Dynamic System and Optimal Control Perspective of Deep Learning

BIN DONG
PEKING UNIVERSITY

Special thanks to Yiping Lu who helped in preparation of the slides.
Outline

- Background and motivation
- Deep neural network and numerical ODE
- Deep neural network and numerical PDE
- An application in image processing and medical imaging
- Optimal control perspective for deep network training
Background & Motivation
Deep Learning: Burning Hot!

Credit: D. Donoho/ H. Monajemi/ V. Panyan “Stats 385” @Stanford
Deep Learning

Deep learning is “alchemy”
- Ali Rahimi, NIPS 2017
Deep Learning

What are still challenging

◦ Learning from limited or/and weakly labelled data
◦ Learning from data of different types
◦ Theoretical guidance, transparency

Should we expect rigorous mathematical analysis of deep learning? Maybe, but…

*We also wish to allow the possibility than an engineer or team of engineers may construct a machine which works, but whose manner of operation cannot be satisfactorily described by its constructors because they have applied a method which is largely experimental* – Alan M. Turing
Deep Learning

What are still challenging
- Learning from limited or/and weakly labelled data
- Learning from data of different types
- Theoretical guidance, transparency

We probably should first find “frameworks” and “links” with mathematics.

- Deep Network ↔ Differential Equations (DE)
- Network Architecture ↔ Numerical DE
- Network Training ↔ Optimal Control
Deep Neural Networks and Numerical ODE

NETWORK STRUCTURE DESIGN
Depth Neural Network

Deep Neural Network

\[ f_1 \left( f_2 \left( f_3 \cdots (x) \right) \right) \]

A Dynamic System?
Motivation

Deep Residual Learning (@CVPR2016)

\[ x_{n+1} = x_n + f(x_n) \]

Forward Euler Scheme

\[ x_t = f(x) \]

- Bo C, Meng L, et al. Reversible Architectures for Arbitrarily Deep Residual Neural Networks, AAAI 2018
Motivation

Deep Residual Learning (@CVPR2016)

Theoretical Convergence Results is built in:

A New Generalization Perspective From Control:
Depth Revolution

Deeper And Deeper
Depth Revolution

Going into infinite layer

Differential Equation As Infinite Layer Neural Network
Revisiting previous efforts in deep learning, we found that diversity, another aspect in network design that is relatively less explored, also plays a significant role.

**PolyStrure:** \( x_{n+1} = x_n + F(x_n) + F(F(x_n)) \)

**Backward Euler Scheme:**

\[
x_{n+1} = x_n + F(x_{n+1}) \Rightarrow x_{n+1} = (I - F)^{-1}x_n
\]

Approximate the operator \((I - F)^{-1}\) by \(I + F + F^2 + \cdots\)
FractalNet (@ICLR2017)

\[ x_{n+1} = k_1 x_n + k_2 (k_3 x_n + f_1(x_n)) + f_2 (k_3 x_n + f_1(x_n)) \]

ODE: Infinite Layer Neural Network

Dynamic System
Continuous limit

Neural Network
Numerical Approximation

Table 1: In this table, we list a few popular deep networks, their associated ODEs and the numerical schemes that are connected to the architecture of the networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>Related ODE</th>
<th>Numerical Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet, ResNeXt, etc.</td>
<td>$u_t = f(u)$</td>
<td>Forward Euler scheme</td>
</tr>
<tr>
<td>PolyNet</td>
<td>$u_t = f(u)$</td>
<td>Approximation of backward Euler scheme</td>
</tr>
<tr>
<td>FractalNet</td>
<td>$u_t = f(u)$</td>
<td>Runge-Kutta scheme</td>
</tr>
<tr>
<td>RevNet</td>
<td>$\dot{X} = f_1(Y), \dot{Y} = f_2(X)$</td>
<td>Forward Euler scheme</td>
</tr>
</tbody>
</table>

WRN, ResNeXt, Inception-ResNet, PolyNet, SENet etc...... :
New scheme to Approximate the right hand side term

Why not change the way to discrete $u_t$?

Experiment

@Linear Multi-step Residual Network

\[ x_t = f(x) \]

\[ x_{n+1} = x_n + f(x_n) \]

Experiment

@Linear Multi-step Residual Network

$$x_{n+1} = x_n + f(x_n)$$

Linear Multi-step Scheme

$$x_{n+1} = (1 - k_n)x_n + k_n x_{n-1} + f(x_n)$$

Experiment

@Linear Multi-step Residual Network

(a) Resnet
(b) LM-Resnet

Table 2: Comparisons of LM-ResNet/LM-ResNeXt with other networks on CIFAR

<table>
<thead>
<tr>
<th>Model</th>
<th>Layer</th>
<th>Error</th>
<th>Params</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet (He et al. 2015b)</td>
<td>20</td>
<td>8.75</td>
<td>0.27M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet (He et al. 2015b)</td>
<td>32</td>
<td>7.51</td>
<td>0.46M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet (He et al. 2015b)</td>
<td>44</td>
<td>7.17</td>
<td>0.66M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>ResNet (He et al. 2015b)</td>
<td>56</td>
<td>6.97</td>
<td>0.85M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>110, pre-act</td>
<td>6.37</td>
<td>1.7M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>20, pre-act</td>
<td>8.33</td>
<td>0.27M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>32, pre-act</td>
<td>7.18</td>
<td>0.46M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>44, pre-act</td>
<td>6.66</td>
<td>0.66M</td>
<td>CIFAR10</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>56, pre-act</td>
<td>6.31</td>
<td>0.85M</td>
<td>CIFAR10</td>
</tr>
</tbody>
</table>

# Experiment

@Linear Multi-step Residual Network

<table>
<thead>
<tr>
<th>Model</th>
<th>Layer</th>
<th>Accuracy</th>
<th>Params</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnet</td>
<td>20</td>
<td>91.25</td>
<td>0.27M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>Resnet</td>
<td>32</td>
<td>92.49</td>
<td>0.46M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>Resnet</td>
<td>44</td>
<td>92.83</td>
<td>0.66M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>Resnet</td>
<td>56</td>
<td>93.03</td>
<td>0.85M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>Resnet</td>
<td>110</td>
<td>93.63</td>
<td>1.7M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>LM-ResNet(Ours)</td>
<td>20</td>
<td>91.67</td>
<td>0.27M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>LM-ResNet(Ours)</td>
<td>32</td>
<td>92.82</td>
<td>0.46M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>LM-ResNet(Ours)</td>
<td>44</td>
<td>92.98</td>
<td>0.66M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>LM-ResNet(Ours)</td>
<td>56</td>
<td>93.69</td>
<td>0.85M</td>
<td>Cifar10</td>
</tr>
<tr>
<td>EM-ResNet(Ours)</td>
<td>40</td>
<td>91.75</td>
<td>0.27M</td>
<td>Cifar10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Layer</th>
<th>top-1</th>
<th>top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet (He et al. (2015b))</td>
<td>50</td>
<td>24.7</td>
<td>7.8</td>
</tr>
<tr>
<td>ResNet (He et al. (2015b))</td>
<td>101</td>
<td>23.6</td>
<td>7.1</td>
</tr>
<tr>
<td>ResNet (He et al. (2015b))</td>
<td>152</td>
<td>23.0</td>
<td>6.7</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>50, pre-act</td>
<td>23.8</td>
<td>7.0</td>
</tr>
<tr>
<td>LM-ResNet (Ours)</td>
<td>101, pre-act</td>
<td><strong>22.6</strong></td>
<td><strong>6.4</strong></td>
</tr>
</tbody>
</table>
Explanation on the performance boost via modified equations

@Linear Multi-step Residual Network

**ResNet**

\[ x_{n+1} = x_n + \Delta t f(x_n) \]

**LM-ResNet**

\[ x_{n+1} = (1 - k_n)x_n + k_n x_{n-1} + \Delta t f(x_n) \]

\[ \dot{u} + \frac{\Delta t}{2} \ddot{u}_n = f(u) \]

\[ (1 + k_n) \dot{u} + (1 - k_n) \frac{\Delta t}{2} \ddot{u}_n = f(u) \]


Plot The Momentum

@Linear Multi-step Residual Network

\[ x_{n+1} = (1 - k_n)x_n + k_n x_{n-1} + \Delta t f(x_n) \]

Learn A Momentum

\[ (1 + k_n) \ddot{u} + (1 - k_n) \frac{\Delta t}{2} \dot{u}_n + o(\Delta t^3) = f(u) \]
Plot The Momentum

@Linear Multi-step Residual Network

\[ x_{n+1} = (1 - k_n)x_n + k_n x_{n-1} + \Delta t f(x_n) \]

Learn A Momentum

\[ (1 + k_n) \dot{u} + (1 - k_n) \frac{\Delta t}{2} \ddot{u}_n + o(\Delta t^3) = f(u) \]
Connection to stochastic dynamic

Noise can avoid overfit?

Dynamic System

Connection to stochastic dynamic

Shake-Shake regularization

\[ x_{n+1} = x_n + \eta f_1(x) + (1 - \eta) f_2(x), \eta \sim U[0, 1] \]

\[ = x_n + f_2(x_n) + \frac{1}{2} (f_1(x_n) - f_2(x_n)) + (\eta - \frac{1}{2}) (f_1(x_n) - f_2(x_n)) \]

Apply data augmentation techniques to internal representations.

Figure 1: Left: Forward training pass. Center: Backward training pass. Right: At test time.


Connection to stochastic dynamic

Deep Networks with Stochastic Depth

\[ x_{n+1} = x_n + \eta_n f(x) \]
\[ = x_n + E\eta_n f(x_n) + (\eta_n - E\eta_n) f(x_n) \]

To reduce the effective length of a neural network during training, we randomly skip layers entirely.

\[ \sqrt{p(t)(1-p(t))} f(X) \odot [1_{N \times 1}, 0_{N,N-1}] dB_t. \]

Fig. 2. The linear decay of \( p_t \) illustrated on a ResNet with stochastic depth for \( p_0 = 1 \) and \( p_L = 0.5 \). Conceptually, we treat the input to the first ResBlock as \( H_0 \), which is always active.


Noise can avoid overfit?

\[ \dot{X}(t) = f(X(t), a(t)) + g(X(t), t)dB_t, X(0) = X_0 \]

The numerical scheme is only need to be **weak convergence**!

\[ E_{\text{data}}(\text{loss}(X(T))) \]

---

Deep Networks with Stochastic Depth

\[ x_{n+1} = x_n + \eta_n f(x) \]

\[ = x_n + E\eta_n f(x_n) + (\eta_n - E\eta_n) f(x_n) \]

We need \( 1 - 2p_n = O(\sqrt{\Delta t}) \)

To reduce the effective length of a neural network during training, we randomly skip layers entirely.


(a) ResNet

(b) Linear Multi-step ResNet


\[(1 + k_n) \ddot{u} + (1 - k_n) \frac{\Delta t}{2} \dot{u}_n + o(\Delta t^3) = f(u) + g(u)dW_t\]
Experiment

@Linear Multi-step Residual Network

<table>
<thead>
<tr>
<th>Model</th>
<th>Layer</th>
<th>Training Strategy</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet(He et al. (2015b))</td>
<td>110</td>
<td>Original</td>
<td>6.61</td>
</tr>
<tr>
<td>ResNet(He et al. (2016b))</td>
<td>110.pre-act</td>
<td>Original</td>
<td>6.37</td>
</tr>
<tr>
<td>ResNet(Huang et al. (2016b))</td>
<td>56</td>
<td>Stochastic depth</td>
<td>5.66</td>
</tr>
<tr>
<td>ResNet(Our Implement)</td>
<td>56.pre-act</td>
<td>Stochastic depth</td>
<td>5.55</td>
</tr>
<tr>
<td>ResNet(Huang et al. (2016b))</td>
<td>110</td>
<td>Stochastic depth</td>
<td>5.25</td>
</tr>
<tr>
<td><strong>ResNet(Huang et al. (2016b))</strong></td>
<td>1202</td>
<td>Stochastic depth</td>
<td><strong>4.91</strong></td>
</tr>
<tr>
<td>ResNet(Ours)</td>
<td>110.pre-act</td>
<td>Gaussian noise (noise level = 0.001)</td>
<td>5.52</td>
</tr>
<tr>
<td>LM-ResNet(Ours)</td>
<td>56.pre-act</td>
<td>Stochastic depth</td>
<td>5.14</td>
</tr>
<tr>
<td><strong>LM-ResNet(Ours)</strong></td>
<td>110.pre-act</td>
<td>Stochastic depth</td>
<td><strong>4.80</strong></td>
</tr>
</tbody>
</table>

Conclusion

@Beyond Finite Layer Neural Network

Earlier Evidence: LISTA

Unrolled Dynamics

\[ Z(k + 1) = h_\theta(W_e X + SZ(k)), \quad Z(0) = 0 \]

ISTA

\[ X \xrightarrow{W_e} \xrightarrow{+} \xrightarrow{S} \xrightarrow{\theta} \]

Unrolling

LISTA

\[ X \xrightarrow{W_e} \xrightarrow{S} \xrightarrow{+} \xrightarrow{S} \xrightarrow{+} \xrightarrow{S} \xrightarrow{+} \xrightarrow{S} \xrightarrow{+} \xrightarrow{S} \xrightarrow{+} \xrightarrow{\theta} \]

Earlier Evidence: TRD

Unrolled Dynamics

Learning a diffusion process for denoising

Average PSNR among a dataset with 68 images
Recent Evidence: Optimization Algorithm Inspired DNN

- Deep neural network as optimization algorithm:
  \[ x_{k+1} = \phi(Wx_k) \quad \text{if} \quad x_{k+1} = x_k - \nabla F(x_k) \]

- Faster algorithm result in better deep neural network:

  **Heavy Ball Net:**
  \[ x_{k+1} = T(x_k) + x_k - x_{k-1} \]

  **Accelerated GD Net:**
  \[ x_{k+1} = \sum_{j=0}^{k} \alpha_{k+1,j} T(x_j) + \beta \left( x_k - \sum_{j=0}^{k} h_{k+1,j} x_j \right) \]

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet ((n = 9))</td>
<td>10.05</td>
<td>39.65</td>
</tr>
<tr>
<td>HB-Net ((16) (n = 9))</td>
<td>10.17</td>
<td>38.52</td>
</tr>
<tr>
<td>ResNet ((n = 18))</td>
<td>9.17</td>
<td>38.13</td>
</tr>
<tr>
<td>HB-Net ((16) (n = 18))</td>
<td>8.66</td>
<td>36.4</td>
</tr>
<tr>
<td>DenseNet ((k = 12, L = 40)*)</td>
<td>7</td>
<td>27.55</td>
</tr>
<tr>
<td>AGD-Net ((18) (k = 12, L = 40))</td>
<td>6.44</td>
<td>26.33</td>
</tr>
<tr>
<td>DenseNet ((k = 12, L = 52))</td>
<td>6.05</td>
<td>26.3</td>
</tr>
<tr>
<td>AGD-Net ((18) (k = 12, L = 52))</td>
<td>5.75</td>
<td>24.92</td>
</tr>
</tbody>
</table>

Recent Evidence: Nonlocal DNN

- "Kinetics" data set: 246k training videos and 20k validation videos.
- Task: classification involving 400 human action categories

Residual Block: $Z^{k+1} := Z^k + \mathcal{F}(Z^k; W^k)$

ResNet Block: $\mathcal{F}(Z^k; W^k) = W_2^k f(W_1^k f(Z^k)), \quad f = \text{ReLU} \circ \text{BN}$

Nonlocal Block: $[\mathcal{F}(Z^k; W^k)]_i = \frac{W_2^k}{C_i(Z^k)} \sum_{j} \omega(Z_i^k, Z_j^k) (W_y^k Z_j^k)$

(c) **Deeper non-local models**: we compare 1, 5, and 10 non-local blocks added to the C2D baseline. We show ResNet-50 (top) and ResNet-101 (bottom) results.

Instability when using multiple blocks!
Recent Evidence: Nonlocal DNN as Nonlocal Diffusion

Design a new **stable** block

\[ Z_{i}^{n+1} := Z_{i}^{n} + \frac{W^n}{C_i(X)} \sum_{j} \omega(X_i, X_j)(Z_j^{n} - Z_i^{n}) \]

<table>
<thead>
<tr>
<th>Model</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>8.19</td>
</tr>
<tr>
<td>2-block (original)</td>
<td>7.83</td>
</tr>
<tr>
<td>3-block (original)</td>
<td>8.28</td>
</tr>
<tr>
<td>4-block (original)</td>
<td>15.02</td>
</tr>
<tr>
<td>Same Place</td>
<td></td>
</tr>
<tr>
<td>2-block (proposed)</td>
<td>7.74</td>
</tr>
<tr>
<td>3-block (proposed)</td>
<td>7.62</td>
</tr>
<tr>
<td>4-block (proposed)</td>
<td>7.37</td>
</tr>
<tr>
<td>5-block (proposed)</td>
<td><strong>7.29</strong></td>
</tr>
<tr>
<td>6-block (proposed)</td>
<td>7.55</td>
</tr>
<tr>
<td>Different Places</td>
<td></td>
</tr>
<tr>
<td>3-block (original)</td>
<td>8.07</td>
</tr>
<tr>
<td>3-block (proposed)</td>
<td><strong>7.33</strong></td>
</tr>
</tbody>
</table>
Deep Neural Networks and Numerical PDE

DATA DRIVEN PHYSIC LAW DISCOVERY
PDE-Net: Learning PDEs from Data

Can we learn principles (e.g. PDEs) from data?

Dynamics of actin in Immunocytoskeleton

Dynamics of Mitochondria

Credit: Kebin Shi, Physics@PKU

PDE-Net: Learning PDEs from Data

Can we learn principles (e.g. PDEs) from data?

Preliminary attempt:
- Combine deep learning and numerical PDEs

Objectives:
- Predictive power (deep learning)
- Transparency (numerical PDEs)

PDE-Net: Learning PDEs from Data

PDE-Net: a flexible and transparent deep network

Assuming:
\[
\frac{\partial u}{\partial t} = F(x, u, \nabla u, \nabla^2 u, \ldots)
\]

Prior knowledge on $F$:
- Type of the PDE
- Maximum order

Constraints on kernels (granting transparency)

- Moment matrix (related to vanishing moments in wavelets)

\[ M(q) = (m_{i,j})_{N \times N}, \text{ where } m_{i,j} = \frac{1}{(i-1)!(j-1)!} \sum_{k \in \mathbb{Z}^2} k_1^{i-1} k_2^{j-1} q[k_1, k_2] \]

- We can approximate any differential operator at any prescribed order by constraining \( M(q) \)

- For example: approximation of \( \frac{\partial f}{\partial x} \) with a 3 × 3 kernel

\[
\begin{pmatrix}
0 & 0 & \star \\
1 & \star & \star \\
\star & \star & \star \\
\end{pmatrix} \sim 
\begin{pmatrix}
0 & 0 & 0 \\
1 & 0 & \star \\
0 & \star & \star \\
\end{pmatrix} \sim 
\begin{pmatrix}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
\end{pmatrix}
\]

1st order learnable 2nd order learnable 1st order frozen

Dong, Q. Jiang and Z. Shen, Multiscale Modeling & Simulation, 2017
PDE-Net: Learning PDEs from Data

Numerical experiments: data set generation
- Convection-diffusion equation (linear)
  \[
  \begin{align*}
  \frac{\partial u}{\partial t} &= a(x, y)u_x + b(x, y)u_y + cu_{xx} + du_{yy} \\
  u|_{t=0} &= u_0(x, y),
  \end{align*}
\]
  \[a(x, y) = 0.5(\cos(y) + x(2\pi - x)\sin(x)) + 0.6,\]
  \[b(x, y) = 2(\cos(y) + \sin(x)) + 0.8,\]
  \[c = 0.2\text{ and } d = 0.3\]
- Diffusion with a nonlinear source (nonlinear)
  \[
  \begin{align*}
  \frac{\partial u}{\partial t} &= c\Delta u + f_s(u) \\
  u|_{t=0} &= u_0(x, y),
  \end{align*}
\]
  \[c = 0.3\text{ and } f_s(u) = 15\sin(u)\]
- Initialization: random function with frequency \(\leq 9\) and \(6\)
- Assumptions on \(F\)
  - Linear:
    \[F = \sum_{0\leq i+j\leq 4} f_{ij}(x, y) \frac{\partial^{i+j} u}{\partial x^i \partial y^j}\]
  - Nonlinear
    \[F = \sum_{1\leq i+j\leq 2} f_{ij}(x, y) \frac{\partial^{i+j} u}{\partial x^i \partial y^j} + f_s(u)\]

PDE-Net: Learning PDEs from Data

Numerical experiments: results
- Prediction: linear (5 × 5 and 7 × 7 filters)

Learnable filters (orange) v.s. frozen filters (blue) in prediction

PDE-Net: Learning PDEs from Data

Numerical experiments: results

- Model estimation: linear

PDE-Net: Learning PDEs from Data

Numerical experiments: results

- Prediction and model estimation: nonlinear (7 × 7 filters)

PDE-Net 2.0: Numeric-Symbolic Hybrid Representation

Symbolic network (granting transparency)

Assuming: \( \frac{\partial u}{\partial t} = F(u, \nabla u, \nabla^2 u, \ldots) \)

Prior knowledge on \( F \):
- Addition and multiplication of the terms;
- Maximum order.

\[ \bar{u}_t \]

\[ \bar{u}_{t+\delta t} = \bar{u}_t + \delta t \cdot \bar{F} \]
PDE-Net 2.0: Numeric-Symbolic Hybrid Representation

Symbolic network (granting transparency)

More Constraints:
- Pseudo-upwinding
- Sparsity on moment matrices
- Sparsity on the symbolic network

Motivated by EQL
PDE-Net 2.0: Numeric-Symbolic Hybrid Representation

Weaker assumption on $F$: unknown type

| Correct PDE | $u_t = -u_{xx} - u_{yy} + 0.05(u_{xx} + u_{yy})$
| Frozen-PDE-Net 2.0 | $u_t = -0.906 u_{xx} - 0.901 u_{yy} + 0.033 u_{xxx} + 0.037 u_{xyy}$
| PDE-Net 2.0 | $u_t = -0.986 u_{xx} - 0.972 u_{yy} + 0.054 u_{xxx} + 0.052 u_{xyy}$

Burger’s Equation

$$\partial_t u + (u \cdot \nabla) u = \nu \nabla^2 u$$

$$\nu = 0.05$$

Application In Image Processing

BLIND IMAGE RESTORATION
Deep Learning For Restoration

One Noise Level One Net

Network 1 \( \sigma = 25 \)
Deep Learning For Restoration

One Noise Level One Net

Network 1 \( \sigma = 25 \)

Network 2 \( \sigma = 35 \)
What We Want

One Model For All Noise Level
What Happen When Meet High Noise Level

Fails!
PDEs In Image Processing

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \text{div} \left( c(|\nabla u|^2) \nabla u \right) \quad \text{in} \quad \Omega \times (0, T), \\
\frac{\partial u}{\partial N} &= 0 \quad \text{on} \quad \partial \Omega \times (0, T), \\
u(0, x) &= u_0(x) \quad \text{in} \quad \Omega,
\end{align*}
\]

Moving Endpoint Control
Terminal time as a variable to train

Early Stopping Is A Regularization
Can we train it?

Need to be learn

\[
\min_{w, \tau} L(X(\tau), y) + \int_0^\tau R(w(t), t) dt \\
\text{s.t. } \dot{X} = f(X(t), w(t)), t \in (0, \tau) \\
X(0) = x_0.
\]

Our Approach: Dynamically Unfolding Recurrent Restorer

A Good Policy Leads To A Good Restorer

Given A Policy -> Train The Restorer
- Good Policy Leads To Better Restorer
- Good Policy Leads To Better Generalization

Table 1: Average peak PSNR on BSD68 with different training strategies.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Average peak PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Noise</td>
<td>25</td>
</tr>
<tr>
<td>35, 45 Naive</td>
<td>27.74</td>
</tr>
<tr>
<td>35, 45 Refined</td>
<td>29.14</td>
</tr>
</tbody>
</table>

DURR Model
Discretize: Turn To An RL Problem

\[ \min_{w,N_i(i=1,2,\ldots,d)} \sum_{i=1}^{d} \sum_{j=1}^{N_i} R_j(w_j)dt + \lambda I(X_{N_i}^i, f_i) \]

\[ s.t. X_n^i = X_{n-1}^i + \Delta t f(X_{n-1}^i, w(t)), n = 1, 2 \ldots, N_i, (i = 1, 2, \ldots, d) \]

\[ X_0^i = x_i, i = 1, 2, \ldots, d \]

Consider the objective as a reward

\[ r(\{X_n^i\}) = \begin{cases} 
\lambda (L(x_{n-1}, y_i) - L(x_n, y_i)) & \text{If choose to continue} \\
0 & \text{Otherwise}
\end{cases} \]

You can also choose other approaches:
- A good image quality assessment without reference.
- A Classifier
- Fixed loop times according to the noise level
- A Person

### DURR Model

#### Results

Table 2: The average PSNR (dB) results on the BSD68 dataset. Values with * means the corresponding noise level is not present in the training data of the model. The top two methods are indicated with colors (red and blue) in top-down order of performance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma = 15$</td>
<td>31.07</td>
<td>31.31</td>
<td>31.60</td>
<td>31.47</td>
<td>31.38*</td>
</tr>
<tr>
<td>$\sigma = 25$</td>
<td>28.55</td>
<td>28.73</td>
<td>29.15</td>
<td>28.96</td>
<td>29.15</td>
</tr>
<tr>
<td>$\sigma = 35$</td>
<td>27.07</td>
<td>27.28</td>
<td>27.66</td>
<td>27.50</td>
<td>27.70</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>25.25*</td>
</tr>
<tr>
<td>$\sigma = 75$</td>
<td>22.63</td>
<td>22.71</td>
<td>18.73*</td>
<td>-</td>
<td>24.69*</td>
</tr>
</tbody>
</table>

### DURR Model Results

Table 2: The average PSNR (dB) results on the BSD68 dataset. Values with * means the corresponding noise level is not present in the training data of the model. The top two methods are indicated with colors (red and blue) in top-down order of performance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma = 15$</td>
<td>31.07</td>
<td>31.31</td>
<td>31.60</td>
<td>31.47</td>
<td>31.38*</td>
</tr>
<tr>
<td>$\sigma = 25$</td>
<td>28.55</td>
<td>28.73</td>
<td>29.15</td>
<td>28.96</td>
<td>29.15</td>
</tr>
<tr>
<td>$\sigma = 35$</td>
<td>27.07</td>
<td>27.28</td>
<td>27.66</td>
<td>27.50</td>
<td>27.70</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>25.25*</td>
</tr>
<tr>
<td>$\sigma = 75$</td>
<td>22.63</td>
<td>22.71</td>
<td>18.73*</td>
<td>-</td>
<td>24.69*</td>
</tr>
</tbody>
</table>
Nose Level Doesn’t Seen In Training

Figure 9: Denoising results of images from BSD68 with extreme noise conditions ($\sigma = 95$).
JPEG Deblocking

---

Table 3: The average PSNR(dB) on the LIVE1 dataset. Values with * means the corresponding QF is not present in the training data of the model. The top two methods are indicated with colors (red and blue) in top-down order of performance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>27.77</td>
<td>28.65</td>
<td>28.98</td>
<td>28.53</td>
<td>29.40</td>
<td>29.23*</td>
</tr>
<tr>
<td>20</td>
<td>30.07</td>
<td>30.81</td>
<td>31.29</td>
<td>30.88</td>
<td>31.59</td>
<td>31.68</td>
</tr>
<tr>
<td>30</td>
<td>31.41</td>
<td>32.08</td>
<td>32.69</td>
<td>32.31</td>
<td>32.98</td>
<td>33.05</td>
</tr>
<tr>
<td>40</td>
<td>32.45</td>
<td>32.99</td>
<td>33.63</td>
<td>33.39</td>
<td>33.96</td>
<td>34.01*</td>
</tr>
</tbody>
</table>

Application In Medical Imaging

UNROLLING REVISITED
Unrolled Dynamics: ADMM-Net

\[
\begin{align*}
\min_{x,z} & \frac{1}{2} \|Ax - y\|^2 + \sum_{l=1}^{L} \lambda_l g(D_l z) \\
\text{s.t.} & \quad z = x.
\end{align*}
\]

\[
\begin{align*}
X^{(n)} : x^{(n)} &= F^T (P^T P + \rho I)^{-1} \left[ P^T y + \rho F(z^{(n-1)} - \beta^{(n-1)}) \right], \\
Z^{(n)} : z^{(n,k)} &= \mu_1 z^{(n,k-1)} + \mu_2 (x^{(n)} + \beta^{(n-1)}) \\
&- \sum_{l=1}^{L} \lambda_l D_l^T \mathcal{H}(D_l z^{(n,k-1)}), \\
M^{(n)} : \beta^{(n)} &= \beta^{(n-1)} + \eta(x^{(n)} - z^{(n)}),
\end{align*}
\]

Sun, Li, and Xu. Deep ADMM-net for compressive sensing MRI. NIPS 2016.
## Unrolled Dynamics: ADMM-Net

<table>
<thead>
<tr>
<th>Method</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>Test Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMSE</td>
<td>PSNR</td>
<td>NMSE</td>
<td>PSNR</td>
<td>NMSE</td>
<td>PSNR</td>
</tr>
<tr>
<td>Zero-filling [46]</td>
<td>0.2624</td>
<td>26.35</td>
<td>0.1700</td>
<td>29.96</td>
<td>0.1247</td>
<td>32.59</td>
</tr>
<tr>
<td>TV [3]</td>
<td>0.1539</td>
<td>30.90</td>
<td>0.0929</td>
<td>35.20</td>
<td>0.0673</td>
<td>37.99</td>
</tr>
<tr>
<td>RecPF [11]</td>
<td>0.1498</td>
<td>30.99</td>
<td>0.0917</td>
<td>35.32</td>
<td>0.0668</td>
<td>38.06</td>
</tr>
<tr>
<td>SIDWT [3]</td>
<td>0.1564</td>
<td>30.81</td>
<td>0.0885</td>
<td>35.66</td>
<td>0.0620</td>
<td>38.72</td>
</tr>
<tr>
<td>PBDW [24]</td>
<td>0.1290</td>
<td>32.45</td>
<td>0.0814</td>
<td>36.34</td>
<td>0.0627</td>
<td>38.64</td>
</tr>
<tr>
<td>PANO [7]</td>
<td>0.1368</td>
<td>31.98</td>
<td>0.0800</td>
<td>36.52</td>
<td>0.0592</td>
<td>39.13</td>
</tr>
<tr>
<td>FDLC [29]</td>
<td>0.1257</td>
<td>32.63</td>
<td>0.0759</td>
<td>36.95</td>
<td>0.0592</td>
<td>39.13</td>
</tr>
<tr>
<td>BM3D-MRI [30]</td>
<td>0.1132</td>
<td>33.53</td>
<td>0.0674</td>
<td>37.98</td>
<td>0.0515</td>
<td>40.33</td>
</tr>
<tr>
<td>Init-Net [10]</td>
<td>0.2589</td>
<td>26.17</td>
<td>0.1737</td>
<td>29.64</td>
<td>0.1299</td>
<td>32.16</td>
</tr>
<tr>
<td>ADMM-Net [10]</td>
<td><strong>0.1082</strong></td>
<td><strong>33.88</strong></td>
<td><strong>0.0620</strong></td>
<td><strong>38.72</strong></td>
<td><strong>0.0480</strong></td>
<td><strong>40.95</strong></td>
</tr>
</tbody>
</table>
Further Application of Unrolling – Task-Based Image Reconstruction

Two-step approach: imaging and diagnosis

Problems of the two-step approach:
- Evaluation of the reconstructed image quality.
- Redundancy in data for a specific task.

Can we make it end-to-end, and does it help?
Further Application of Unrolling – Task-Based Image Reconstruction

D. Wu et al., End-to-End Lung Nodule Detection in Computed Tomography, MICCAI Workshop, 2018. (arXiv:1711.02074)
Further Application of Unrolling – Task-Based Image Reconstruction

D. Wu et al., End-to-End Lung Nodule Detection in Computed Tomography, MICCAI Workshop, 2018. (arXiv:1711.02074)
Further Application of Unrolling – Task-Based Image Reconstruction

Similar Ideas

D. Wu et al., End-to-End Lung Nodule Detection in Computed Tomography, MICCAI Workshop, 2018. (arXiv:1711.02074)
Deep Network Training

OPTIMAL CONTROL PERSPECTIVE
Optimization: Solving The “KKT” Condition

@Maximum Principle Based Algorithms

\[ H = p \cdot f - L \]

\[ \min_{\theta \in \mathcal{U}} \sum_{i=1}^{K} \Phi_i(X_T^i) + \int_{0}^{T} L(\theta_t) dt, \]

\[ \dot{X}_t^i = f(t, X_t^i, \theta_t), \quad X_0^i = x^i, \quad 0 \leq t \leq T, \quad i = 1, \ldots, K, \]

**Theorem 1** (Pontryagin’s Maximum Principle). Let \( \theta^* \in \mathcal{U} \) be an essentially bounded optimal control, i.e. a solution to (1), and \( X^* \) the corresponding optimally controlled process and \( \text{ess sup}_{t \in [0,T]} \| \theta_t^* \|_\infty < \infty \). Then, there exists an absolutely continuous co-state process \( P^*: [0,T] \rightarrow \mathbb{R}^d \) such that the Hamilton’s equations

\[ \dot{X}_t^* = \nabla_p H(t, X_t^*, P_t^*, \theta_t^*), \quad X_0^* = x, \]

\[ \dot{P}_t^* = -\nabla_x H(t, X_t^*, P_t^*, \theta_t^*), \quad P_T^* = -\nabla \Phi(X_T^*), \]

are satisfied. Moreover, for each \( t \in [0,T] \), we have the Hamiltonian maximization condition

\[ H(t, X_t^*, P_t^*, \theta_t^*) \geq H(t, X_t^*, P_t^*, \theta) \text{ for all } \theta \in \Theta \]
Optimization: Solving The “KKT” Condition

@Maximum Principle Based Algorithms

\[ H = p \cdot f - L \]

\[ \begin{align*}
    \min_{\theta \in \mathcal{U}} & \sum_{i=1}^{K} \Phi_i(X^i_T) + \int_{0}^{T} L(\theta_t) dt, \\
    \dot{X}^i_t &= f(t, X^i_t, \theta_t), \quad X^i_0 = x^i, \quad 0 \leq t \leq T, \quad i = 1, \ldots, K,
\end{align*} \]

\[ \text{Theorem 1 (Pontryagin’s Maximum Principle). Let } \theta^* \in \mathcal{U} \text{ be an essentially bounded optimal control, i.e. a solution to (1), and } X^* \text{ the corresponding optimally controlled process and ess sup}_{t \in [0, T]} \|P^*_t\|_\infty < \infty. \text{ Then, there exists an absolutely continuous co-state process } P^*: [0, T] \to \mathbb{R}^d \text{ such that the Hamilton’s equations}
\]

\[ \begin{align*}
    \dot{X}^*_t &= \nabla_p H(t, X^*_t, P^*_t, \theta^*_t), \quad X^*_0 = x, \\
    \dot{P}^*_t &= -\nabla_x H(t, X^*_t, P^*_t, \theta^*_t), \quad P^*_T = -\nabla \Phi(X^*_T),
\end{align*} \]

\[ \text{are satisfied. Moreover, for each } t \in [0, T], \text{ we have the Hamiltonian maximization condition}
\]

\[ H(t, X^*_t, P^*_t, \theta^*_t) \geq H(t, X^*_t, P^*_t, \theta) \text{ for all } \theta \in \Theta \]
Optimization: Solving The “KKT” Condition

@Maximum Principle Based Algorithms

\[ H = p \cdot f - L \]

\[
\begin{align*}
\min_{\theta \in \mathcal{U}} & \sum_{i=1}^{K} \Phi_i(X_i^T) + \int_{0}^{T} L(\theta_t)dt, \\
\dot{X}_i^t & = f(t, X_i^t, \theta_t), \quad X_0^i = x^i, \quad 0 \leq t \leq T, \quad i = 1, \ldots, K,
\end{align*}
\]

\[ 
\text{Theorem 1 (Pontryagin’s Maximum Principle). Let } \theta^* \in \mathcal{U} \text{ be an essentially bounded optimal control, i.e. a solution to (1), and } X^* \text{ the corresponding optimally controlled process and } \text{ess sup}_{t \in [0,T]} \|\theta_t^*\|_{\infty} < \infty. \text{ Then, there exists an absolutely continuous co-state process } P^* : [0,T] \to \mathbb{R}^d \text{ such that the Hamiltonian equations}
\]

\[
\begin{align*}
\dot{X}_i^* & = \nabla_p H(t, X_i^t, P_i^t, \theta_t^t), \quad X_0^* = x, \\
\dot{P}_i^* & = -\nabla_x H(t, X_i^t, P_i^t, \theta_t^t), \quad P_T^* = -\nabla \Phi(X_T^*),
\end{align*}
\]

\[ 
\text{are satisfied. Moreover, for each } t \in [0,T], \text{ we have the Hamiltonian maximization condition}
\]

\[
H(t, X_i^*, P_i^*, \theta_t^t) \geq H(t, X_i^*, P_i^*, \theta) \text{ for all } \theta \in \Theta
\]
Optimization: Solving The “KKT” Condition

@Maximum Principle Based Algorithms

\[ \dot{X}_t = \nabla P \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \quad X_0^* = x, \]
\[ \dot{P}_t^* = -\nabla X \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \quad P_T^* = -\nabla X \Phi(X_T^*), \]
\[ \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*) \geq \tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*), \quad \theta \in \Theta, t \in [0, T]. \]

Solving it via Gauss-Seidel Iteration

---

Algorithm 2: Extended MSA (E-MSA)

1. Initialize: $\theta^0 \in \mathcal{U}$. Hyper-parameter: $\rho$
2. for $k = 0$ to $\#$Iterations do
3. Solve $\dot{X}_t^{\theta^k} = f(t, X_t^{\theta^k}, \theta_t^k)$, $X_0^{\theta^k} = x$
4. Solve $\dot{P}_t^{\theta^k} = -\nabla X H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k)$, $P_T^{\theta^k} = -\nabla \Phi(X_T^{\theta^k})$
5. Set $\theta_t^{k+1} = \arg \max_{\theta \in \Theta} \tilde{H}(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta, \dot{X}_t^{\theta^k}, \dot{P}_t^{\theta^k})$ for each $t \in [0, T]$

Qianxiao Li, Long Chen, Cheng Tai, and Weinan E Maximum Principle Based Algorithms for Deep Learning
Optimization: Solving The “KKT” Condition

@Maximum Principle Based Algorithms

\[
\begin{align*}
\dot{X}_t^* &= \nabla_p \bar{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \\
\dot{P}_t^* &= -\nabla_x \bar{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \\
\bar{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*) &\geq \bar{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*), \\
X_0^* &= x, \\
P_T^* &= -\nabla_x \Phi(X_T^*), \\
\theta &\in \Theta, t \in [0, T].
\end{align*}
\]

Solving it via Gauss-Seidel Iteration

Algorithm 2: Extended MSA (E-MSA)

1: Initialize: $\theta^0 \in \mathcal{U}$. Hyper-parameter: $\rho$
2: for $k = 0$ to #Iterations do
3: \hspace{1cm} Solve $\dot{X}_t^{\theta_k} = f(t, X_t^{\theta_k}, \theta_t^k)$, $X_0^{\theta_k} = x$
4: \hspace{1cm} Solve $\dot{P}_t^{\theta_k} = -\nabla_x H(t, X_t^{\theta_k}, P_t^{\theta_k}, \theta_t^k)$, $P_T^{\theta_k} = -\nabla \Phi(X_T^{\theta_k})$
5: \hspace{1cm} Set $\theta_t^{k+1} = \arg \max_{\theta \in \Theta} \bar{H}(t, X_t^{\theta_k}, P_t^{\theta_k}, \theta, \dot{X}_t^{\theta_k}, \dot{P}_t^{\theta_k})$ for each $t \in [0, T]$

Back Propagation: argmax step instead of a gradient ascent
Works For Binary NN

Li Q, Hao S. An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks. ICML2018.
Neural ODE

Algorithm 1 Reverse-mode derivative of an ODE initial value problem

Input: dynamics parameters $\theta$, start time $t_0$, stop time $t_1$, final state $z(t_1)$, loss gradient $\partial L/\partial z(t_1)$

\[
\frac{\partial L}{\partial t_1} = \frac{\partial L}{\partial z(t_1)}^T f(z(t_1), t_1, \theta) \quad \triangleright \text{Compute gradient w.r.t. } t_1
\]

\[
s_0 = [z(t_1), \frac{\partial L}{\partial z(t_1)}, 0, -\frac{\partial L}{\partial t_1}] \quad \triangleright \text{Define initial augmented state}
\]

def aug_dynamics([z(t), a(t), -], t, \theta):

\[
\text{return } [f(z(t), t, \theta), -a(t)^T \frac{\partial f}{\partial z}, -a(t)^T \frac{\partial f}{\partial \theta}, -a(t)^T \frac{\partial f}{\partial \theta}] \quad \triangleright \text{Define dynamics on augmented state}
\]

\[
[z(t_0), \frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial \theta}] = \text{ODESolve}(s_0, \text{aug_dynamics}, t_1, t_0, \theta) \quad \triangleright \text{Solve reverse-time ODE}
\]

\[
\text{return } \frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial \theta} \quad \triangleright \text{Return all gradients}
\]

Recall the PMP

\[
X_t^* = \nabla_p H(t, X_t^*, P_t^*, \theta_t^*), \quad X_0^* = x,
\]

\[
P_t^* = -\nabla_x H(t, X_t^*, P_t^*, \theta_t^*), \quad P_0^* = -\nabla \Phi(X_0^*).
\]
VAE and Normalizing Flow

Variational Principle: estimating the density of data \( x \) by maximizing \(-F(x)\)

\[
\log p_\theta(x) = \log \int p_\theta(x|z)p(z)dz \\
= \log \int \frac{q_\phi(z|x)}{q_\phi(z|x)}p_\theta(x|z)p(z)dz \\
\geq -\text{KL}[q_\phi(z|x)\|p(z)] + \mathbb{E}_q[\log p_\theta(x|z)] = -F(x),
\]
VAE and Normalizing Flow

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

\[ \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z)) \]

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!
VAE and Normalizing Flow

Normalizing flow for variational inference: provides a more flexible family of estimators of the unknown $p(z|x)$

$$z_K = f_K \circ \ldots \circ f_2 \circ f_1(z_0)$$

$$\ln q_K(z_K) = \ln q_0(z_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|$$

where $f_j$ are smooth invertible maps

**Algorithm 1** Variational Inf. with Normalizing Flows

<table>
<thead>
<tr>
<th>Parameters: $\phi$ variational, $\theta$ generative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>while</strong> not converged <strong>do</strong></td>
</tr>
<tr>
<td>$x \leftarrow {\text{Get mini-batch}}$</td>
</tr>
<tr>
<td>$z_0 \sim q_0(\bullet</td>
</tr>
<tr>
<td>$z_K \leftarrow f_K \circ f_{K-1} \circ \ldots \circ f_1(z_0)$</td>
</tr>
<tr>
<td>$\mathcal{F}(x) \approx \mathcal{F}(x, z_K)$</td>
</tr>
<tr>
<td>$\Delta \theta \propto -\nabla_\theta \mathcal{F}(x)$</td>
</tr>
<tr>
<td>$\Delta \phi \propto -\nabla_\phi \mathcal{F}(x)$</td>
</tr>
<tr>
<td><strong>end while</strong></td>
</tr>
</tbody>
</table>
Use the change of variables theorem to compute exact changes in probability if samples are transformed through a bijective function $f$:

$$z_1 = f(z_0) \implies \log p(z_1) = \log p(z_0) - \log \left| \det \frac{\partial f}{\partial z_0} \right|$$

Use NODE:

$$\frac{\partial \log p(z(t))}{\partial t} = -\text{tr} \left( \frac{df}{dz(t)} \right)$$

Reducing the calculation cost of gradient from $O(d^3)$ to $O(d)$
VAE and Normalizing Flow

Normalizing flow for image synthesis:

VAE and Normalizing Flow

Normalizing flow for image synthesis:

Applied Math Perspective on Deep Learning

Take home message:

Deep Network ↔ Differential Equations (DE)
Network Architecture ↔ Numerical DE
Network Training ↔ Optimal Control

Likewise for coffee:
From David Wipf’s Slide@ICASSP2018
Thanks and Questions?