back Propagation!
An Optimal Control View of Deep Learning

\[
\min_{\theta(\cdot)} J[\theta(\cdot)] = \mathcal{L}(x(T)) + \int_0^T R(x(t), \theta(t)) \, dt \\
\text{s.t.} \quad \dot{x}(t) = f(x(t), \theta(t))
\]

(6)

Gradient Based Training: Adjoint Equation

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

A New method? NO!

**Adjoint Equation = Back Propagation!**

Benefit:

- **Invertible**: Neural Ordinary Differential Equation Neurips2018.
- **Find out structure!** (Our work)
Outline Of The Talk

1 Modeling

2 Optimization
   ▪ Algorithm Design
   ▪ Theory

3 Inferencing
Adversarial Training

**Robust Optimization**

\[
\min_{\theta} \mathbb{E}_{(x,y) \sim D} \max_{\|\eta\| \leq \epsilon} \ell(\theta; x + \eta, y),
\]
Adversarial Training

Robust Optimization

\[ \min_{\theta} \mathbb{E}_{(x,y) \sim D} \max \ell(\theta; x + \eta, y), \quad \|\eta\| \leq \epsilon \]

PGD Method

- Gradient ascent on \( x \).
  \[ x^{t+1} = \prod_{x+S} (x^t + \alpha \text{sign}(\nabla_x \ell)) \]
  for \( r \) times.

- Gradient Descent On \( \theta \).
  \[ \theta = \theta - \nabla_\theta \ell \]
  for 1 times.
**Our Intuition: Splitting The Gradient**

**YOPO (You Only Propogate Once)**

1. initialize perturbation $\eta$

2. for $k = 1$ to $m$ do

   3. $p \leftarrow \nabla f_0 \ell(x + \eta)$

   4. for $i = 1$ to $n$ do

      5. $\eta \leftarrow \eta + \alpha \cdot p \cdot \nabla_x f_0 (x + \eta)$

   6. end for

7. accumulate gradient $U \leftarrow U + \nabla_\theta \ell(x + \eta)$

8. end for

9. Use $U$ tp perform SGD / momentum SGD

$m$ times full backprop.

Focus on first layer.

splitting

use intermediate adversarial examples
A Differential Game View of Adversarial Training

Adversarial Training:

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

subject to:

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

Differential Game:

$$\min_{\theta(\cdot)} \max_{\eta(\cdot)} J[\theta(\cdot), \eta(\cdot)] = \ell(x(T)) + \int_0^T R(x(t), \theta(t), \eta(t)) dt$$  \hspace{1cm} (8)

subject to:

$$\dot{x}(t) = f(x(t), \theta(t), \eta(t))$$

Differential game is optimal control with 2 controls, each having opposite target.
YOPO: An Optimal Control View

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

Define Hamiltonian $H(x, p, \theta, \eta) := p \cdot f(x, \theta, \eta) + r(x, \theta, \eta)$

PMP for differential game tells us there exists an adjoint dynamic $p$ satisfying:

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

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

$H(x^* (t), p^* (t), \theta^* (t), \eta^* (t)) \geq H(x^* (t), p^* (t), \theta^* (t), \eta^*) \geq H(x^* (t), p^* (t), \theta, \eta^* (t))$, $\forall t, \eta, \theta$
YOPO: An Optimal Control View

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

Define Hamiltonian

\[ H(x, p, \theta, \eta) := p \cdot f(x, \theta, \eta) + r(x, \theta, \eta) \]

PMP for differential game tells us there exists an adjoint dynamic \( p(\cdot) \) satisfying:

\[
\begin{align*}
\dot{x}^*(t) &= \nabla_p H(x^*(t), p^*(t), \theta^*(t), \eta^*(t)) \\
\dot{p}^*(t) &= -\nabla_x H(x^*(t), p^*(t), \theta^*(t), \eta^*(t)) \\
H(x^*(t), p^*(t), \theta^*(t), \eta) &\geq H(x^*(t), p^*(t), \theta^*(t), \eta^*(t)) \\
&\geq H(x^*(t), p^*(t), \theta, \eta^*(t)), \quad \forall t, \eta, \theta
\end{align*}
\]
YOPO: An Optimal Control View

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

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

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

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

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

\[
H(x^*(t), p^*(t), \theta^*(t), \eta) \geq H(x^*(t), p^*(t), \theta^*(t), \eta^*(t))
\]

\[
\geq H(x^*(t), p^*(t), \theta, \eta^*(t)), \quad \forall t, \eta, \theta
\]

Parameter \( \theta, \eta \) should optimize the Hamiltonian. \( \eta(0) \) only coupled with the first layer.
Decoupled Training

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


## CIFAR10 WideResNet34 Results

<table>
<thead>
<tr>
<th>Training Methods</th>
<th>Clean Data</th>
<th>PGD-20 Attack</th>
<th>Training Time (mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural train</td>
<td>95.03%</td>
<td>0.00%</td>
<td>233</td>
</tr>
<tr>
<td>PGD-3</td>
<td>90.07%</td>
<td>39.18%</td>
<td>1134</td>
</tr>
<tr>
<td>PGD-5</td>
<td>89.65%</td>
<td>43.85%</td>
<td>1574</td>
</tr>
<tr>
<td>PGD-10</td>
<td>87.30%</td>
<td>47.04%</td>
<td>2713</td>
</tr>
<tr>
<td>Free-8 ¹</td>
<td>86.29%</td>
<td>47.00%</td>
<td>667</td>
</tr>
<tr>
<td>YOPO-3-5 (Ours)</td>
<td>87.27%</td>
<td>43.04%</td>
<td>299</td>
</tr>
<tr>
<td>YOPO-5-3 (Ours)</td>
<td>86.70%</td>
<td>47.98%</td>
<td>476</td>
</tr>
</tbody>
</table>

Table: Results of Wide ResNet34 for CIFAR10.
Take Home Message

Bridging

- Adversarial Training
- Differential Game.

YOPO (You Only Propagate Once)

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

2. Use intermediate perturbation to update weights

YOPO can be understood as discretization way solving PMP

Continuous Depth Neural Networks:
Take Home Message

Bridging
- Adversarial Training
- Differential Game.

YOPO (You Only Propogate Once)

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

- Adversarial Training
- Differential Game.

**YOPO (You Only Propogate Once)**

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 $\theta$
Take Home Message

Bridging

- Adversarial Training
- Differential Game.

**YOPO** *(You Only Propogate Once)*

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 $\theta$

YOPO can be understood as

**discretization way solving PMP**
Outline Of The Talk

1. Modeling

2. Optimization
   - Algorithm Design
   - Theory

3. Inferencing
Global Convergence Proof Of NN

- Neural Tangent Kernel ([Jacot et al. 2019]): Linearize the model
  \[ f_{\text{NN}}(\theta) = f_{\text{NN}}(\theta_{\text{init}}) + \langle \nabla_{\theta} f_{\text{NN}}(\theta_{\text{init}}), \theta - \theta_{\text{init}} \rangle \]
Global Convergence Proof Of NN

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

\[
L(\rho) = \| \mathbb{E}_{\theta \sim \rho} g(\theta, x) - f(x) \|^2
\]

- **Pro:** SGD = Wasserstein Gradient Flow ([Mei et al. 2018][Chizat et al. 2018][Rotskoff et al. 2018])
- **Con:** Hard to generalize beyond two layer ODE as Continuous Depth Neural Networks:
Global Convergence Proof Of NN

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

  - **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
  \[ L(\rho) = \| \mathbb{E}_{\theta \sim \rho} g(\theta, x) - f(x) \|^2 \]
  the objective is a convex function
Global Convergence Proof Of NN

- Neural Tangent Kernel ([Jacot et al. 2019]): Linearize the model
  \[ f_{NN}(\theta) = f_{NN}(\theta_{init}) + \langle \nabla_{\theta} f_{NN}(\theta_{init}), \theta - \theta_{init} \rangle \]
  - **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
Mean Field ResNet

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

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: \( E(\rho) = \mathbb{E}_{x \sim \mu} \left[ \frac{1}{2} \left( \langle w_1, X_\rho(x, 1) \rangle - y(x) \right)^2 \right] \).
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

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

where $X_\rho$ 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

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

(9)

where $X_\rho$ 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.

$$\dot{p}_\rho(x, t) = -\delta_X H_\rho(p_\rho, x, t)$$

$$= -p_\rho(x, t) \int \nabla_X f(X_\rho(x, t), \theta) \rho(\theta, t) \, d\theta,$$

Here the Hamiltonian is defined as $H_\rho(p, x, t) = p(x, t) \cdot \int f(x, \theta) \rho(\theta, t) \, d\theta$. 
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 \( p_{\rho} \) 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
Deep Residual Network Behaves Like an Ensemble Of Shallow Models

\[ x^1 = x^0 + \frac{1}{L} \int_{\theta^0} \sigma(\theta^0 x^0) \rho(\theta^0) d\theta^0. \]

Deep Residual Network Behaves Like an Ensemble Of Shallow Models

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

\[ X^2 = X^0 + \frac{1}{L} \int_{\theta^0} \sigma(\theta^0 X^0) \rho^0(\theta^0) d\theta^0 + \int_{\theta^1} \sigma(\theta^1 X^0) \rho^1(\theta^1) d\theta^1 + \frac{1}{L} \int_{\theta^0} \sigma(\theta^0 X^0) \rho^0(\theta^0) d\theta^0 \int_{\theta^0} \sigma(\theta^0 X^0) \rho^0(\theta^0) d\theta^0 \rho^1(\theta^1) d\theta^1 + h.o.t. \]
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

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

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

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

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

The characteristic function $\mathbb{1}_{\left[\tau_i, \tau'_i\right]}$ 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 $(\theta_i, \tau_i)$ based on $\tau_i$ to be $(\theta^i, \tau^i)$ 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 $\theta^i$ and $\tau^i$.
end while
### Numerical Results

<table>
<thead>
<tr>
<th></th>
<th>Vanilla</th>
<th>mean-field</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet20</td>
<td>8.75</td>
<td>8.19</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet32</td>
<td>7.51</td>
<td>7.15</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet44</td>
<td>7.17</td>
<td>6.91</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet56</td>
<td>6.97</td>
<td>6.72</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet110</td>
<td>6.37</td>
<td>6.10</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet164</td>
<td>5.46</td>
<td>5.19</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNeXt29(864d)</td>
<td>17.92</td>
<td>17.53</td>
<td>CIFAR100</td>
</tr>
<tr>
<td>ResNeXt29(1664d)</td>
<td>17.65</td>
<td>16.81</td>
<td>CIFAR100</td>
</tr>
</tbody>
</table>

**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.
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 \( \ell_2 \) 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 \( \ell_2 \) 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.
Outline Of The Talk

1. Modeling

2. Optimization
   - Algorithm Design
   - Theory

3. Inferencing
Inference

On Going
Thanks


Contact: yplu@stanford.edu