Differential Equation, Deep Learning, Sparse Representation And Beyond

YIPING LU
PEKING UNIVERSITY
SCHOOL OF MATHEMATICAL SCIENCE
HTTP://ABOUT.2PRIME.CN

Joint working with
Ting Lin, Zhihan Li, Zichao Long, Xianzhong Ma, Bin Dong(SMS and BICMR, Peking Unviersity)
Xiaoshuai Zhang, Jiaying Liu,Liwei Wang(EECS, Peking University)
Aoxiao Zhong, Quanzheng Li(Harvard University)
Zuowei Shen(NUS)

The Level Set Collective, UCLA, 2018/08
Methods In Inverse Problem

Total Variation
PM Equation
Nonlinear Diffusion

Compressed Sensing
Low-rank Matrix

Deep Learning
Neural Networks
ResNets


Methods In Imaging Science


Traditional Wisdom In Deep Learning

Past
Algorithms originated by humans dominate

Present
?

Future
Blackbox DNNs possibly dominate everything

Likewise for coffee:

From David Wipf’s Slide@ICASSP2018
Compressed Sensing
BEFORE DEEP LEARNING
Geometry View Of Sparsity

Sparsity After Transform

$|Wu|_1$
Geometry View Of Sparsity

Sparsity After Transform

\[ |Wu|_1 \]

Geometry

\[ BV(\Omega) = \left\{ u \in L^1(\Omega) ; \int_{\Omega} |Du| < \infty \right\}. \]

\[ Du = \nabla u \, dx + (u^+ - u^-) n_u \mathcal{H}^{N-1}_{|S_u} + C_u. \]
Geometry View Of Sparsity

Sparsity After Transform

\[ |Wu|_1 \]

Discretization

\[ \int_{\Omega} |Du| \]

Geometry

\[ BV(\Omega) = \left\{ u \in L^1(\Omega); \int_{\Omega} |Du| < \infty \right\}. \]

How To Understand **True Sparsity**

### Analysis Based Approach

$$\min_u \frac{1}{2} \|Au - f\|_2^2 + \left\| \sum_{L:\text{level}} \left( \sum_{j:\text{band}} \lambda_{l,j} |W_{l,j}u|^q \right)^{1/q} \right\|_1$$

### $\ell_0$ Minimization For Wavelet Frame Based Image Restoration

$$\min_{u \in \mathcal{Y}} \frac{1}{2} \|Au - f\|_D^2 + \sum_i \lambda_i \| (Wu)_i \|_0$$
Unnatural l0 Approximates Mumford-Shah

\[ \| u_n - f_n \|^2 + \| H(W_n u_n) \| \]

Here \( f_n = T_n f \)

Gamma Converge

\[ \| u - f \|_{L^2}^2 + \alpha \| \nabla u \|_{L^2(K^c)}^2 + \beta \mathcal{H}(K) \]
Unnatural l0 in literature


**Gamma Converge Result**

Ting Lin, Yiping Lu, Bin Dong, Zuowei Shen *In Preparation*

---

**Edge Driven Unnatural Zero Wavelet Frame Based Model**

\[
\inf_{u \in I_2, \Gamma \subset O^2} \left\{ \| [\lambda \cdot W u]_{\Gamma c} \|_2^2 + \| [\mu \cdot W u]_{\Gamma} \|_W + \frac{1}{2} \| Au - f \|_2^2 \right\}
\]

**Edge Driven Unnatural Zero Variational Model**

\[
\inf_{u \in I_2, \Gamma \subset O^2} \left\{ \| D u \|_2^2 + \sum |\Gamma_j^*| + \frac{1}{2} \| Au - f \|_2^2 \right\}
\]

---

**Convergence**

With the same assumptions, for each series \( u_n \) in \( H^{1,s} \), \( u_n \rightarrow u \), we have

\[
\lim_{n \rightarrow \infty} E_n(u_n) = E(u). \quad (10)
\]


<table>
<thead>
<tr>
<th>wavelet</th>
<th>gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>Image restoration</td>
</tr>
<tr>
<td>Unnatural L0</td>
<td>Image restoration</td>
</tr>
</tbody>
</table>

Have the property represents the smooth interior
Numerical Result

\[ \text{Edge} + \text{Smooth}: l_1 + l_2 \quad \text{Edge} + \text{Smooth}: l_0 + l_2 \]

[Reconstructed]

Test Image: Yuka From NGT48

Jump Set Estimation
PDE and Learning In Imaging

Optimization Energy Function

Approximate

Variation form

Gradient Descent

Gradient Flow


Neural Network Is Approximation
Deep Neural Networks As Numerical Scheme
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)

- Bo C, Meng L, et al. *Reversible Architectures for Arbitrarily Deep Residual Neural Networks*
Previous Works

TRD(@CVPR2015): learn a diffusion process for denoising

Depth Revolution

Deeper And Deeper
Depth Revolution

Going into infinite layer

Differential Equation
As Infinite Layer
Neural Network

152 layers

ILSVRC'15
ResNet
3.57

ILSVRC'14
GoogleNet
6.7

ILSVRC'14
VGG
7.3

ILSVRC'13
8 layers
11.7

ILSVRC'12
AlexNet
16.4

shallow

ILSVRC'11
25.8

ILSVRC'10
28.2
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)

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

PDE: Infinite Layer Neural Network

Dynamic System  
Continuous limit

Neural Network  
Numerical Approximation

<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>$X = f_1(Y), \dot{Y} = f_2(X)$</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

\[ x_t = f(x) \]

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

Linear Multi-step Scheme

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

Linear Multi-step Residual Network
(a) ResNet

(b) Linear Multi-step ResNet
(a) ResNet

(b) Linear Multi-step ResNet

Only One More Parameter
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>ResNet (He et al. (2016))</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>

2x fewer parameters
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**

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

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

**LM-ResNet**

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

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


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

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

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

Noise can avoid overfit?
**Previous Works**

Shake-Shake regularization

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

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

Apply data augmentation techniques to internal representations.

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

Previous Works

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.
Bridge the stochastic control

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**!

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

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.
\[(1 + k_n) \dot{u} + (1 - k_n) \frac{\Delta t}{2} \ddot{u}_n + o(\Delta t^3) = f(u) + g(u)dW_t\]
### 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>ResNet (Huang et al. (2016b))</td>
<td>1202</td>
<td>Stochastic depth</td>
<td>4.91</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>LM-ResNet (Ours)</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

Neural Network ↔ Dynamic System

Stochastic Learning ↔ Stochastic Dynamic System

New Discretization

LM-ResNet

Original One:  LM-Resnet56 Beats Resnet110

Modified Equation

Stochastic Depth One:  LM-Resnet110 Beats Resnet1202
Application In Learning A PDE

DATA DRIVEN PHYSIC LAW DISCOVERY
Learn A PDE?
Learn A PDE?
Convolutional Operator Vs Differential Operator

Proposition 2.1. 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

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

where $C_{\alpha}$ is the constant defined by

\[
C_{\alpha} = \frac{1}{\alpha!} \sum_{k \in \mathbb{Z}^2} k^\alpha q[k].
\]

If, in addition, $q$ has total sum rules of order $K \setminus \{\lvert \alpha \rvert + 1\}$ for some $K > \lvert \alpha \rvert$, then

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

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

One Delta-t Block

\[ u_t = F(x, u, \nabla u, \nabla^2 u, \ldots), \quad x \in \Omega \subset \mathbb{R}^2, \quad t \in [0, T]. \]
PDE-Net

1st $\delta t$ – block

$u^0 \rightarrow D_{00}u^0 \rightarrow D_{10}u^0 \rightarrow (x,y) \rightarrow \hat{u}^1 \rightarrow \frac{D_{00}\hat{u}^1}{D_{10}\hat{u}^1} \rightarrow (x,y) \rightarrow \frac{u^1}{\text{loss 1}} \rightarrow \hat{u}^1$

2nd $\delta t$ – block

$\ldots \rightarrow \hat{u}^n \rightarrow \frac{D_{00}\hat{u}^n}{D_{10}\hat{u}^n} \rightarrow (x,y) \rightarrow \frac{u^n}{\text{loss n}} \rightarrow \hat{u}^n \rightarrow \ldots$

n-th $\delta t$ – block

$\ldots \rightarrow \hat{u}^n \rightarrow \frac{D_{00}\hat{u}^n}{D_{10}\hat{u}^n} \rightarrow (x,y) \rightarrow \frac{u^n}{\text{loss n}} \rightarrow \hat{u}^n \rightarrow \ldots$
Numerical Result

We consider a 2-dimensional linear variable-coefficient convection-diffusion equation on $\Omega = [0, 2\pi] \times [0, 2\pi]$, 

$$
\begin{align*}
\frac{\partial u}{\partial t} &= a(x, y)u_x + b(x, y)u_y + cu_{xx} + du_{yy} \\
                  &= u_0(x, y),
\end{align*}
$$

with $(t, x, y) \in [0, 0.2] \times \Omega$, \quad (8)

where 

$$
a(x, y) = 0.5(\cos(y) + x(2\pi - x)\sin(x)) + 0.6, \quad b(x, y) = 2(\cos(y) + \sin(x)) + 0.8,
$$

$c = 0.2$ and $d = 0.3$. 

Numerical Result
Nonlinear Examples

True Dynamic

Learned Dynamic

Learned Coefficient

Target Equation

\[
\begin{aligned}
\frac{\partial u}{\partial t} + c \Delta u + f_s(u) &= 0, \\
u|_{t=0} &= u_0(x, y),
\end{aligned}
\]

with \((t, x, y) \in [0, 0.2] \times \Omega\).
Learn Filter Vs No Learning
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
Existing Works


designed a 20-layer neural network, called DnCNN-B, which had a huge number of parameters

unrolled a projection gradient algorithm for a constrained optimization model
What Happen When Meet High Noise Level

Fails!
Existing Works


Designed according to the upper bound of the noise level
\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*}
Determine The End Time

Filter Residual

\[ T = \arg\min_t \| \text{corr}(u(0) - u(t), u(t)) \|. \]  

(12)

Filter Result
Moving Endpoint Control

Terminal time as a variable to train

Early Stopping Is A Regularization

Can we train it?

Need to be learn

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

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

\[
\min \int_0^\tau R(w(t), t)dt \\
\text{s.t.} \quad \dot{X} = f(X(t), w(t)), \\
\tau \text{ is the first time that dynamic meets } X \\
\tau \text{ is the time} \\
\text{arrives the moon}
\]

Control dynamic: 
Physic Law
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>Training Noise</th>
<th>Policy</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>35, 45</td>
<td>Naive</td>
<td></td>
<td>27.74</td>
<td>27.17</td>
<td>26.66</td>
<td>26.24</td>
<td>26.75</td>
<td>25.61</td>
<td>24.75</td>
</tr>
<tr>
<td>35, 45</td>
<td>Refined</td>
<td></td>
<td>29.14</td>
<td>28.33</td>
<td>27.67</td>
<td>27.19</td>
<td>27.69</td>
<td>26.61</td>
<td>25.88</td>
</tr>
</tbody>
</table>
Complete DURR Discretize: Turn To An RL Problem

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

\[
s.t. X_n^i = X_{n-1}^i + \Delta t f(X_{n-1}^i, w(t)), n = 1, 2, \ldots, N_i, (i = 1, 2, \ldots, d)
X_0^i = x_i, i = 1, 2, \ldots, d
\]

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

Consider the objective as a reward

**Algorithm 1** Dynamically Unfolding Recurrent Restorer (DURR) Training via Policy Gradient

**Input:** The target \(y_i\) and noisy observation \(x_i\)

1. Initialize the weights of the restoration unit and the policy unit.
2. Pretrain the restoration unit with defined policies.
3. Set epochs \(M\) and the hyper-parameters in the algorithm.
4. for \(t \leftarrow 1\) to \(M\) do
5. Fix the restoration unit and simulate the forward trajectories using \(\pi_\theta\)
6. Calculate the policy gradient and then perform the optimization:

\[
\theta \leftarrow \theta - \eta \nabla_\theta \mathbb{E}_{X \sim \pi_\theta} \left[ \sum_{n=1}^{N_i} r(\{X_n^i, w\}) \left( \nabla_\theta \sum_{n=1}^{N_i} \log P(X_n^i, \theta) \right) \right]
\]

. The expectation here is estimated on the sampled trajectory.

**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
**Completed DURR**

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>σ = 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>σ = 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>σ = 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>σ = 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>σ = 65</td>
<td>24.69</td>
<td>24.51</td>
<td>23.40*</td>
<td>-</td>
<td>25.25*</td>
</tr>
<tr>
<td>σ = 75</td>
<td>22.63</td>
<td>22.71</td>
<td>18.73*</td>
<td>-</td>
<td>24.69*</td>
</tr>
</tbody>
</table>
Completed DURR

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>σ = 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>σ = 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>σ = 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>σ = 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>σ = 65</td>
<td>24.69</td>
<td>24.51</td>
<td>23.40*</td>
<td>-</td>
<td>25.25*</td>
</tr>
<tr>
<td>σ = 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>
Image quality's relation to loop times.

Bad Image Needs More Loops
Real Noise

Generalize

We can generalize.
# Processing Sequence For Real Image

Works like a bilateral filter

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

![Image of processing sequence]
Shortage And Further Direction

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>
Shortage And Further Direction
Recent Other Methods

\[ E(u, f_j) = \sum_{i=1}^{M} \lambda_{\sigma_i} \frac{1}{2} \| u - f_i \|^2 + \sum_{i=1}^{N_k} \rho_i (k_i \ast u), \]

**Different data term**

\[
\Theta^* = \arg\min_{\Theta} \mathcal{L}(\Theta) = \sum_{s=1}^{S} \ell (u_T^s, u_{gt}^s)
\]
\[
\begin{align*}
  u_0^s &= f_{js}, & t = 1 \cdots T \\
  u_t^s &= u_{t-1}^s - \left( \sum_{i=1}^{N_k} \bar{k}_i^t \ast \phi_i^t (k_i^t \ast u_{t-1}^s) + \lambda_{\sigma_j}^t (u_{t-1}^s - f_{js}^s) \right),
\end{align*}
\]

**Related Works**


Also other applications

Different Damage level

De-rain  De-haze  De-Blur
Deep Sparse Representation

ON GOING – I CAN’T GET GOOD RESULTS.
Deep Image Prior

\[ x^* = \min_x E(x; x_0) + R(x), \]

where \( E(x; x_0) \) is a task-dependent data term, \( x_0 \) the noisy/low-resolution/occluded image, and \( R(x) \) a regularizer.

In terms of \( x^* \), the prior \( R(x) \) defined by (2) is an indicator function \( R(x) = 0 \) for all images that can be produced from \( x \) by a deep ConvNet of a certain architecture, and \( R(x) = +\infty \) for all other signals. Since no aspect of the network is pre-trained from data, such deep image prior is effectively handcrafted, just like the TV norm. We show that this hand-crafted prior works very well for various image restoration tasks.

Figure 1: **The architecture used in the experiments.** We use “hourglass” (also known as “decoder-encoder”) architecture. We sometimes add skip connections (yellow arrows). \( n_u[i], n_d[i], n_s[i] \) correspond to the number of filters at depth \( i \) for the upsampling, downsampling and skip-connections respectively. The values \( k_u[i], k_d[i], k_s[i] \) correspond to the respective kernel sizes.

[https://dmitryulyanov.github.io/deep_image_prior](https://dmitryulyanov.github.io/deep_image_prior)
Multi-Layer DDTF

Idea: Neural Network = Shrinkage In Sparse Coding

- \[ \min ||u - f||^2 + \lambda_1 ||f - D_1 \Gamma_1||^2 + \mu_1 ||\Gamma_1||_0 + \lambda_2 ||\Gamma_2 - D_2 \Gamma_2||^2 + \mu_2 ||\Gamma_2||_0 \]
- \[ s.t. D_1^T D_1 = I_d, D_2^T D_2 = I_d \]

Alternating Minimization

Also Can Do Multilayer

Smaller Kernel=> Faster Speed
### Multi-Layer DDTF

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>DDDTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td>picture</td>
<td>L0</td>
<td>DDTF</td>
<td></td>
</tr>
<tr>
<td>deblur</td>
<td>fingerprint</td>
<td>27.83</td>
<td>29.01</td>
<td>29.33</td>
</tr>
<tr>
<td>lena</td>
<td>32.45</td>
<td></td>
<td>33.24</td>
<td>33.31</td>
</tr>
<tr>
<td>hill</td>
<td>30.38</td>
<td></td>
<td>30.6</td>
<td>30.6</td>
</tr>
<tr>
<td>couple</td>
<td>29.5</td>
<td></td>
<td>29.59</td>
<td></td>
</tr>
<tr>
<td>boat</td>
<td>29.74</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Problems:**
- small kernel
- shallow layers

Converge to stationary point


Yong Zhang, Bin Dong and Zhaosong Lu, l0 minimization of wavelet frame based image restoration, Mathematics of Computation, 82, 995-1015, 2013
“Hourglass” View

Reconstruction needs previous layer information
Some success work in this direction

Gradient calculation by KKT condition and bp algorithm.

min \frac{1}{S} \sum_{s=1}^{S} L(y_{s}, f(A_{s}^{(h)}, w)) + \frac{\mu}{2} R(\theta),
\text{s.t.} \quad \alpha_{s}^{(H)} = \arg \min_{\alpha_{s}^{(H)} \geq 0} F(D^{(H)}, x_{s}^{(H)}, \alpha_{s}^{(H)}),
\text{s.t.} \quad \alpha_{s}^{(L)} = \arg \min_{\alpha_{s}^{(L)} \geq 0} F(D^{(L)}, x_{s}^{(L)}, \alpha_{s}^{(L)}),
\text{s.t.} \quad \lambda^{(h)} > 0, \quad x_{s}^{(h)} = \psi(\alpha_{s}^{(h-1)}), \quad \forall h = 1, \ldots, H,
Some success work in this direction

<table>
<thead>
<tr>
<th>Method</th>
<th># Params</th>
<th># Layers</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCKN [34]</td>
<td>10.50M</td>
<td>10</td>
<td>10.20</td>
<td>-</td>
</tr>
<tr>
<td>OMP [18]</td>
<td>0.70M</td>
<td>2</td>
<td>18.50</td>
<td>-</td>
</tr>
<tr>
<td>PCANet [36]</td>
<td>0.28B</td>
<td>3</td>
<td>21.33</td>
<td>-</td>
</tr>
<tr>
<td>NOMP [7]</td>
<td>1.09B</td>
<td>4</td>
<td>18.60</td>
<td>39.92</td>
</tr>
<tr>
<td>NiN [32]</td>
<td>-</td>
<td>-</td>
<td>8.81</td>
<td>35.68</td>
</tr>
<tr>
<td>DSN [33]</td>
<td>1.34M</td>
<td>7</td>
<td>7.97</td>
<td>36.54</td>
</tr>
<tr>
<td>WRN [12]</td>
<td>36.5M</td>
<td>28</td>
<td>4.00</td>
<td>19.25</td>
</tr>
<tr>
<td>ResNet-110</td>
<td>0.85M</td>
<td>110</td>
<td>6.41</td>
<td>27.22</td>
</tr>
<tr>
<td>ResNext-29</td>
<td>68.10M</td>
<td>29</td>
<td>3.58</td>
<td>17.31</td>
</tr>
<tr>
<td>SwapOut-20</td>
<td>1.10M</td>
<td>20</td>
<td>5.68</td>
<td>25.86</td>
</tr>
<tr>
<td>SwapOut-32</td>
<td>7.43M</td>
<td>32</td>
<td>4.76</td>
<td>22.72</td>
</tr>
<tr>
<td>SCN-1</td>
<td>0.17M</td>
<td>15</td>
<td>8.86</td>
<td>25.08</td>
</tr>
<tr>
<td>SCN-2</td>
<td>0.35M</td>
<td>15</td>
<td>7.18</td>
<td>22.17</td>
</tr>
<tr>
<td>SCN-4</td>
<td>0.69M</td>
<td>15</td>
<td>5.81</td>
<td>19.93</td>
</tr>
</tbody>
</table>


Thank You

Homepage: http://about.2prime.cn