Dynamic System and Optimal Control Perspective of Deep Learning

BIN DONG

PEKING UNIVERSITY

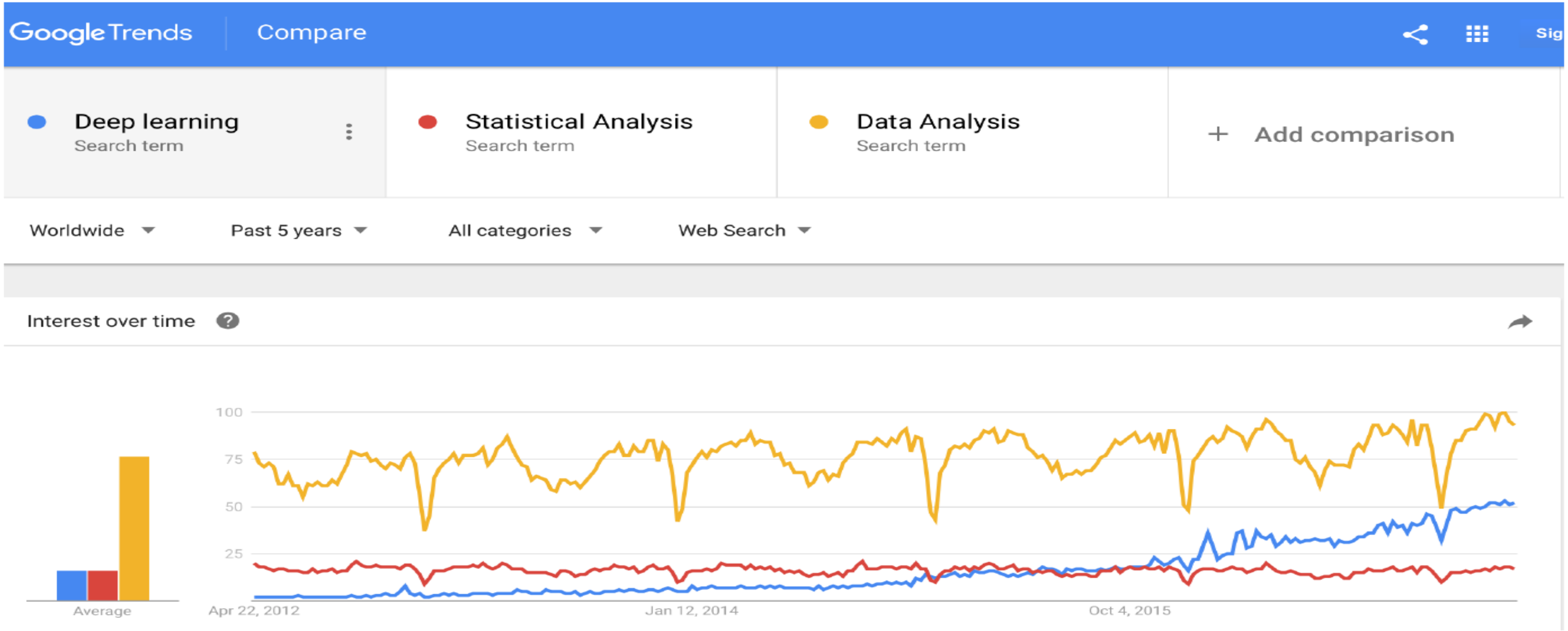
Special thanks to **Yiping Lu** who helped in preparation of the slides.

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

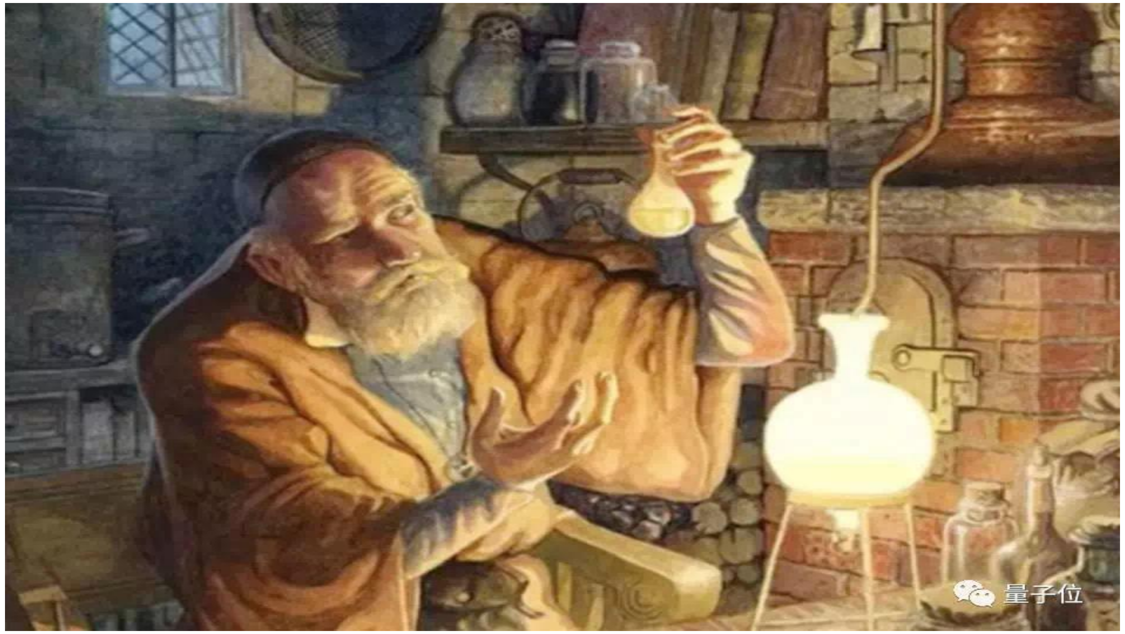
Deep Learning: Burning Hot!

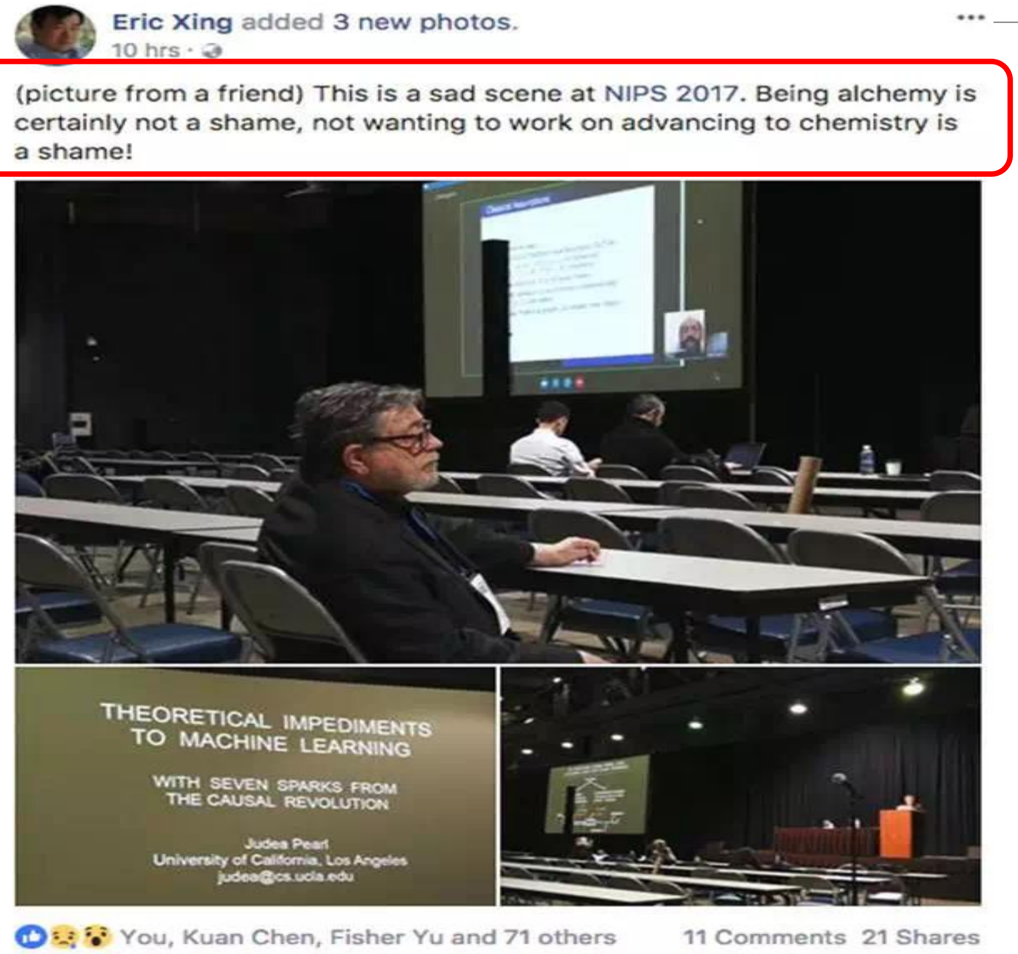


Credit: D. Donoho/ H. Monajemi/ V. Papyan “Stats 385”@Stanford

Deep Learning

Deep learning is "alchemy"
- Ali Rahimi, NIPS 2017





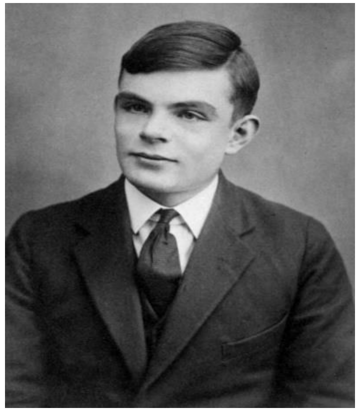
Deep Learning

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

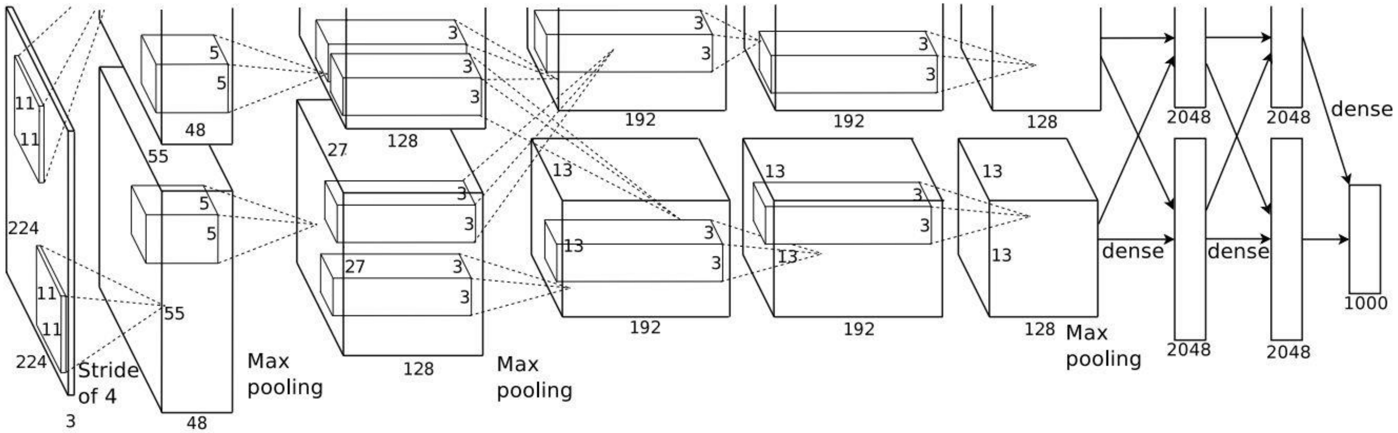
NETWORK STRUCTURE DESIGN

Depth Neural Network

Deep Neural Network

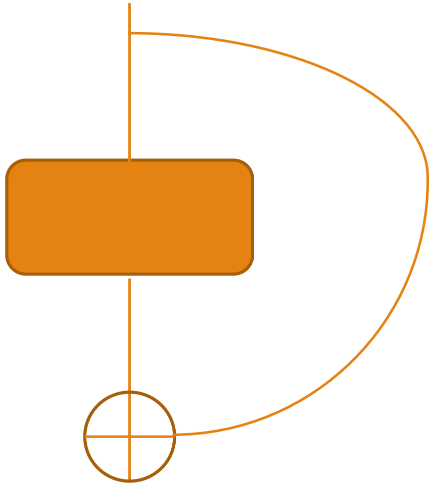
$$f_1\left(f_2(f_3\cdots(x))\right)$$

A Dynamic System?



Motivation

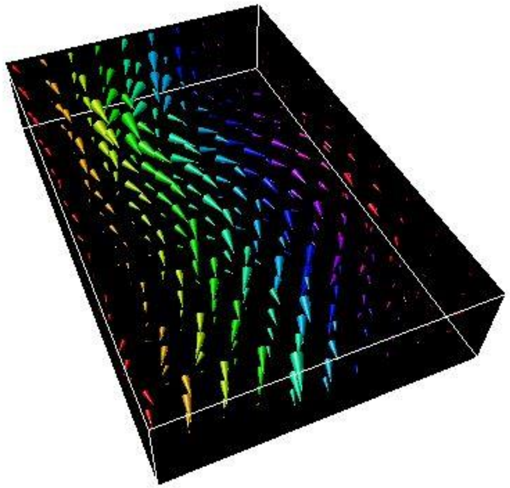
Deep Residual Learning(@CVPR2016)



$$x_t = f(x)$$

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

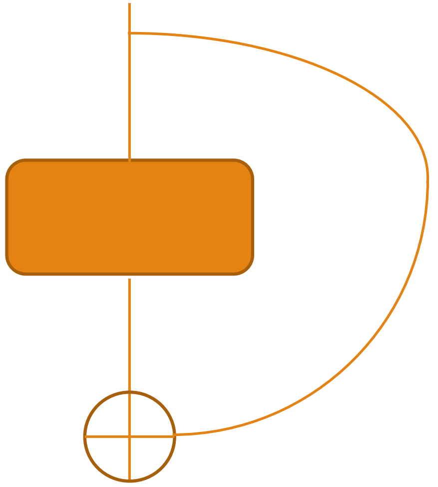
Forward Euler Scheme



- Weinan E. A Proposal on Machine Learning via Dynamical Systems. Communications in Mathematical Science, 2017.
- Haber E, Ruthotto L. Stable architectures for deep neural networks[J]. Inverse Problems, 2017.
- Bo C, Meng L, et al. Reversible Architectures for Arbitrarily Deep Residual Neural Networks, AAAI 2018
- Lu Y. et al., Beyond Finite Layer Neural Network: Bridging Deep Architects and Numerical Differential Equations, ICML 2018.

Motivation

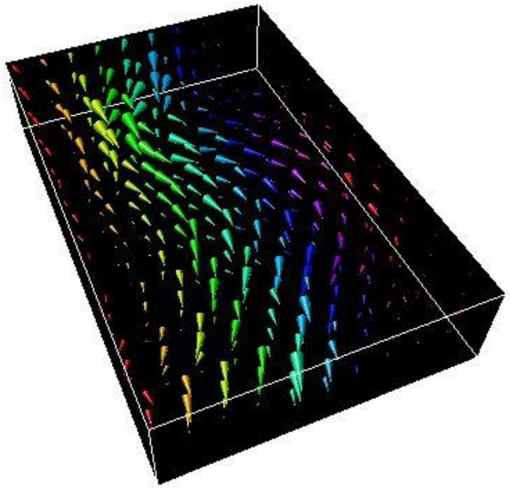
Deep Residual Learning(@CVPR2016)



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

$$x_t = f(x)$$

<ins>**Forward Euler Scheme**</ins>



Theoretical Convergence Results is built in:

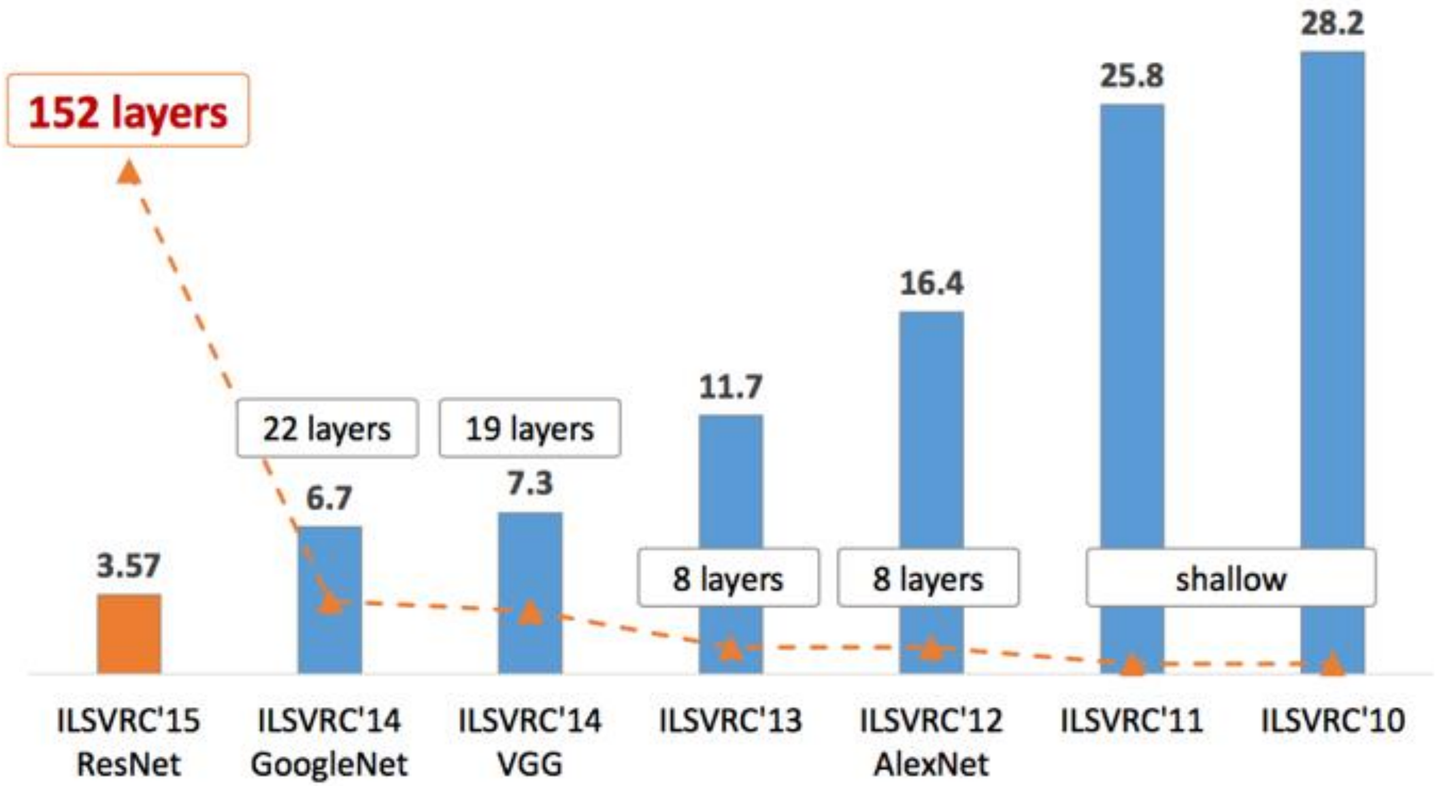
Thorpe, Matthew, and Yves van Gennip. "Deep Limits of Residual Neural Networks *preprint arXiv:1810.11741*(2018).

A New Generalization Perspective From Control:

Han, Jiequn, and Qianxiao Li. "A mean-field optimal control formulation of deep learning." arXiv preprint arXiv:1807.01083(2018).

Depth Revolution

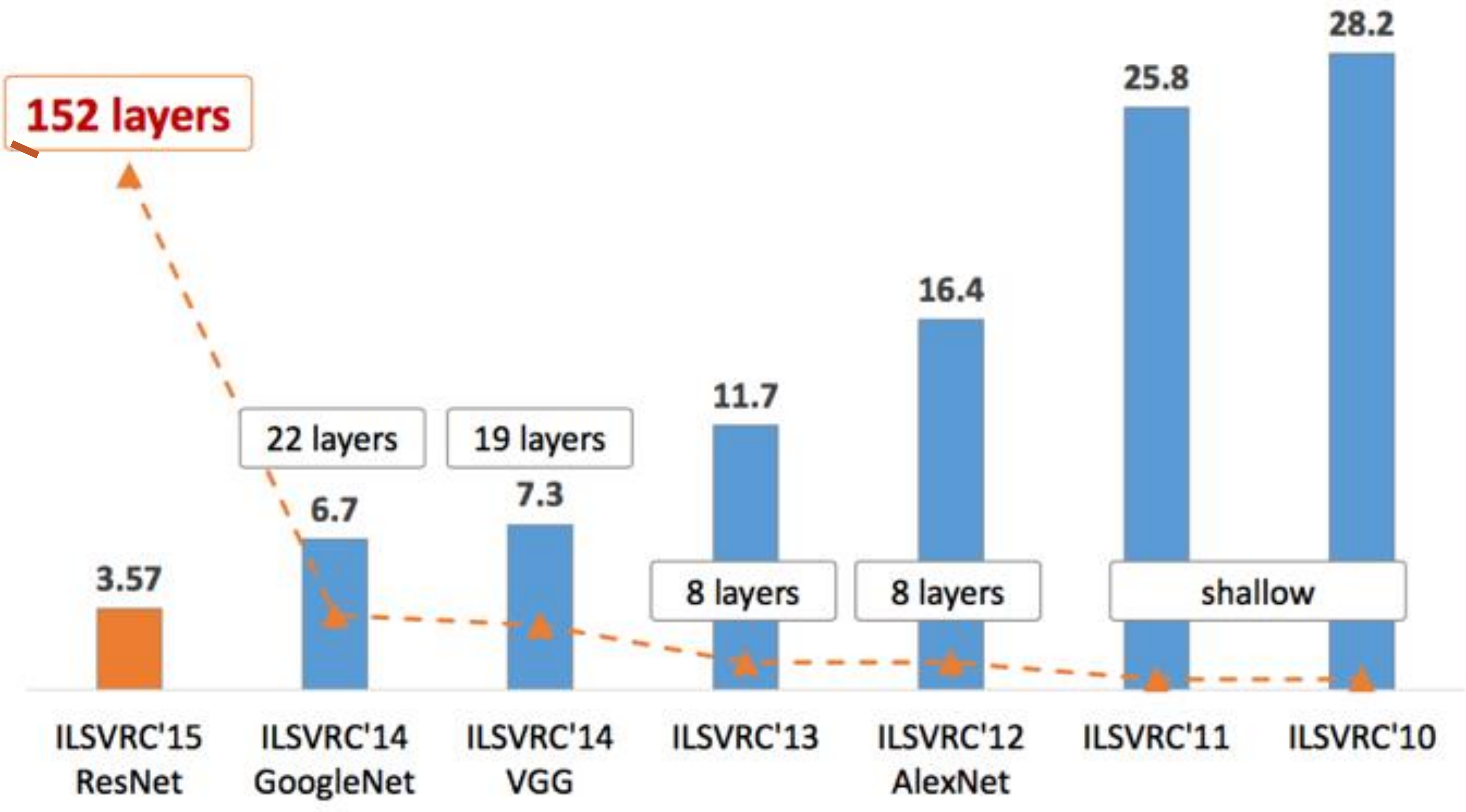
Deeper And Deeper



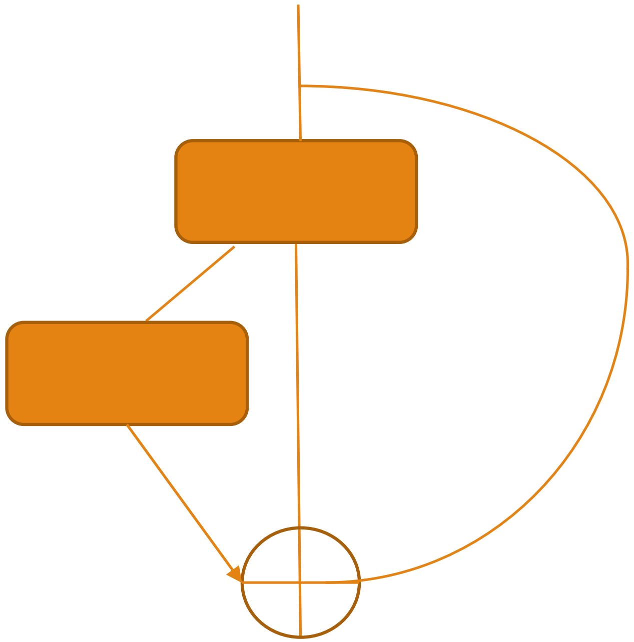
Depth Revolution

Going into infinite layer

Differential Equation
As Infinite Layer
Neural Network



Polynet(@CVPR2017)



(b) Polynet

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: $x_{n+1} = x_n + F(x_n) + F(F(x_n))$

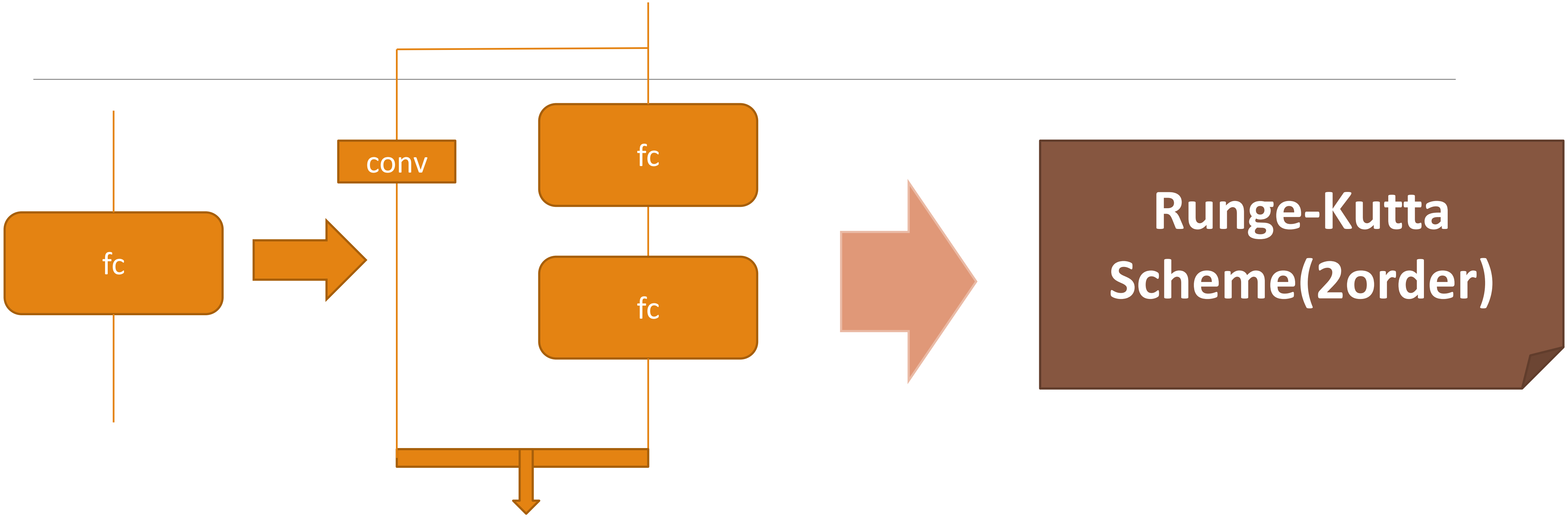
Backward Euler Scheme:

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

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

Zhang X, Li Z, Loy C C, et al. PolyNet: A Pursuit of Structural Diversity in Very Deep Networks. CVPR 2017

FractalNet(@ICLR2017)



$$x_{n+1} = k_1 x_n + k_2(k_3 x_n + f_1(x_n)) + f_2(k_3 x_n + f_1(x_n))$$

Larsson G, Maire M, Shakhnarovich G. FractalNet: Ultra-Deep Neural Networks without Residuals. ICLR 2017.

ODE: Infinite Layer Neural Network

Dynamic System ⟷ **Neural Network**

Continuous limit — 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.

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

WRN, ResNeXt, Inception-ResNet, PolyNet, SENet etc…… :

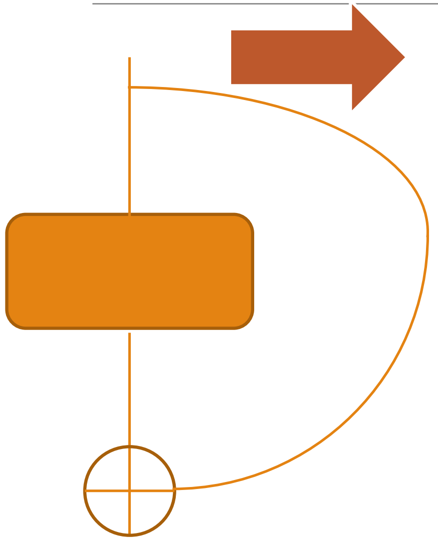
New scheme to Approximate the right hand side term

Why not change the way to discrete u_t?

Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Experiment

@Linear Multi-step Residual Network



$$x_t = f(x)$$

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

Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

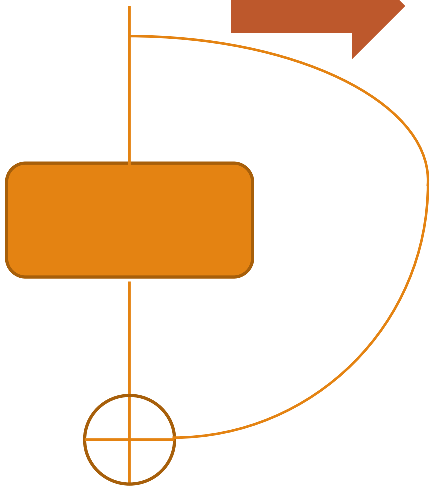
Experiment

@Linear Multi-step Residual Network

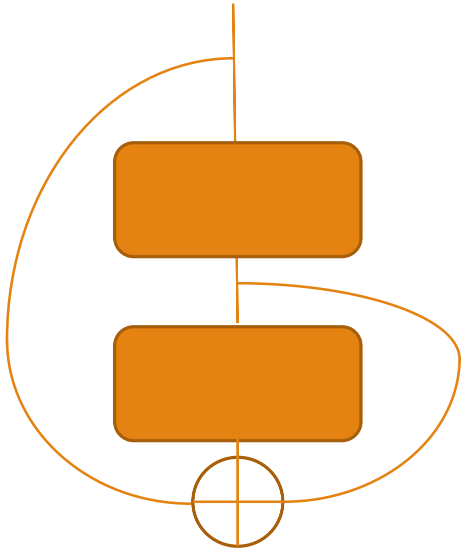
Linear Multi-step Scheme

$$\boldsymbol{x_t = f(x)}$$

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

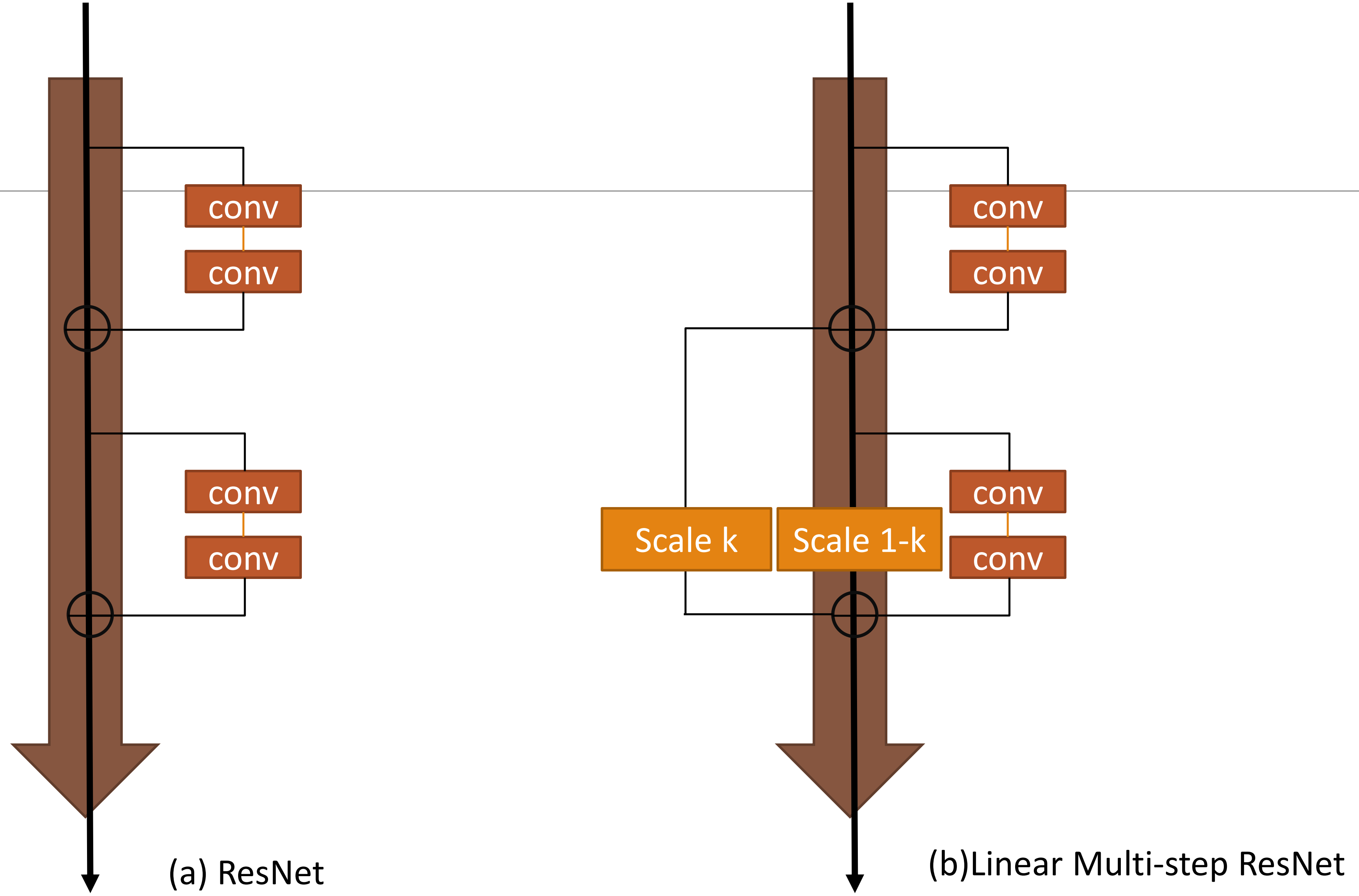


$$\boldsymbol{x_{n+1} = x_n + f(x_n)}$$

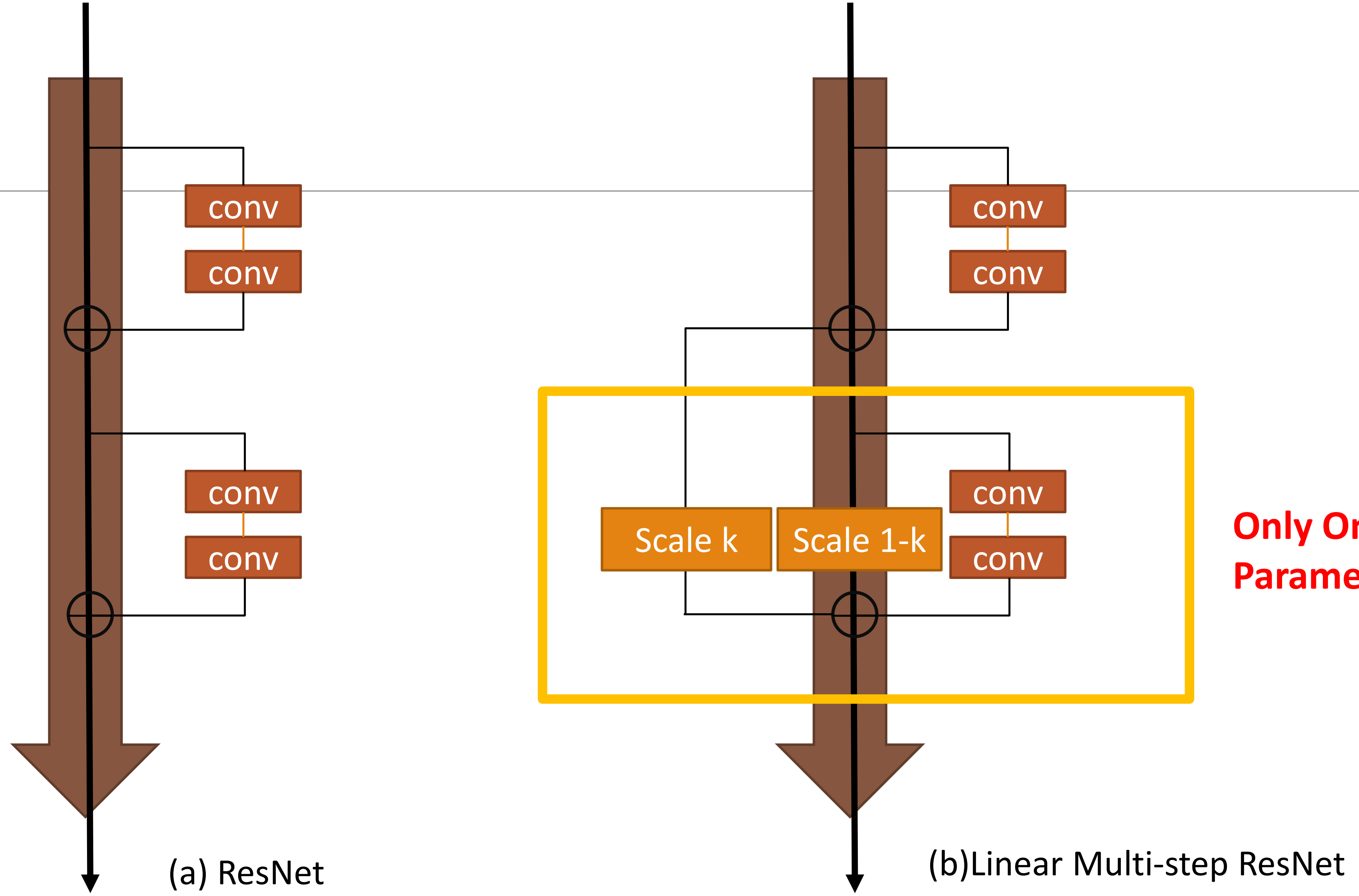


Linear Multi-step Residual Network

Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018



Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018



Only One More Parameter

Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Experiment

@Linear Multi-step Residual Network

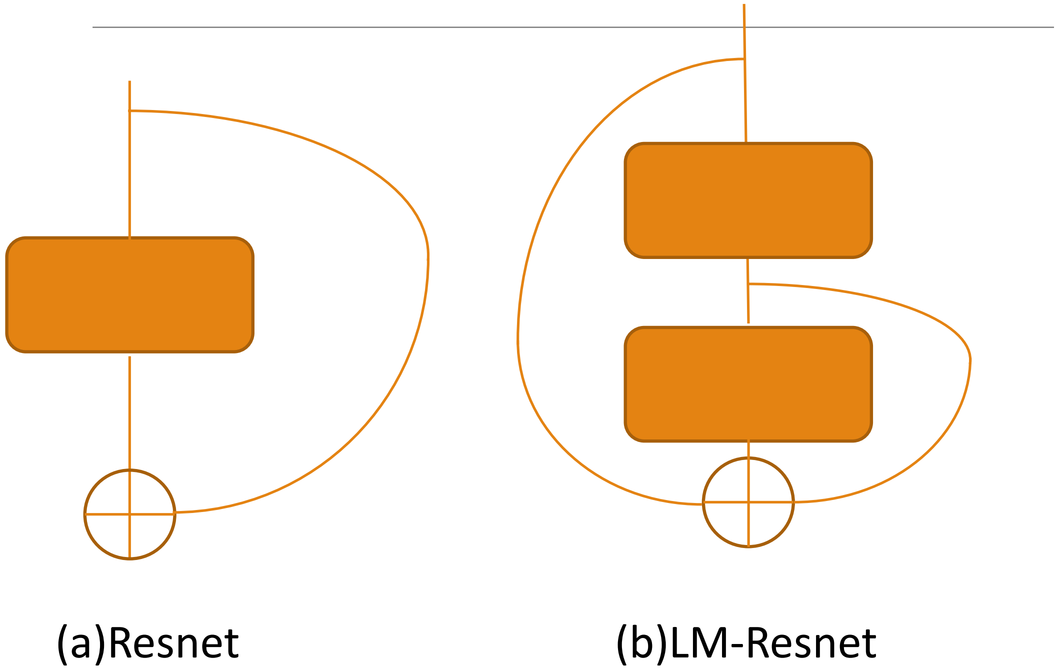
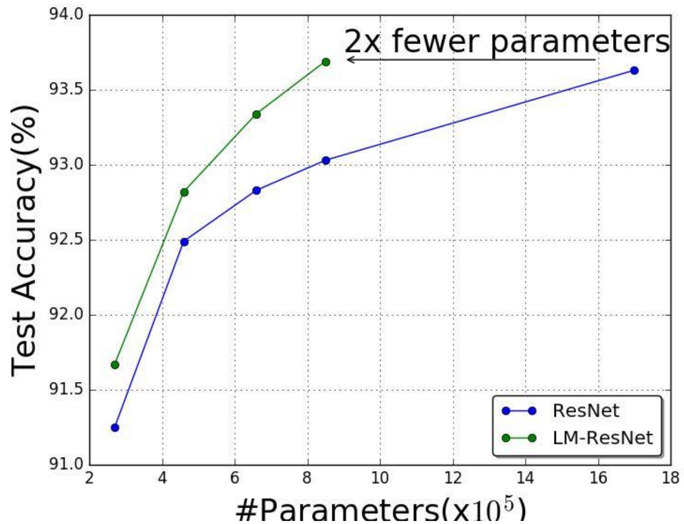


Table 2: Comparisons of LM-ResNet/LM-ResNeXt with other networks on CIFAR

Model	Layer	Error	Params	Dataset
ResNet (He et al. (2015b))	20	8.75	0.27M	CIFAR10
ResNet (He et al. (2015b))	32	7.51	0.46M	CIFAR10
ResNet (He et al. (2015b))	44	7.17	0.66M	CIFAR10
ResNet (He et al. (2015b))	56	6.97	0.85M	CIFAR10
ResNet (He et al. (2016))	110, pre-act	6.37	1.7M	CIFAR10
LM-ResNet (Ours)	20, pre-act	8.33	0.27M	CIFAR10
LM-ResNet (Ours)	32, pre-act	7.18	0.46M	CIFAR10
LM-ResNet (Ours)	44, pre-act	6.66	0.66M	CIFAR10
LM-ResNet (Ours)	56, pre-act	**6.31**	0.85M	CIFAR10



Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Experiment

@Linear Multi-step Residual Network

Table 2: Linear Multi-step Resnet Test On Cifar

Model	Layer	Accuracy	Params	Dataset
Resnet	20	91.25	0.27M	Cifar10
Resnet	32	92.49	0.46M	Cifar10
Resnet	44	92.83	0.66M	Cifar10
Resnet	56	93.03	0.85M	Cifar10
Resnet	110	93.63	1.7M	Cifar10
LM-Resnet(Ours)	20	91.67	0.27M	Cifar10
LM- Resnet(Ours)	32	92.82	0.46M	Cifar10
LM- Resnet(Ours)	44	92.98	0.66M	Cifar10
LM- Resnet(Ours)	56	**93.69**	0.85M	Cifar10
EM- Resnet(Ours)	40	91.75	0.27M	Cifar10
Resnet	110	72.24	1.7M	Cifar100
Resnet	164	75.67	2.55M	Cifar100
Resnet	1202	77.29	18.88M	Cifar100
ResneXt	29(8×64d)	82.23	34.4M	Cifar100
ResneXt	29(16×64d)	82.69	68.1M	Cifar100
LM-Resnet(Ours)	110	73.16	1.7M	Cifar100
LM-Resnet(Ours)	164	76.74	2.55M	Cifar100
LM-ResneXt(Ours)	29(8×64d)	82.51	34.4M	Cifar100
LM-ResneXt(Ours)	29(16×64d)	**83.21**	68.1M	Cifar100

Table 3: Single-crop error rate on ImageNet (validation set)

Model	Layer	top-1	top-5
ResNet (He et al. (2015b))	50	24.7	7.8
ResNet (He et al. (2015b))	101	23.6	7.1
ResNet (He et al. (2015b))	152	23.0	6.7
LM-ResNet (Ours)	50, pre-act	23.8	7.0
LM-ResNet (Ours)	101, pre-act	**22.6**	**6.4**

Explanation on the performance boost via *modified equations*

@Linear Multi-step Residual Network

ResNet

$$x_{n+1} = x_n + \Delta t f(x_n) \quad \Rightarrow \quad \dot{u} + \frac{\Delta t}{2}\ddot{u}_n = f(u)$$

LM-ResNet

$$x_{n+1} = (1 - k_n)x_n + k_n x_{n-1} + \Delta t f(x_n) \quad \Rightarrow \quad (1 + k_n)\dot{u} + (1 - k_n)\frac{\Delta t}{2}\ddot{u}_n = f(u)$$

[1] Dong B, Jiang Q, Shen Z. Image restoration: wavelet frame shrinkage, nonlinear evolution PDEs, and beyond. Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal 2017.
[2] Su W, Boyd S, Candes E J. A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights. Advances in Neural Information Processing Systems, 2015.
[3] A. Wibisono, A. Wilson, and M. I. Jordan. A variational perspective on accelerated methods in optimizationProceedings of the National Academy of Sciences 2016.

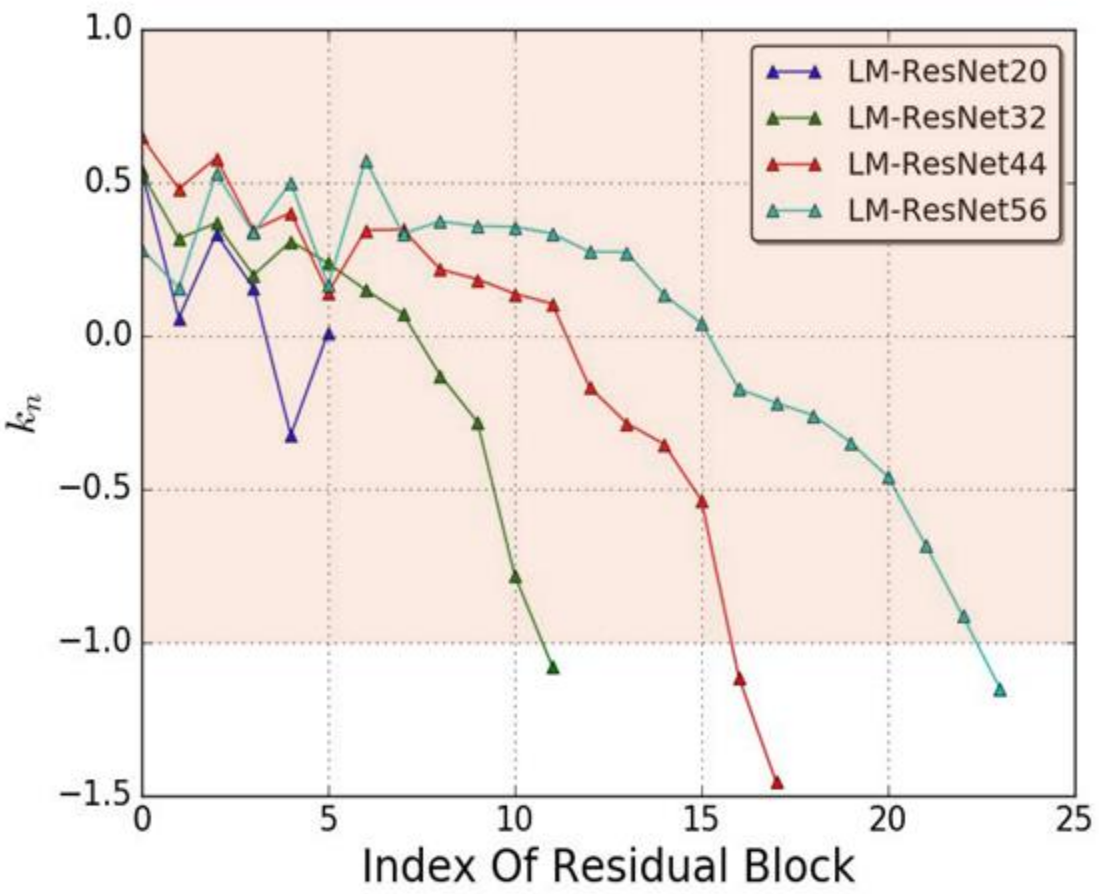
Plot The Momentum

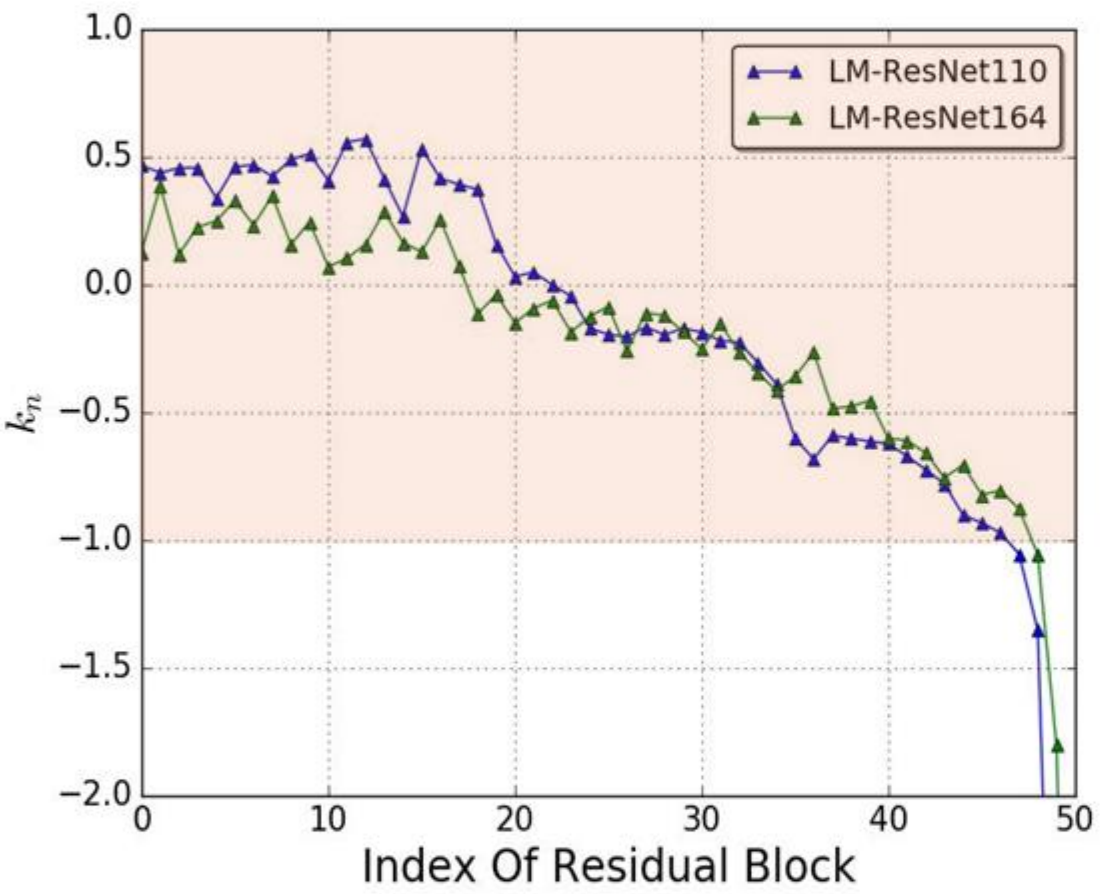
$$x_{n+1} = (1 - k_n)x_n + k_n x_{n-1} + \Delta t f(x_n)$$

@Linear Multi-step Residual Network

Learn A Momentum

$$(1 + k_n)\,\dot{u} + (1 - k_n)\frac{\Delta t}{2}\ddot{u}_n + o(\Delta t^3) = f(u)$$





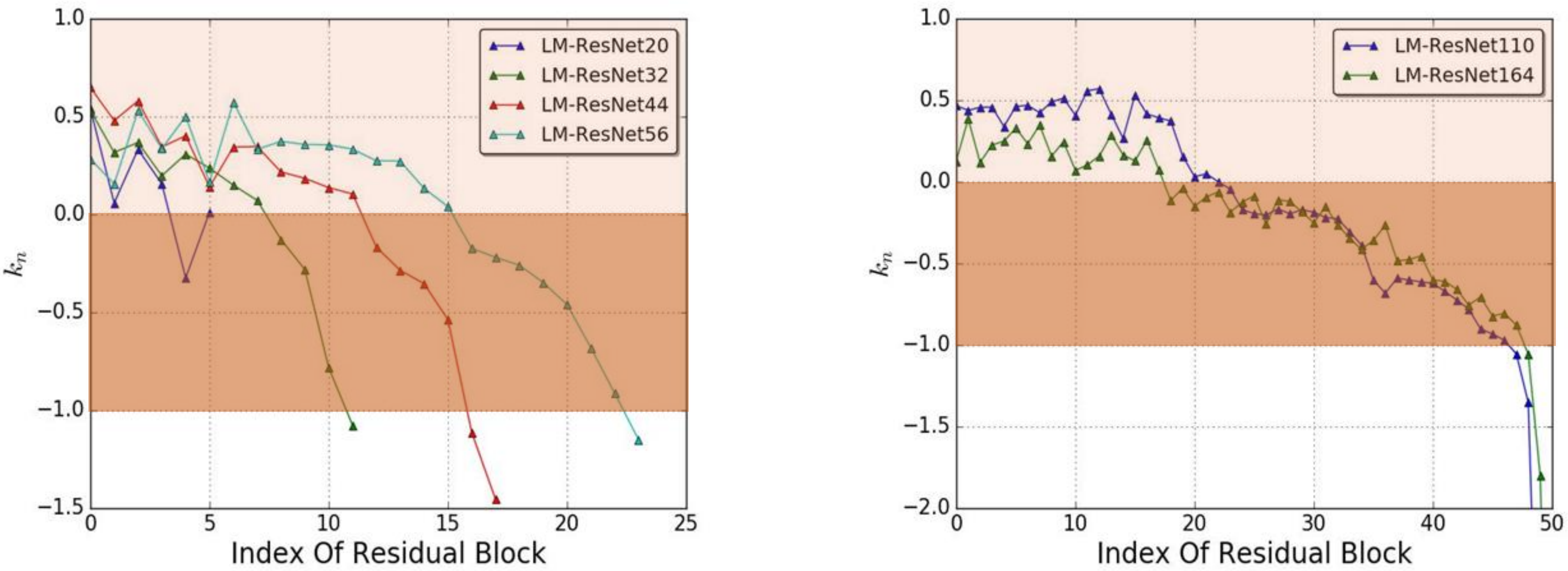
Plot The Momentum

@Linear Multi-step Residual Network

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

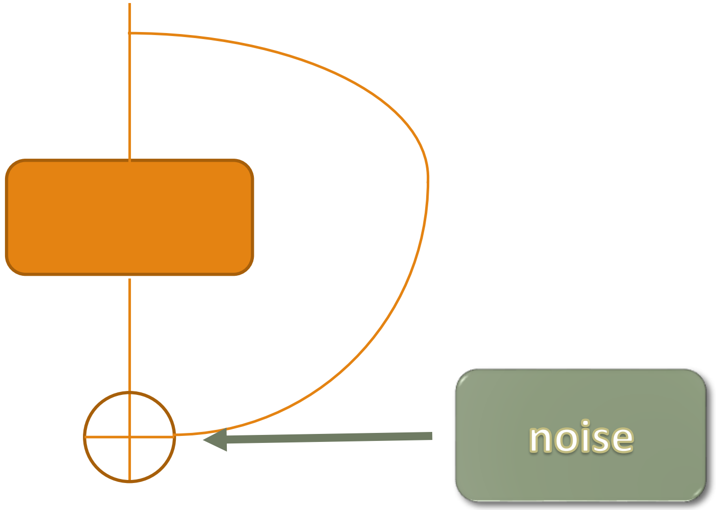
Learn A Momentum

$$(1 + k_n)\,\dot{u} + (1 - k_n)\frac{\Delta t}{2}\ddot{u}_n + o(\Delta t^3) = f(u)$$

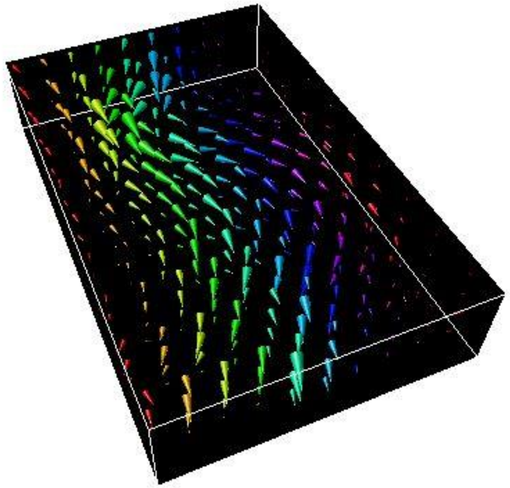


Connection to stochastic dynamic

Noise can avoid overfit?



Dynamic System



Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Connection to stochastic dynamic

Shake-Shake regularization

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

$$= x_n + f_2(x_n) + \frac{1}{2}\big(f_1(x_n) - f_2(x_n)\big) + (\eta - \frac{1}{2})\big(f_1(x_n) - f_2(x_n)\big)$$

$$\frac{1}{\sqrt{12}}(f_1(X) - f_2(X)) \odot [\mathbf{1}_{N\times 1}, \mathbf{0}_{N,N-1}]dB_t$$

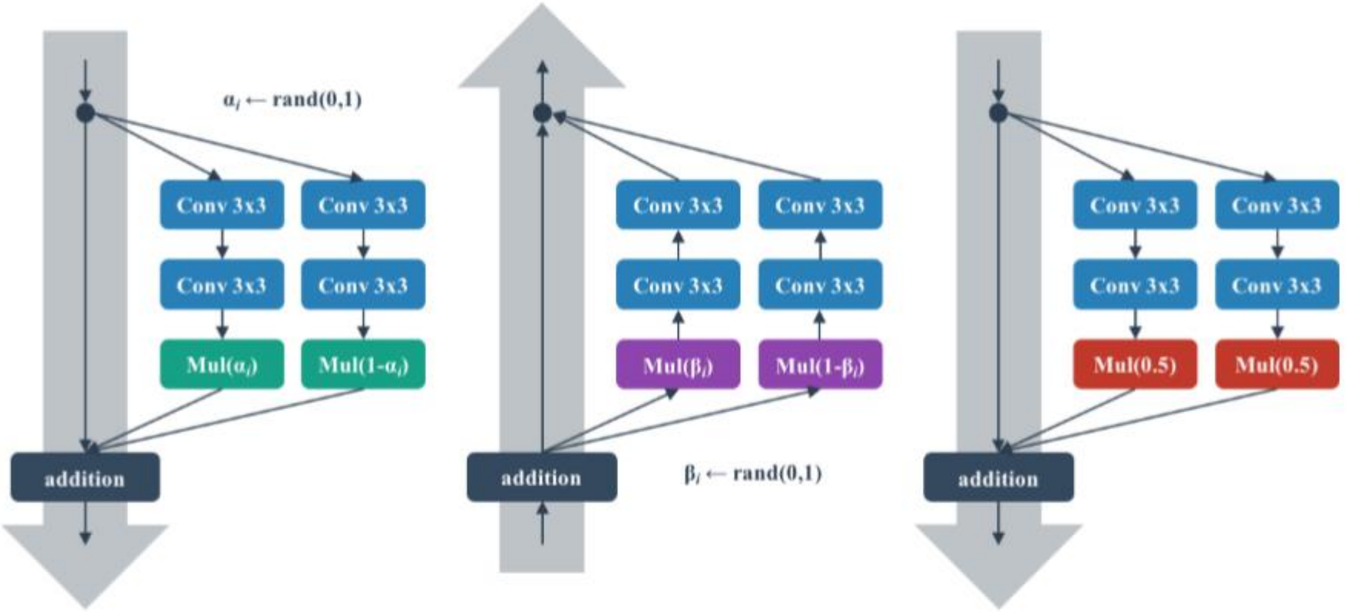


Figure 1: **Left:** Forward training pass. **Center:** Backward training pass. **Right:** At test time.

Gastaldi X. Shake-Shake regularization. ICLR Workshop Track2017.

Apply data augmentation techniques to internal representations.

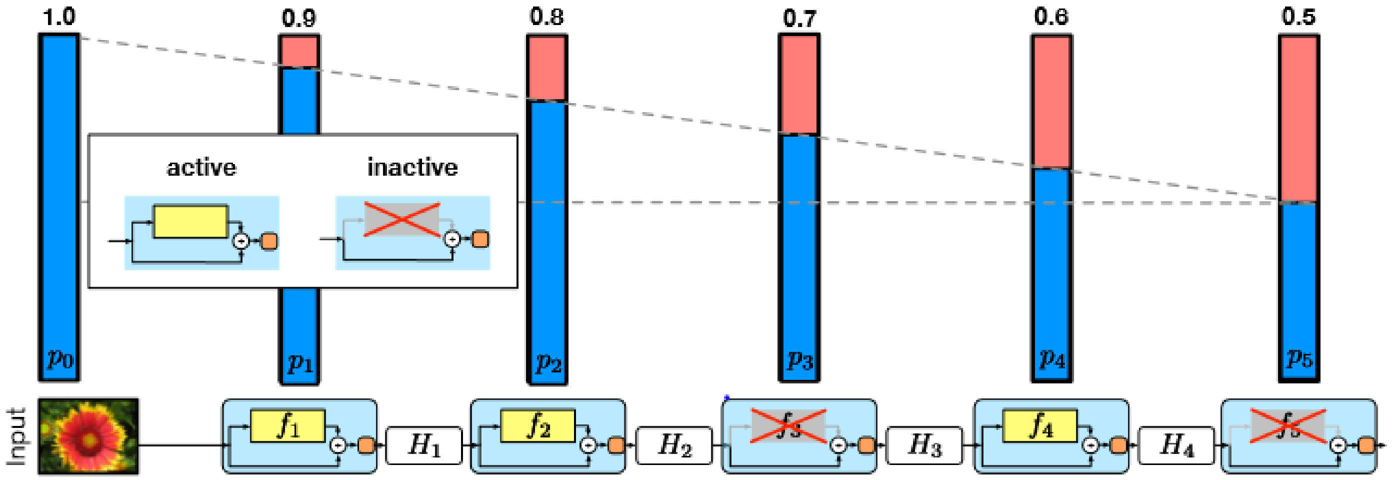
Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Connection to stochastic dynamic

Deep Networks with Stochastic Depth

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

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



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

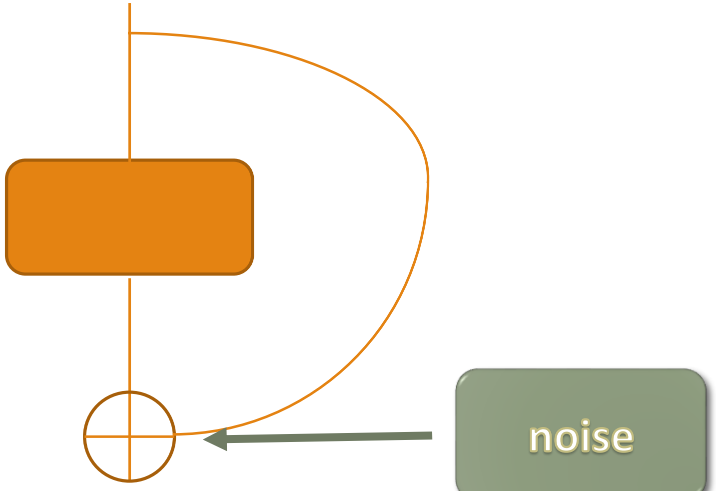
Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

Huang G, Sun Y, Liu Z, et al. Deep Networks with Stochastic Depth ECCV2016.

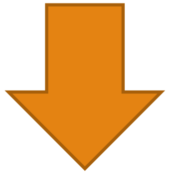
Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Connection to stochastic dynamic

Noise can avoid overfit?



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



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

$$\boldsymbol{E_{data}}(loss(X(T))$$

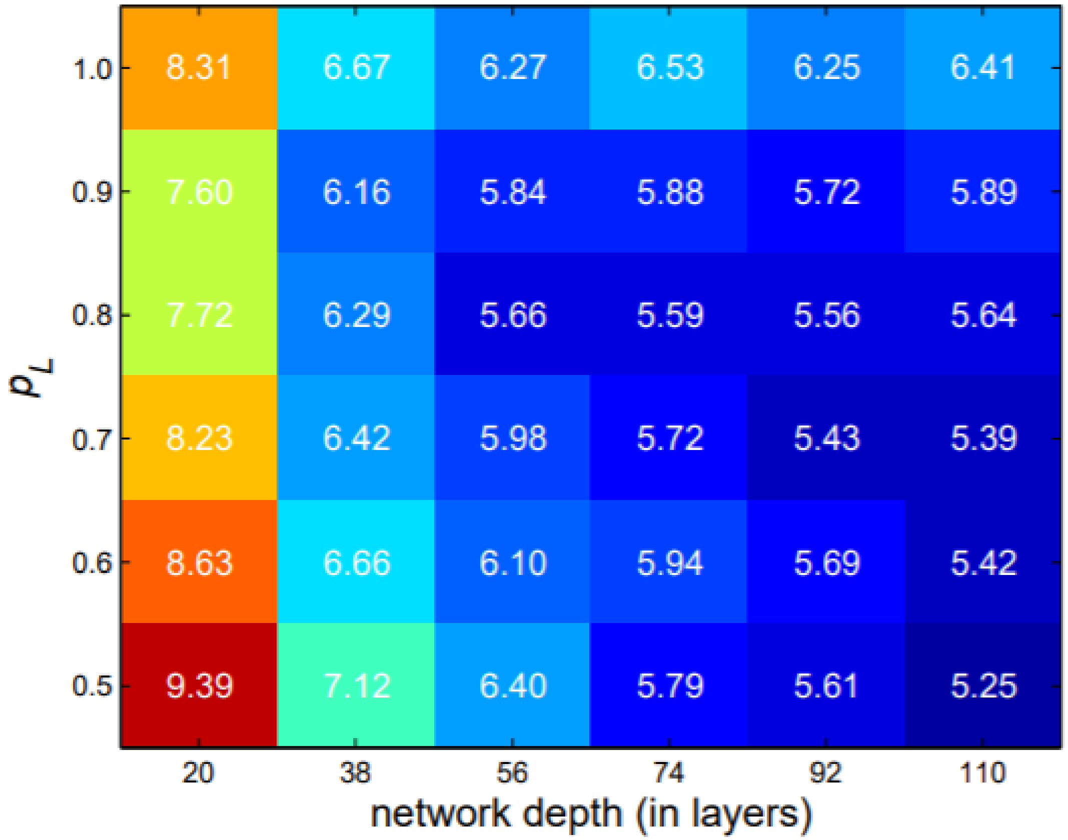
Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Connection to stochastic dynamic

Deep Networks with Stochastic Depth

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

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

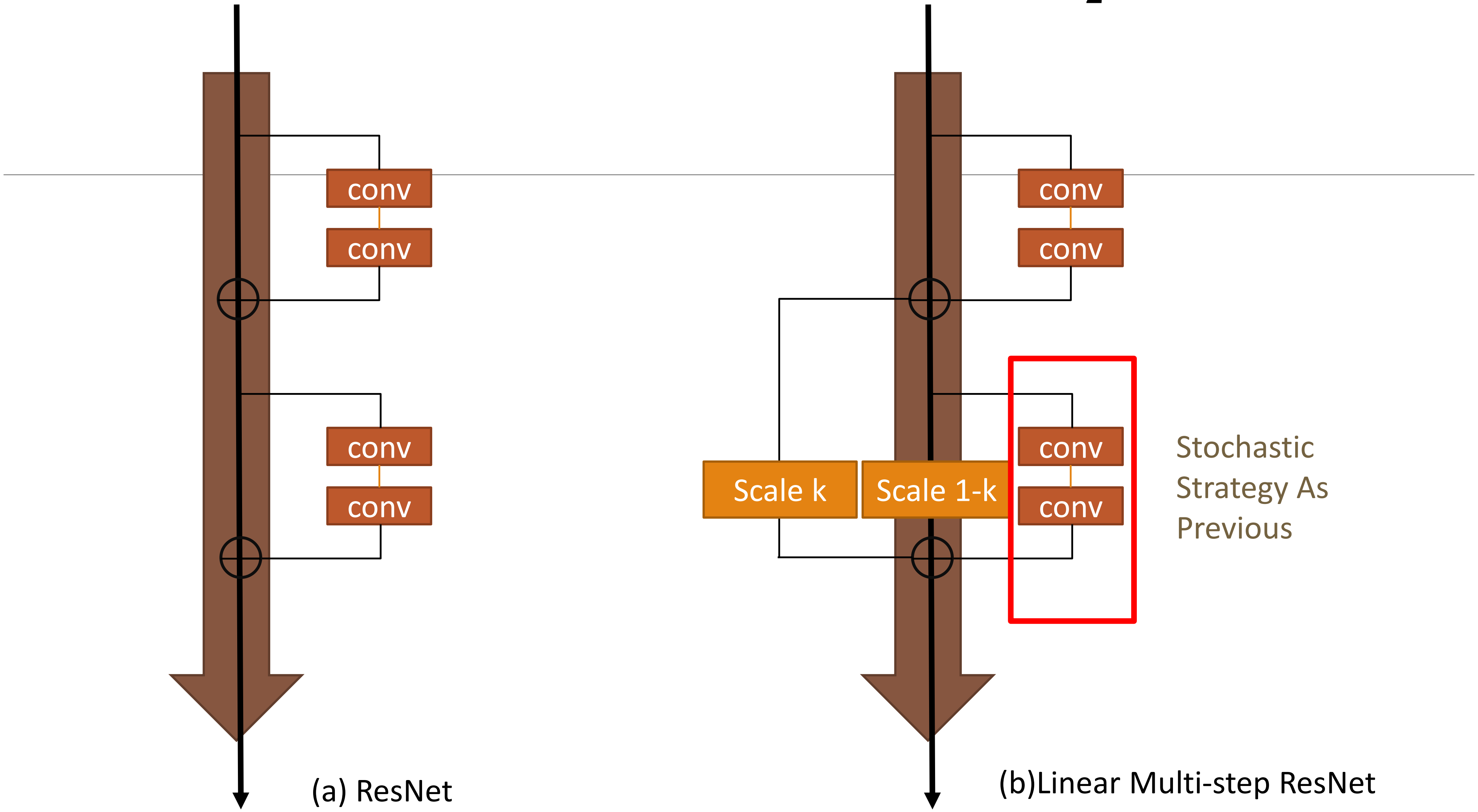


To reduce the effective length of a neural network during training, we randomly skip layers entirely.

Huang G, Sun Y, Liu Z, et al. Deep Networks with Stochastic Depth ECCV2016.

Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

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



(a) ResNet

(b)Linear Multi-step ResNet

Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

Experiment

@Linear Multi-step Residual Network

Table 4: Test on stochastic training strategy on CIFAR10

Model	Layer	Training Strategy	Error
ResNet(He et al. (2015b))	110	Original	6.61
ResNet(He et al. (2016))	110,pre-act	Orignial	6.37
ResNet(Huang et al. (2016b))	56	Stochastic depth	5.66
ResNet(Our Implement)	56,pre-act	Stochastic depth	5.55
ResNet(Huang et al. (2016b))	110	Stochastic depth	5.25
ResNet(Huang et al. (2016b))	1202	Stochastic depth	4.91
ResNet(Ours)	110,pre-act	Gaussian noise (noise level = 0.001)	5.52
LM-ResNet(Ours)	56,pre-act	Stochastic depth	5.14
LM-ResNet(Ours)	110,pre-act	Stochastic depth	**4.80**

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

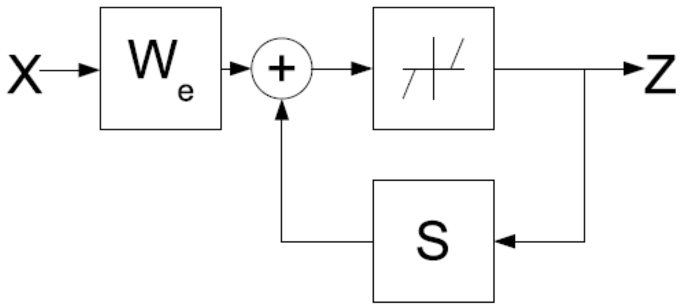
Lu, Yiping, et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations." ICML 2018

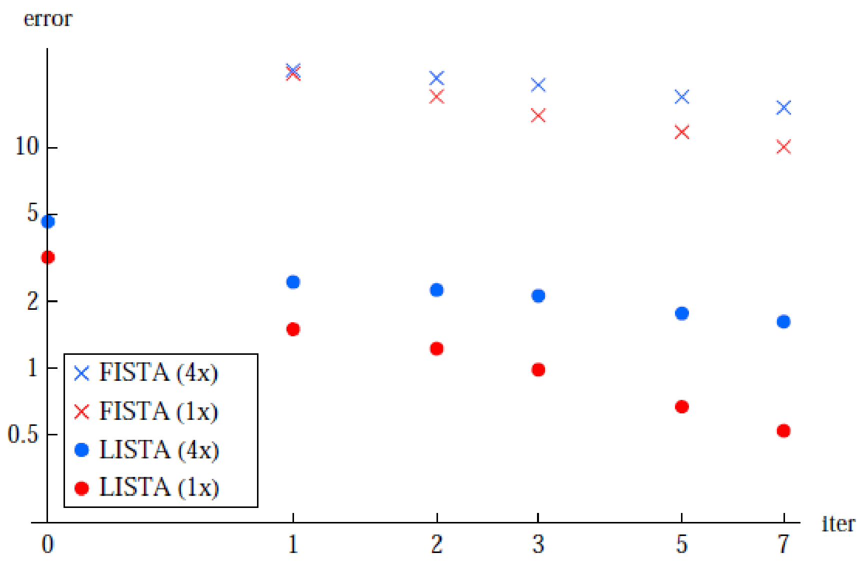
Earlier Evidence: LISTA

Unrolled Dynamics

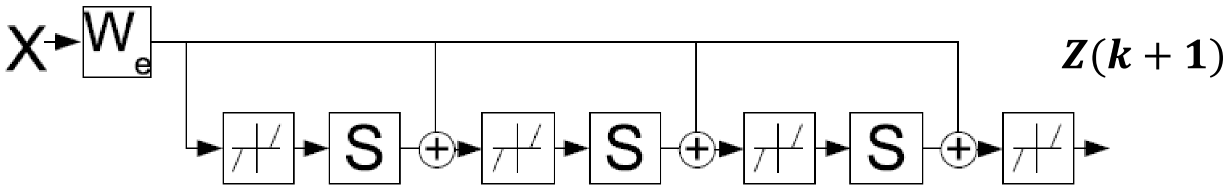
$$Z(k+1) = h_\theta(W_e X + SZ(k)),\ Z(0) = 0$$

ISTA





Unrolling

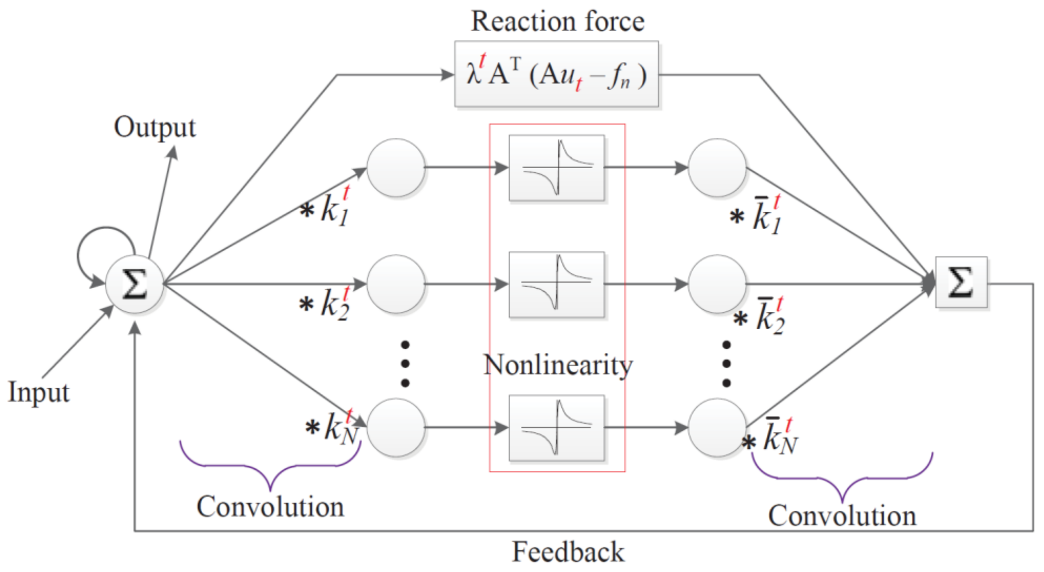


LISTA

Gregor, K., and LeCun, Y. Learning fast approximations of sparse coding. In ICML 2010 (pp. 399-406).

Earlier Evidence: TRD

Unrolled Dynamics



$$u_t = u_{t-1} - \left(\sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1}) + \lambda^t (u_{t-1} - f_n) \right)$$

Learning a diffusion process for denoising

Method	σ 15	σ 25	St.	$\sigma = 15$ $\text{TRD}_{5\times5}$	$\sigma = 15$ $\text{TRD}_{7\times7}$
BM3D	31.08	28.56	2	31.14	31.30
LSSC	31.27	28.70	5	31.30	**31.42**
EPLL	31.19	28.68	8	31.34	**31.43**
opt-MRF	31.18	28.66		$\sigma = 25$	
RTF_5	–	28.75		$\text{TRD}_{5\times5}$	$\text{TRD}_{7\times7}$
WNNM	31.37	28.83	2	28.58	28.77
$\text{CSF}^5_{5\times5}$	31.14	28.60	5	28.78	**28.92**
$\text{CSF}^5_{7\times7}$	31.24	28.72	8	28.83	**28.95**

Average PSNR among a dataset with 68 images

Chen Y, Yu W, Pock T. On learning optimized reaction diffusion processes for effective image restoration CVPR2015

Recent Evidence: Optimization Algorithm Inspired DNN

- Deep neural network as optimization algorithm:

$$\boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{W}\boldsymbol{x}_k) \Longleftrightarrow \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \nabla \boldsymbol{F}(\boldsymbol{x}_k)$$

- Faster algorithm result in better deep neural network:

Heavy Ball Net:

$$x_{k+1} = T(x_k) + x_k - x_{k-1}$$

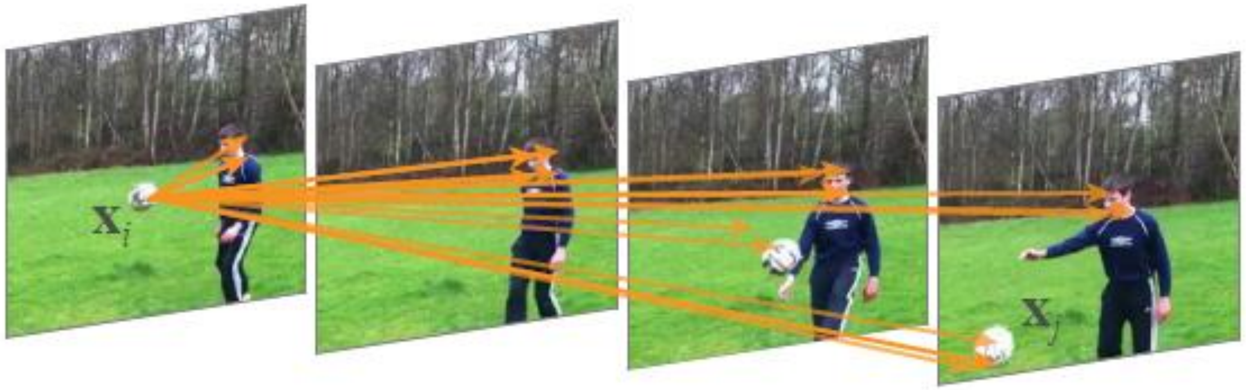
Accelerated GD Net:

$$x_{k+1} = \sum_{j=0}^{k} \alpha_{k+1,j} T(x_j) + \beta \left(x_k - \sum_{j=0}^{k} h_{k+1,j} x_j \right)$$

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

Recent Evidence: Nonlocal DNN

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



	model	top-1	top-5
R50	baseline	71.8	89.7
	1-block	72.7	90.5
	5-block	73.8	91.0
	10-block	**74.3**	**91.2**
R101	baseline	73.1	91.0
	1-block	74.3	91.3
	5-block	**75.1**	**91.7**
	10-block	**75.1**	91.6

(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.

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

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

Nonlocal Block: $[\mathcal{F}(Z^k ; W^k)]_i = \frac{W_Z^k}{\mathcal{C}_i(Z^k)} \sum_{\forall j} \omega(Z_i^k, Z_j^k)(W_g^k Z_j^k)$

Instability when using multiple blocks!

X. Wang, et al. "Non-local neural networks." CVPR 2018.

Recent Evidence: Nonlocal DNN as Nonlocal Diffusion

Design a new **stable** block

$$Z_i^{n+1} := Z_i^n + \frac{W^n}{\mathcal{C}_i(X)} \sum_{\forall j} \omega(X_i, X_j)(Z_j^n - Z_i^n)$$

Model		Error (%)
Same Place	baseline	8.19
	2-block (original)	7.83
	3-block (original)	8.28
	4-block (original)	15.02
	2-block (proposed)	7.74
	3-block (proposed)	7.62
	4-block (proposed)	7.37
	5-block (proposed)	**7.29**
	6-block (proposed)	7.55
Different Places	3-block (original)	8.07
	3-block (proposed)	**7.33**

- Nonlocal diffusion
- Nonlocal Neural Network
- Nonlocal Markov Jump Process

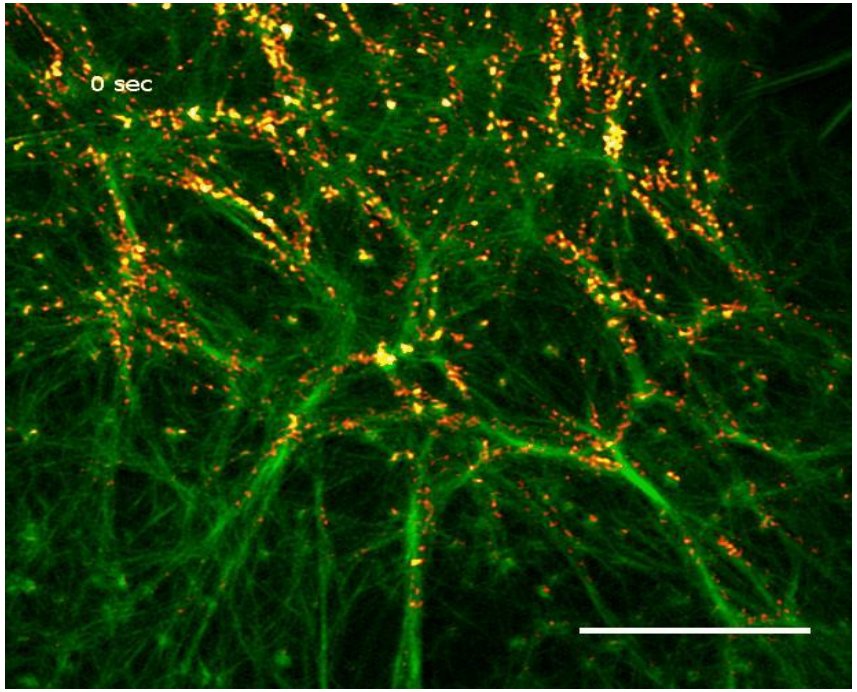
Tao Y, Sun Q, Du Q, et al. Nonlocal Neural Networks, Nonlocal Diffusion and Nonlocal Modeling. NeurIPS 2018.

Deep Neural Networks and Numerical PDE

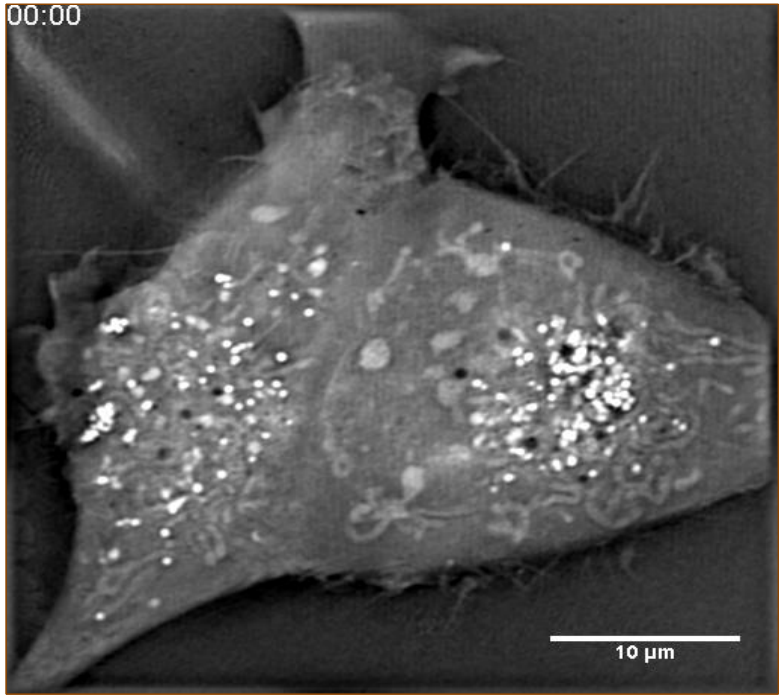
DATA DRIVEN PHYSIC LAW DISCOVERY

PDE-Net: Learning PDEs from Data

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



Dynamics of actin in Immunocytoskeleton

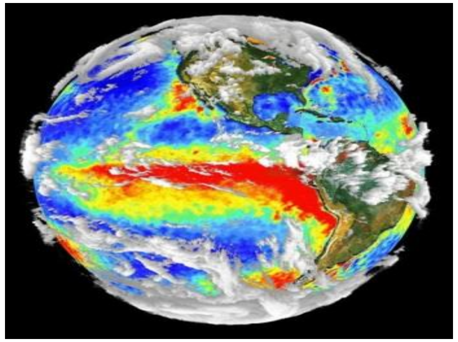


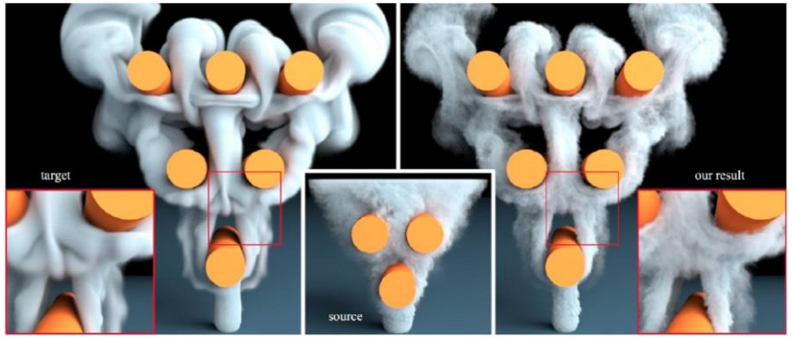
Dynamics of Mitochondria

Credit: Kebin Shi, Physics@PKU

PDE-Net: Learning PDEs from Data

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





S. Sato et al., Siggraph 2018

Preliminary attempt:

- Combine deep learning and numerical PDEs

Objectives:

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

Long Z et al. PDE-Net: Learning PDEs from Data. ICML 2018.

PDE-Net: Learning PDEs from Data

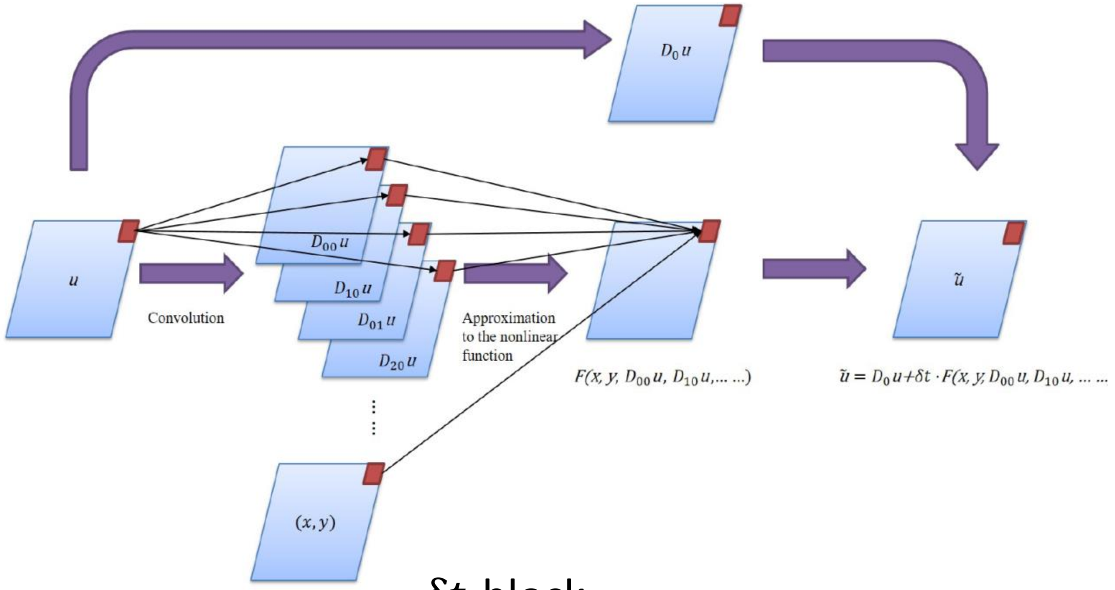
PDE-Net: a flexible and transparent deep network

Assuming :

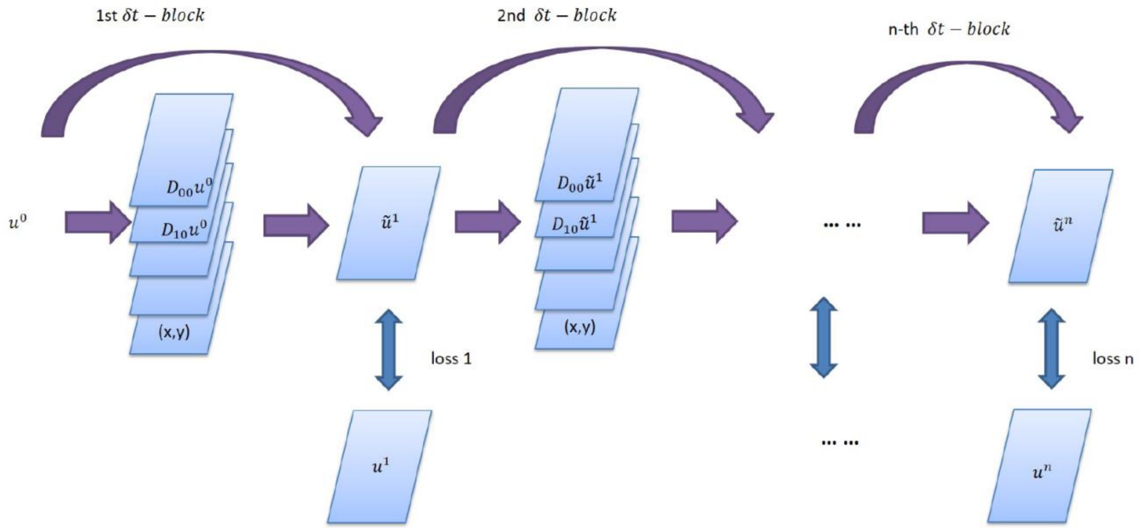
$$\frac{\partial u}{\partial t} = F(x, u, \nabla u, \nabla^2 u, \dots)$$

Prior knowledge on F:

- Type of the PDE
- Maximum order



δt-block



PDE-Net: multiple δt-blocks

PDE-Net: Learning PDEs from Data

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

$$\begin{pmatrix} 0 & 0 & \star \\ 1 & \star & \star \\ \star & \star & \star \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & \star \\ 0 & \star & \star \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

1st order learnable	2st 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)

$$\begin{cases} \frac{\partial u}{\partial t} &= a(x,y)u_x + b(x,y)u_y + cu_{xx} + du_{yy} \\ u|_{t=0} &= u_0(x,y), \end{cases}$$

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

$$b(x,y) = 2(\cos(y) + \sin(x)) + 0.8,$$

$$c = 0.2 \text{ and } d = 0.3$$

- Diffusion with a nonlinear source (nonlinear)

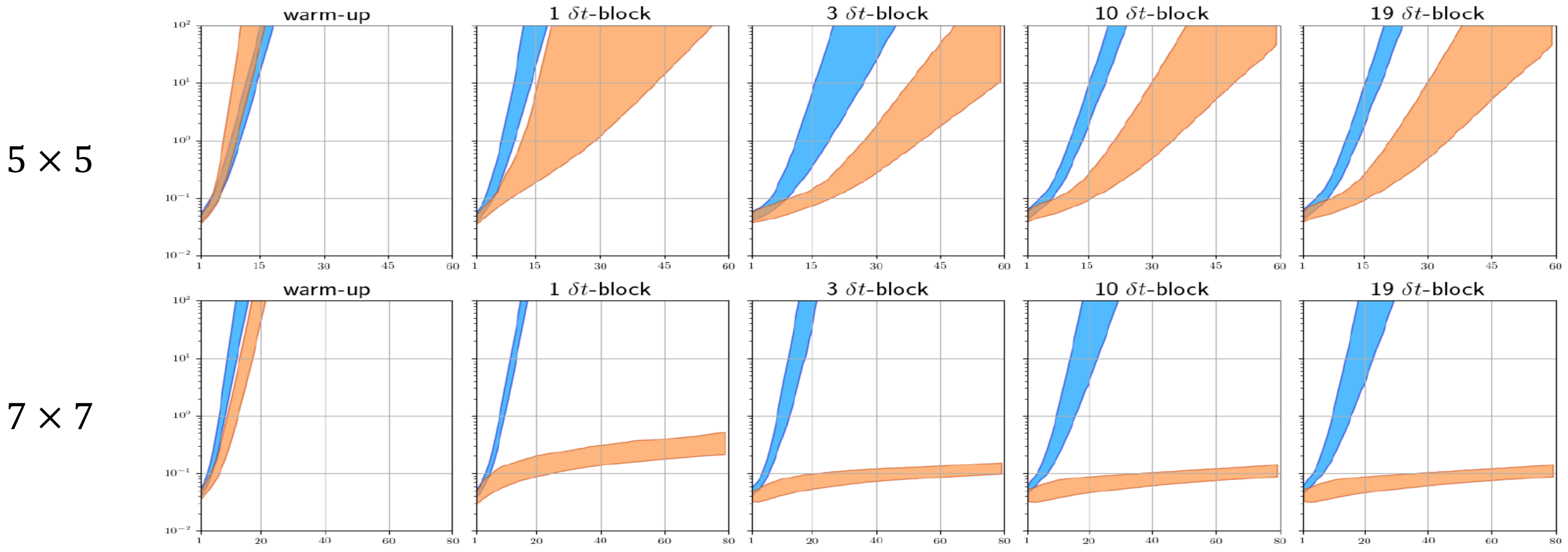
$$\begin{cases} \frac{\partial u}{\partial t} &= c\Delta u + f_s(u) \\ u|_{t=0} &= u_0(x,y), \end{cases} \qquad c = 0.3 \text{ and } f_s(u) = 15\sin(u)$$

- 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)

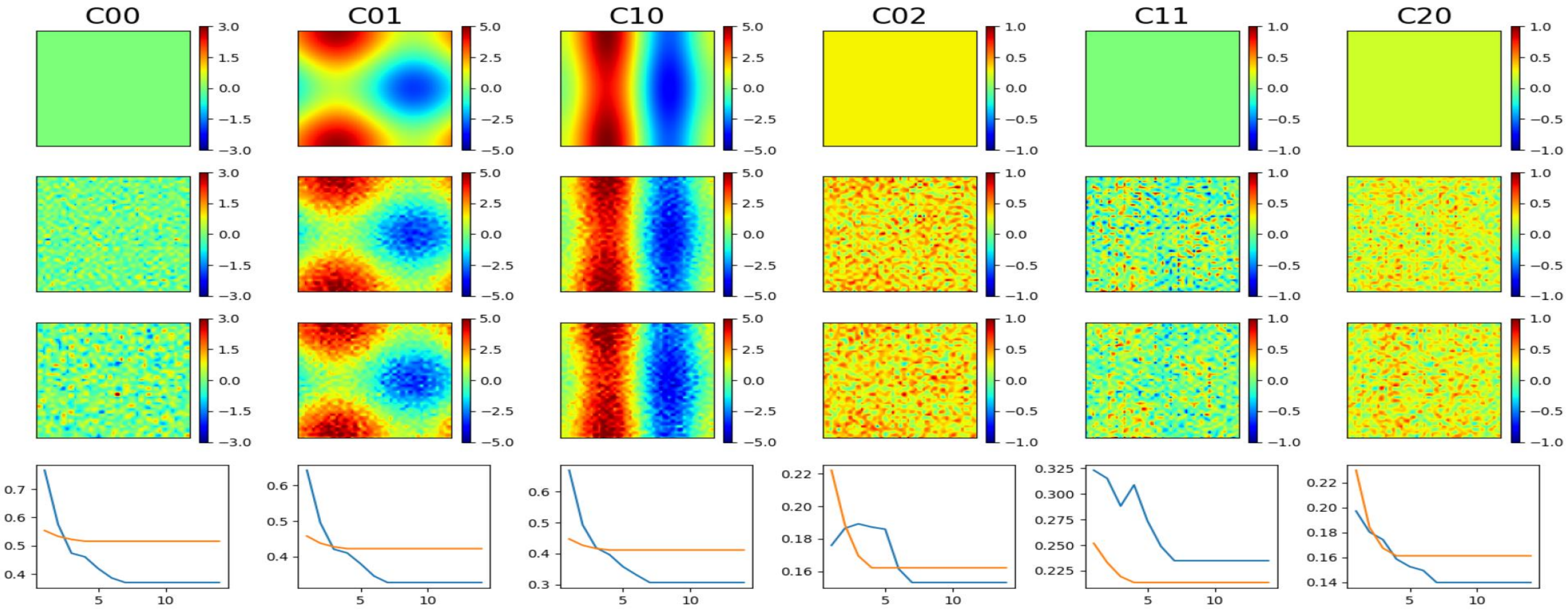


Learnable filters (orange) v.s. frozen filters (blue) in prediction

PDE-Net: Learning PDEs from Data

Numerical experiments: results

- Model estimation: linear

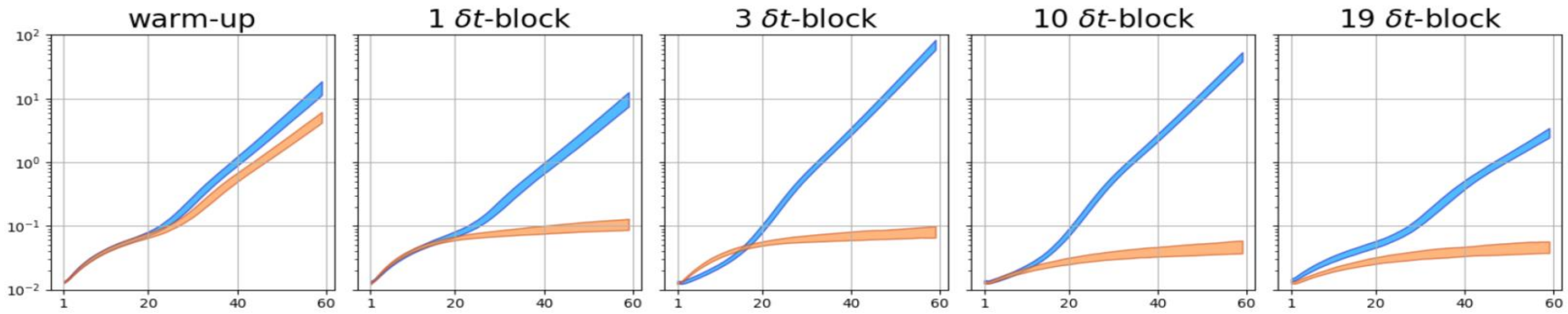


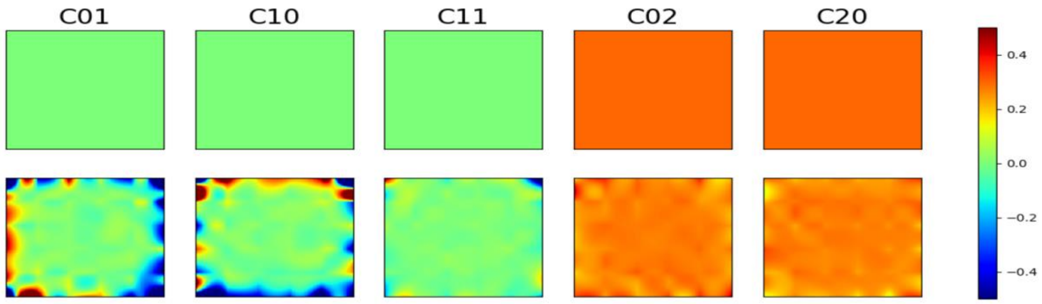
Long Z et al. PDE-Net: Learning PDEs from Data. ICML 2018.

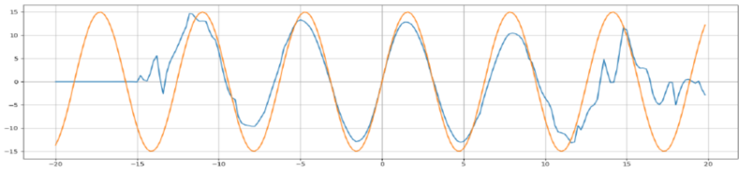
PDE-Net: Learning PDEs from Data

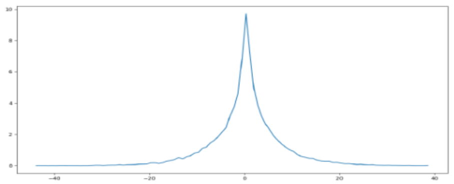
Numerical experiments: results

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









Long Z et al. PDE-Net: Learning PDEs from Data. ICML 2018.

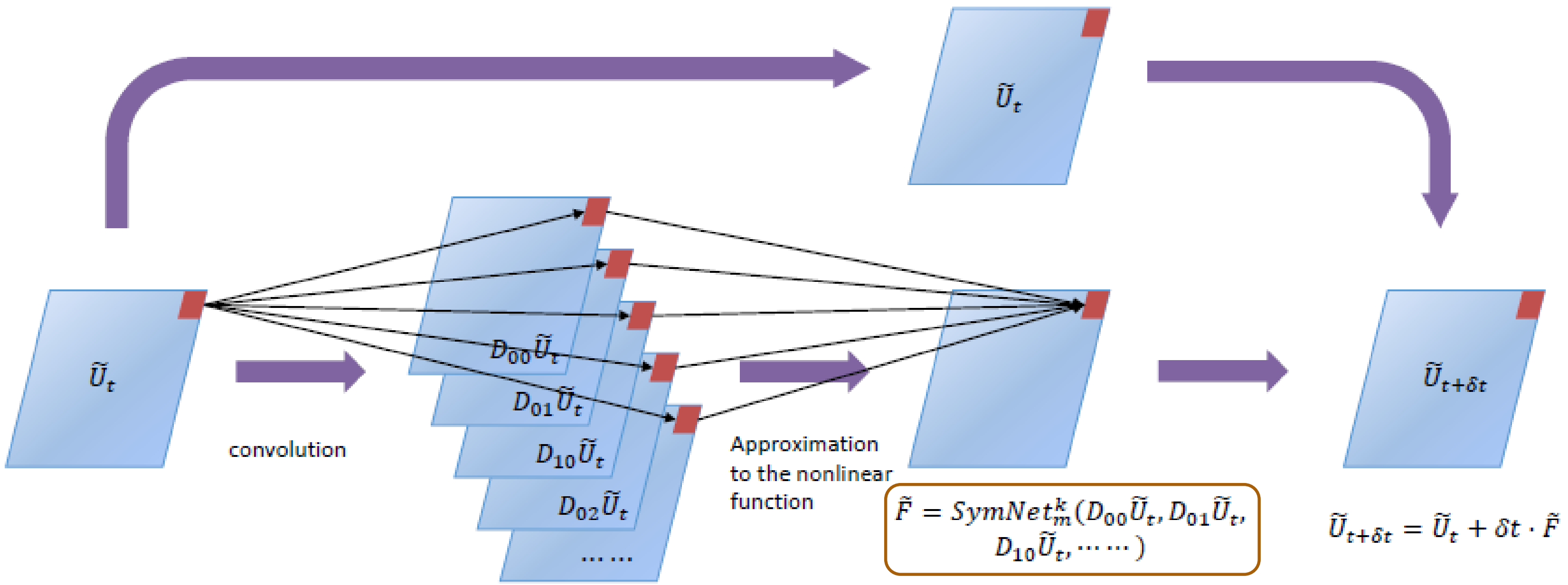
PDE-Net 2.0: Numeric-Symbolic Hybrid Representation

Symbolic network (granting transparency)

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

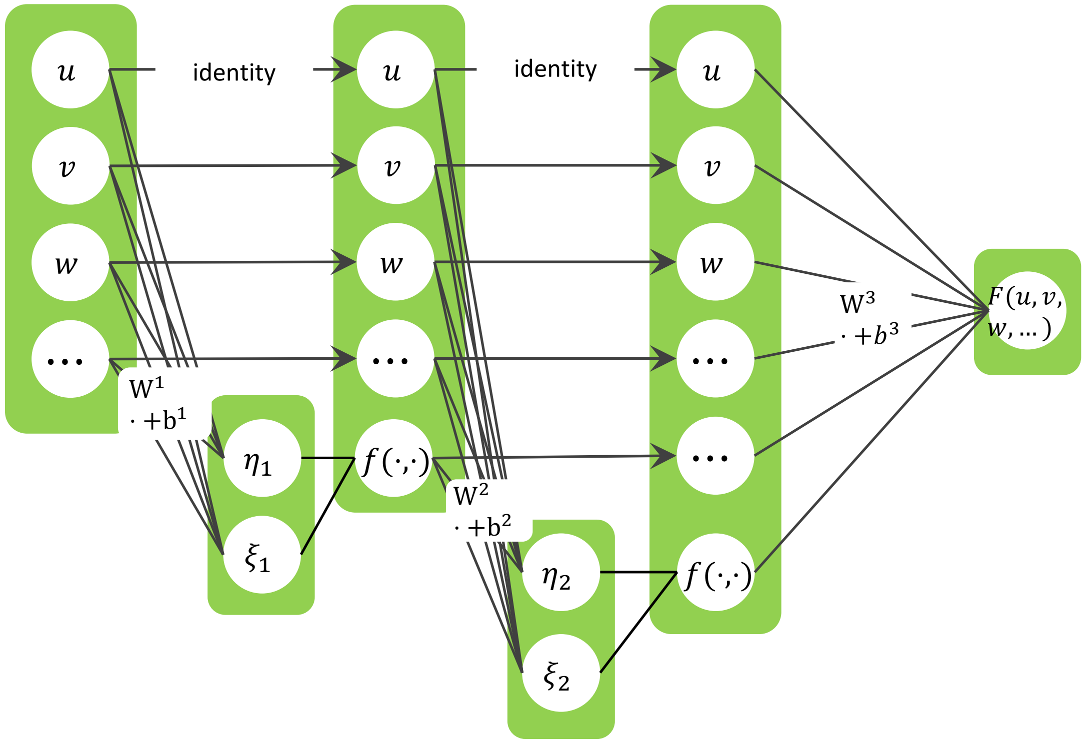
Prior knowledge on F:

- Addition and multiplication of the terms;
- Maximum order.



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

- *Sahoo, S. S.; Lampert, C. H. & Martius, G. ICML 2018.*
- Martius, Georg, and Christoph H. Lampert. *arXiv preprint arXiv:1610.02995* (2016).

PDE-Net 2.0: Numeric-Symbolic Hybrid Representation

Weaker assumption on F: unknown type

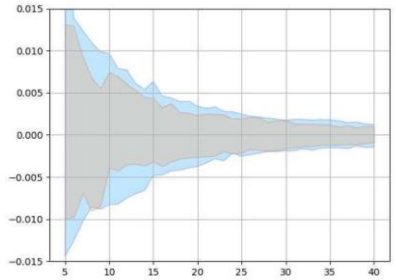
Burger's Equation

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} = \nu \nabla^2 \boldsymbol{u}$$

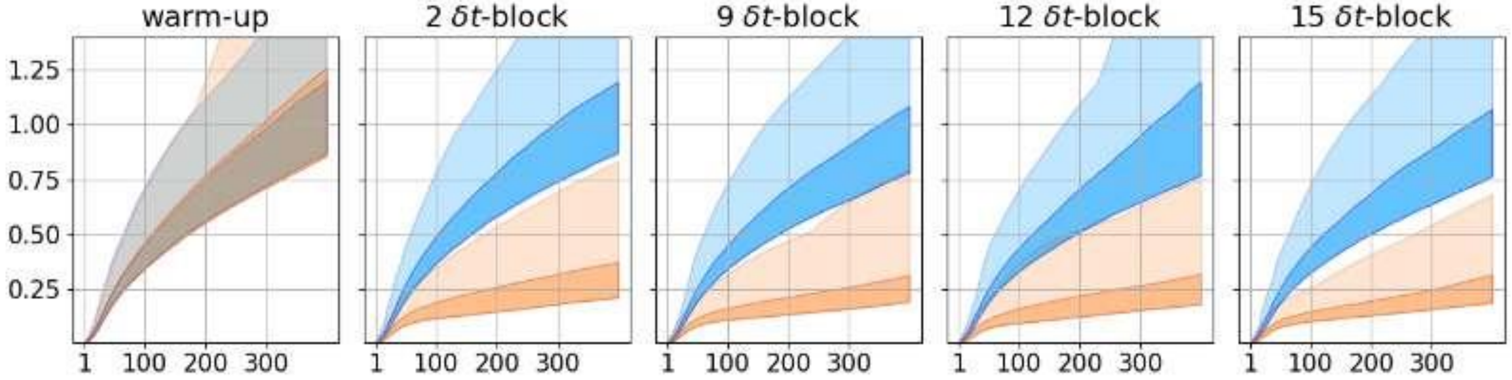
$$\nu = 0.05$$

Correct PDE	$u_t = -uu_x - vu_y + 0.05(u_{xx} + u_{yy})$ $v_t = -uv_x - vv_y + 0.05(v_{xx} + v_{yy})$
Frozen-PDE-Net 2.0	$u_t = -0.906uu_x - 0.901vu_y + 0.033u_{xx} + 0.037u_{yy}$ $v_t = -0.907vv_y - 0.902uv_x + 0.039v_{xx} + 0.032v_{yy}$
PDE-Net 2.0	$u_t = -0.986uu_x - 0.972u_yv + 0.054u_{xx} + 0.052u_{yy}$ $v_t = -0.984uv_x - 0.982vv_y + 0.055v_{xx} + 0.050v_{yy}$

Model recovery



Remainer weights of $\boldsymbol{u}, \boldsymbol{v}$



Prediction

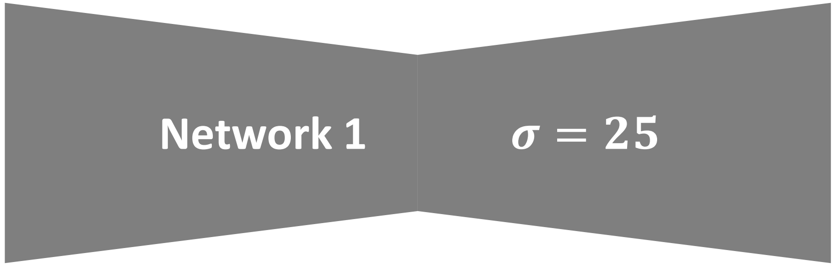
Long Z, Lu Y and Dong B. "PDE-Net 2.0: Learning PDEs from Data with A Numeric-Symbolic Hybrid Deep Network", arXiv:1812.04426, 2018

Application In Image Processing

BLIND IMAGE RESTORATION

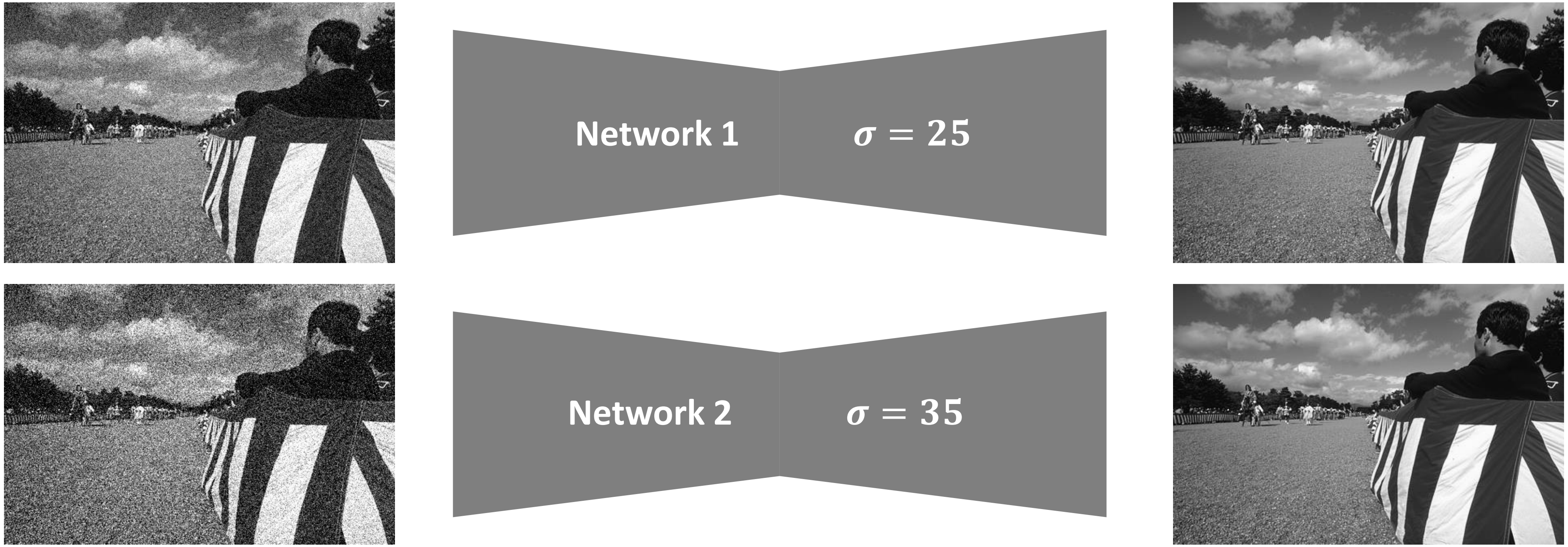
Deep Learning For Restoration

One Noise Level One Net



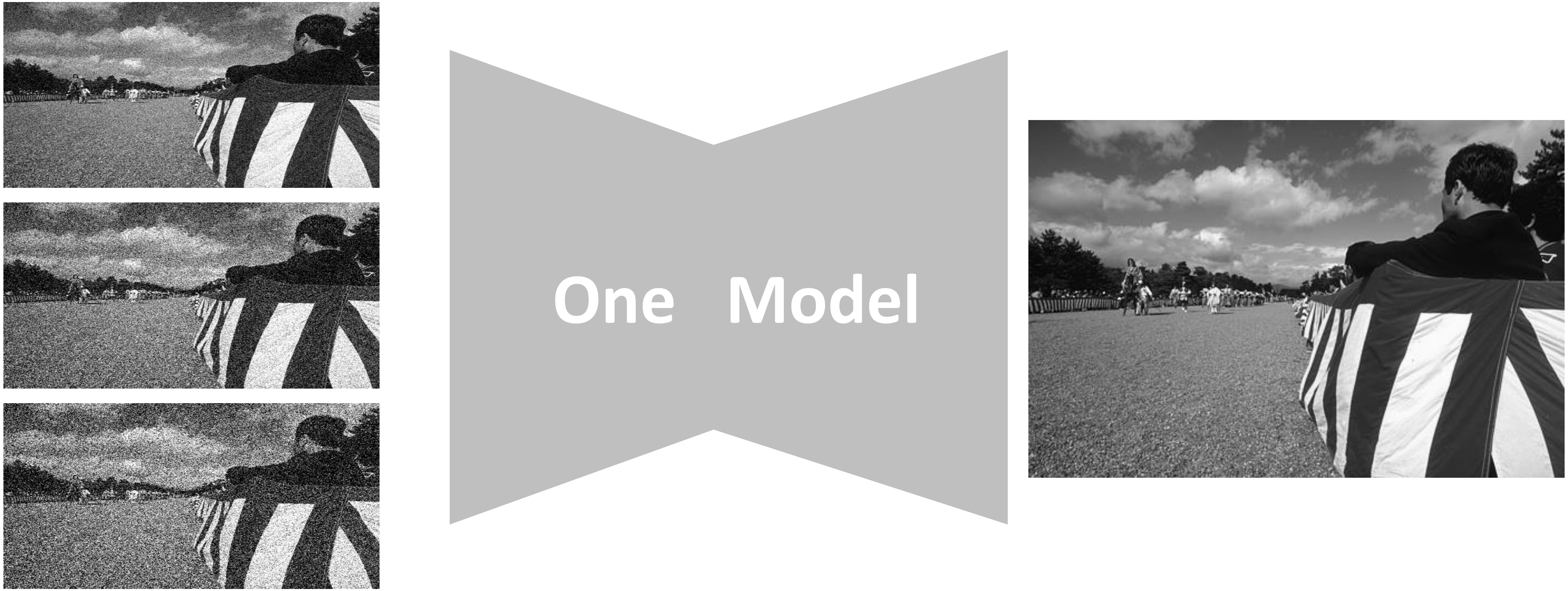
Deep Learning For Restoration

One Noise Level One Net



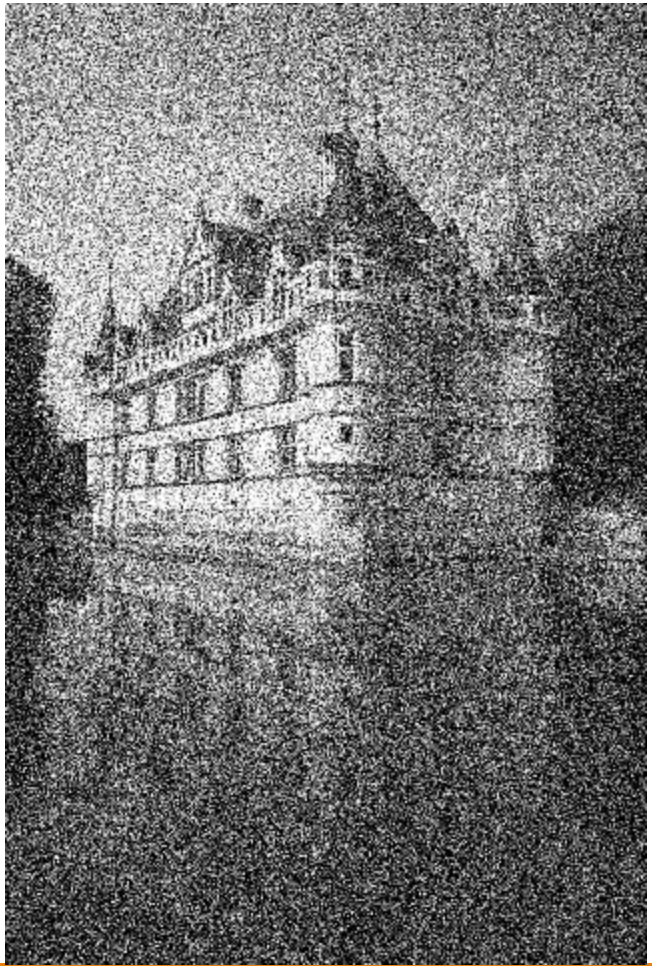
What We Want

One Model For All Noise Level

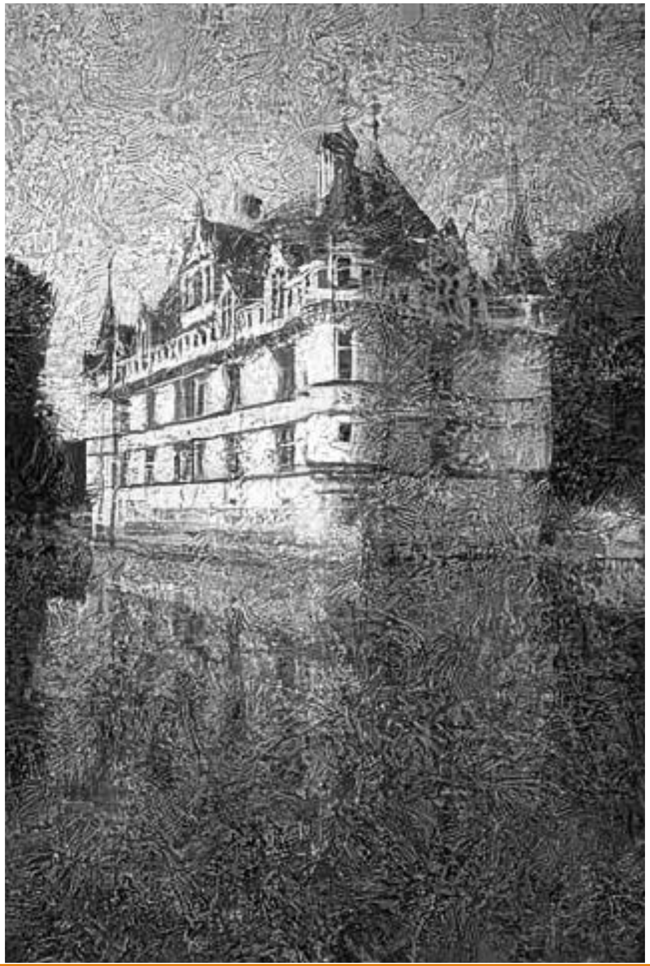


What Happen When Meet High Noise Level

Fails!



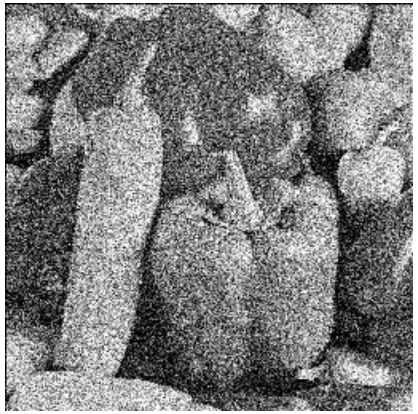
BM3D



DnCNN
(Zhang et al. TIP, 2017)

PDEs In Image Processing

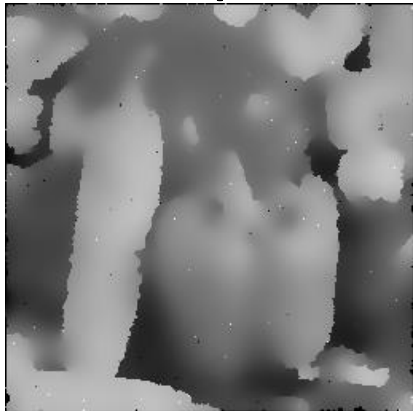
input

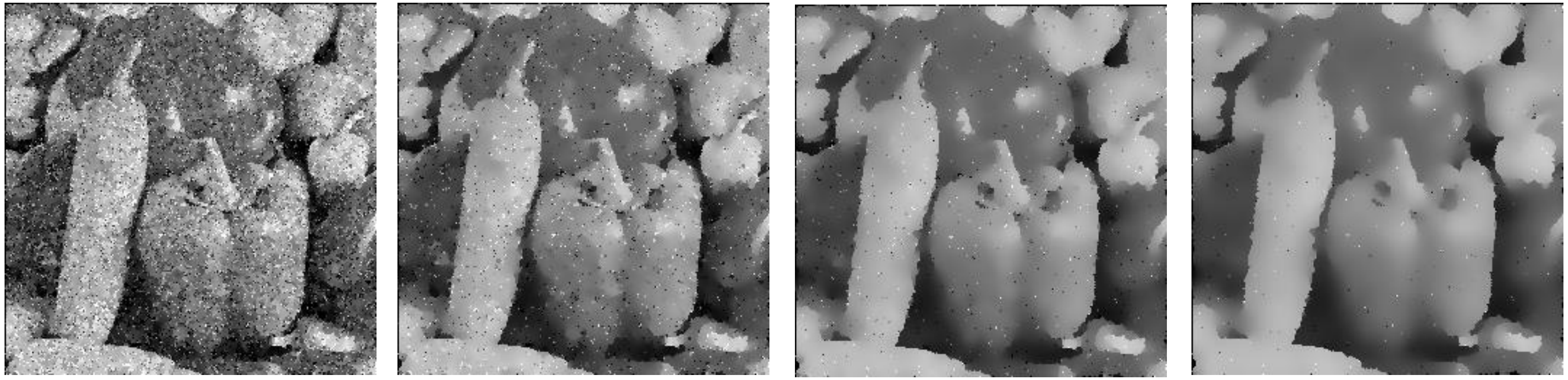


$$\begin{cases} \dfrac{\partial u}{\partial t} = \operatorname{div}\left(c(|\nabla u|^2)\ \nabla u\right) \ \text{ in } \ \Omega \times (0, \mathrm{T}), \\ \dfrac{\partial u}{\partial N} = 0 \ \text{ on } \ \partial\Omega \times (0, T), \\ u(0, x) = u_0(x) \ \text{ in } \ \Omega, \end{cases}$$

processing

output

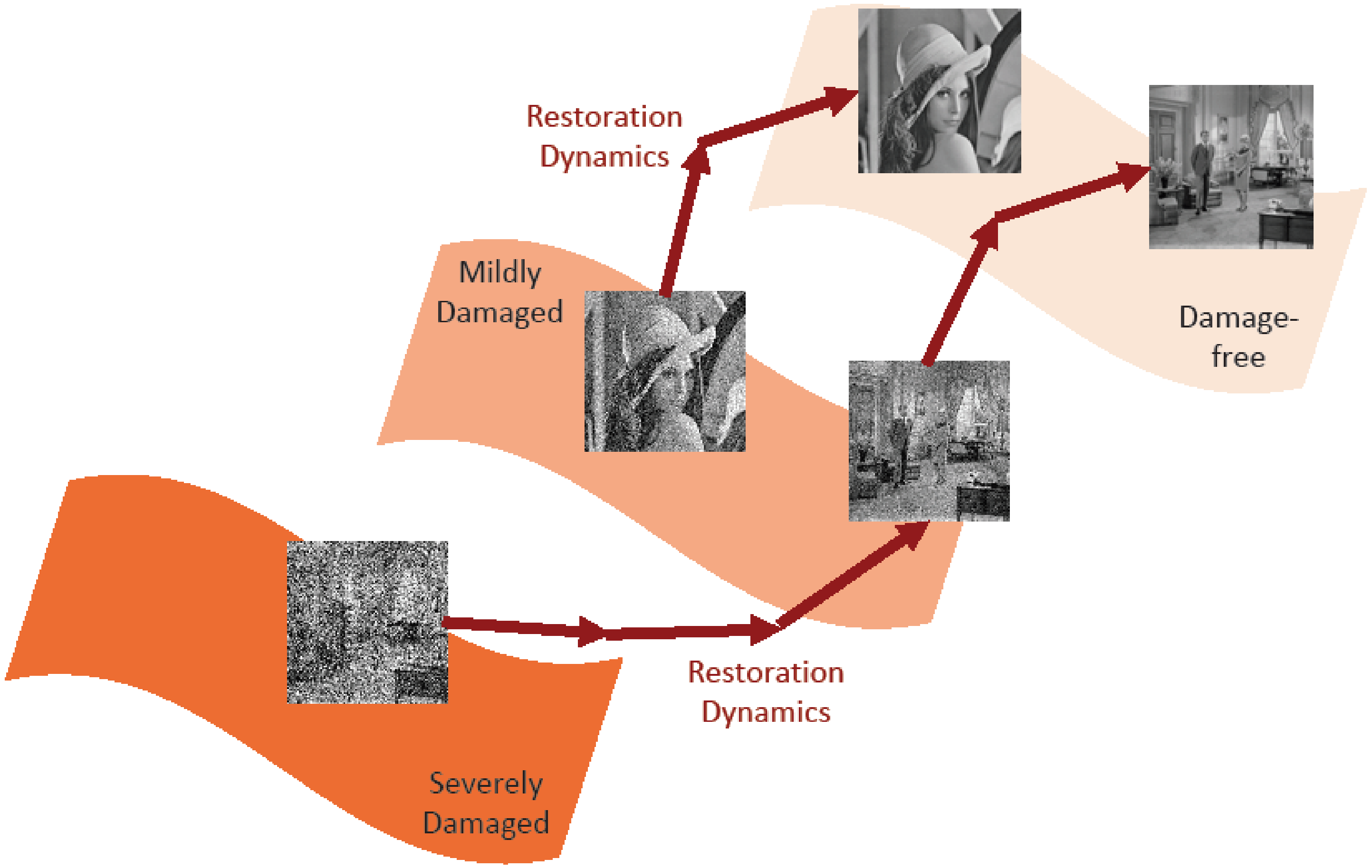




Zhang X, Lu Y et al., Dynamically Unfolding Recurrent Restorer: A Moving Endpoint Control Method for Image Restoration ICLR *2019.*

Moving Endpoint Control

Terminal time as a variable to train



Early Stopping Is A Regularization

Can we train it?

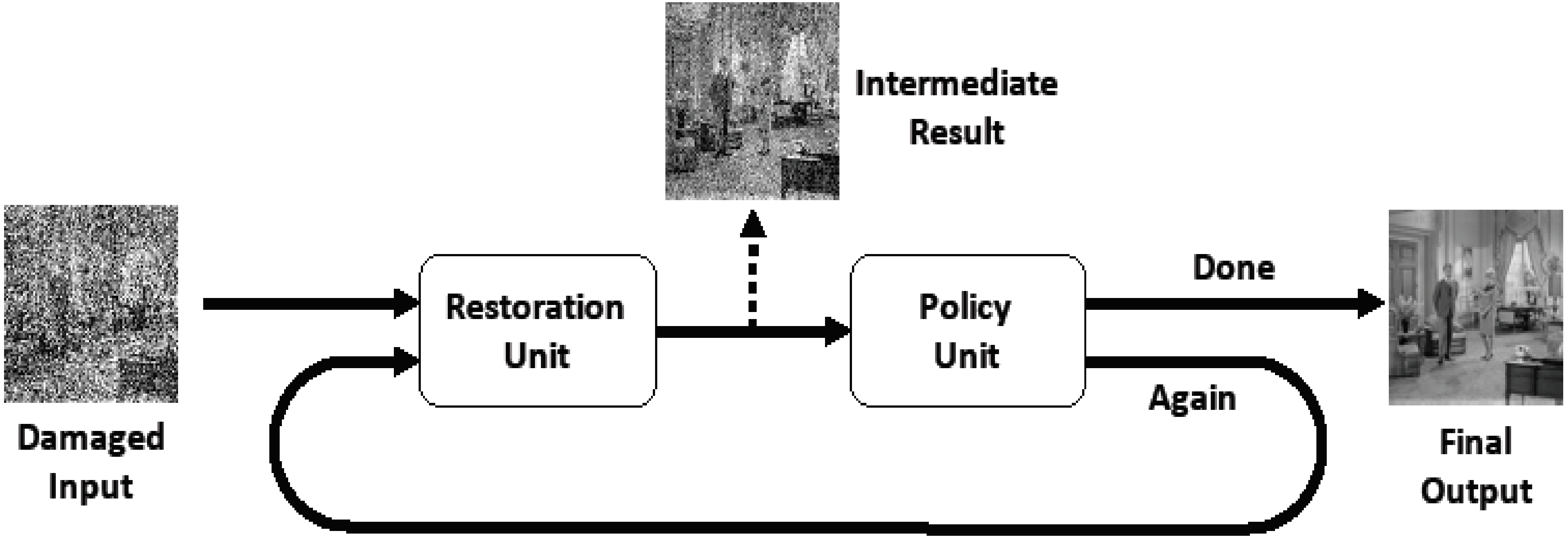
Need to be learn

$$\min_{w,\tau} L(X(\tau), y) + \int_0^\tau R(w(t), t)dt$$

$$s.t.\ \dot{X} = f(X(t), w(t)), t \in (0, \tau)$$

$$X(0) = x_0.$$

Our Approach: Dynamically Unfolding Recurrent Restorer



Zhang X, Lu Y et al., Dynamically Unfolding Recurrent Restorer: A Moving Endpoint Control Method for Image Restoration ICLR *2019.*

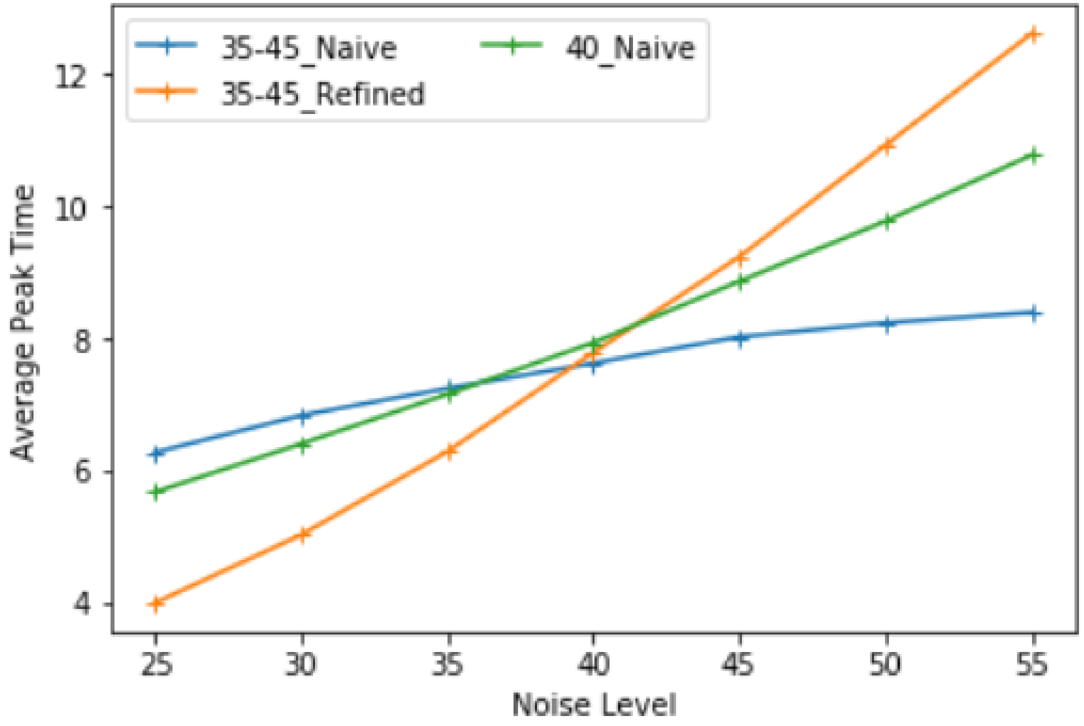
A Good Policy Leads To A Good Restorer

$$\min_{w,\tau} L(X(\tau), y) + \int_0^\tau R(w(t), t)dt$$

Policy Unit

$$s.t.\ \dot{X} = f(X(t), w(t)), t \in (0, \tau)$$
$$X(0) = x_0.$$

Restoration Unit



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.

Strategy: Training Noise	Strategy: Policy	Noise Level: 25	30	35	40	45	50	55
40	Naive	28.61	28.13	27.62	**27.19**	26.57	26.17	24.00
35, 45	Naive	27.74	27.17	26.66	26.24	26.75	25.61	24.75
35, 45	Refined	**29.14**	**28.33**	**27.67**	**27.19**	**27.69**	**26.61**	**25.88**

Zhang X, Lu Y et al., Dynamically Unfolding Recurrent Restorer: A Moving Endpoint Control Method for Image Restoration ICLR *2019.*

DURR Model
Discretize: Turn To An RL Problem

$$\min_{w,N_i(i=1,2,\cdots,d)} \sum_{i=1}^{d}\sum_{j=1}^{N_i} R_j(w_j)dt + \lambda l(X^i_{N_i}, f_i)$$

$$s.t. X^i_n = X^i_{n-1} + \Delta t f(X^i_{n-1}, w(t)), n = 1, 2, \cdots, N_i, (i = 1, 2, \cdots, d)$$

$$X^i_0 = x_i, i = 1, 2, \cdots, d$$

Consider the objective as a reward

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

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 π_θ
6: Calculate the policy gradient and then perform the optimization:

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

. The expectation here is estimated on the sampled trajectory.

You can also choose other approaches:

- A good image quality assessment without reference.
- A Classifer
- 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.

	BM3D [11]	WNMM [21]	DnCNN-B [44]	$UNLNet_5$ [28]	DURR
$\sigma = 15$	31.07	31.31	31.60	31.47	31.38*
$\sigma = 25$	28.55	28.73	29.15	28.96	29.15
$\sigma = 35$	27.07	27.28	27.66	27.50	27.70
$\sigma = 45$	25.99	26.26	26.62	26.48	26.71
$\sigma = 55$	25.26	25.49	25.80	25.64	25.91
$\sigma = 65$	24.69	24.51	23.40*	-	25.25*
$\sigma = 75$	22.63	22.71	18.73*	-	24.69*

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.

	BM3D [11]	WNMM [21]	DnCNN-B [44]	$UNLNet_5$ [28]	DURR
$\sigma = 15$	31.07	31.31	31.60	31.47	31.38*
$\sigma = 25$	28.55	28.73	29.15	28.96	29.15
$\sigma = 35$	27.07	27.28	27.66	27.50	27.70
$\sigma = 45$	25.99	26.26	26.62	26.48	26.71
$\sigma = 55$	25.26	25.49	25.80	25.64	25.91
$\sigma = 65$	24.69	24.51	23.40*	-	25.25*
$\sigma = 75$	22.63	22.71	18.73*	-	24.69*

Nose Level Doesn’t Seen In Training

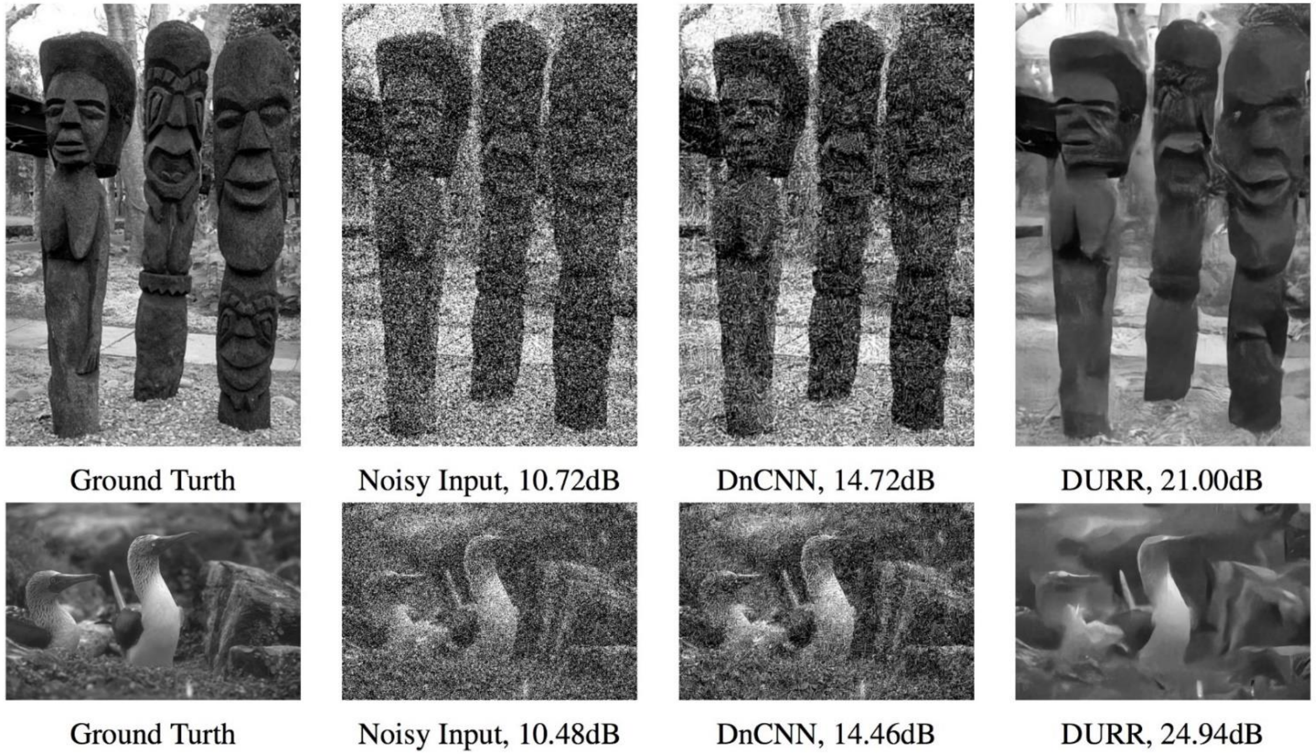
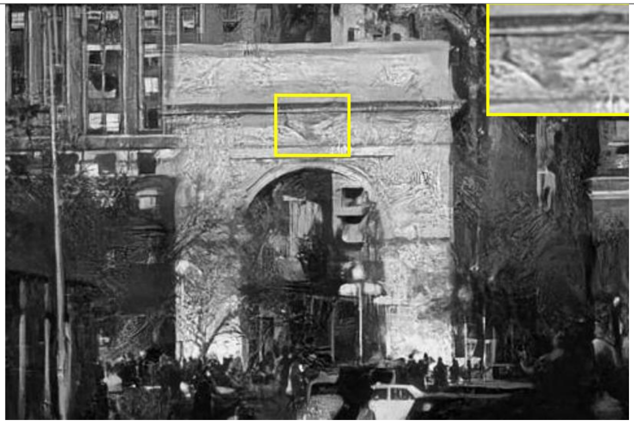
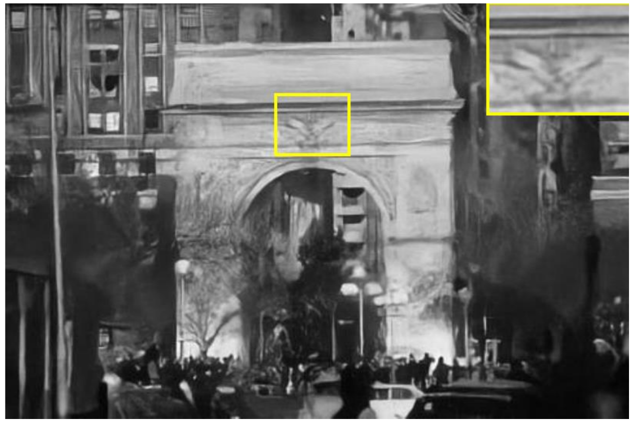


Figure 9: Denoising results of images from BSD68 with extreme noise conditions ($\sigma = 95$).



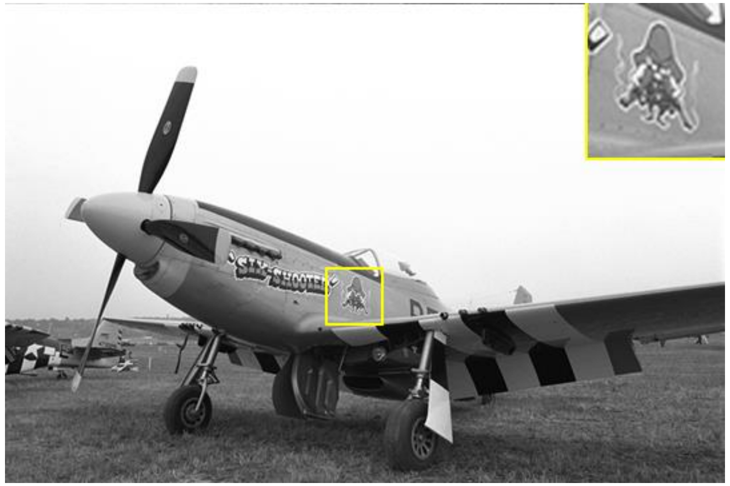
DnCNN



DURR

Zhang X, Lu Y et al., Dynamically Unfolding Recurrent Restorer: A Moving Endpoint Control Method for Image Restoration ICLR *2019.*

JPEG Deblocking



Ground Truth

DnCNN-B

Our DURR

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.

QF	JPEG	SA-DCT [18]	AR-CNN [14]	AR-CNN-B	DnCNN-3 [44]	DURR
10	27.77	28.65	28.98	28.53	29.40	29.23*
20	30.07	30.81	31.29	30.88	31.59	31.68
30	31.41	32.08	32.69	32.31	32.98	33.05
40	32.45	32.99	33.63	33.39	33.96	34.01*

One Model For All QF

Zhang X, Lu Y et al., Dynamically Unfolding Recurrent Restorer: A Moving Endpoint Control Method for Image Restoration ICLR *2019.*

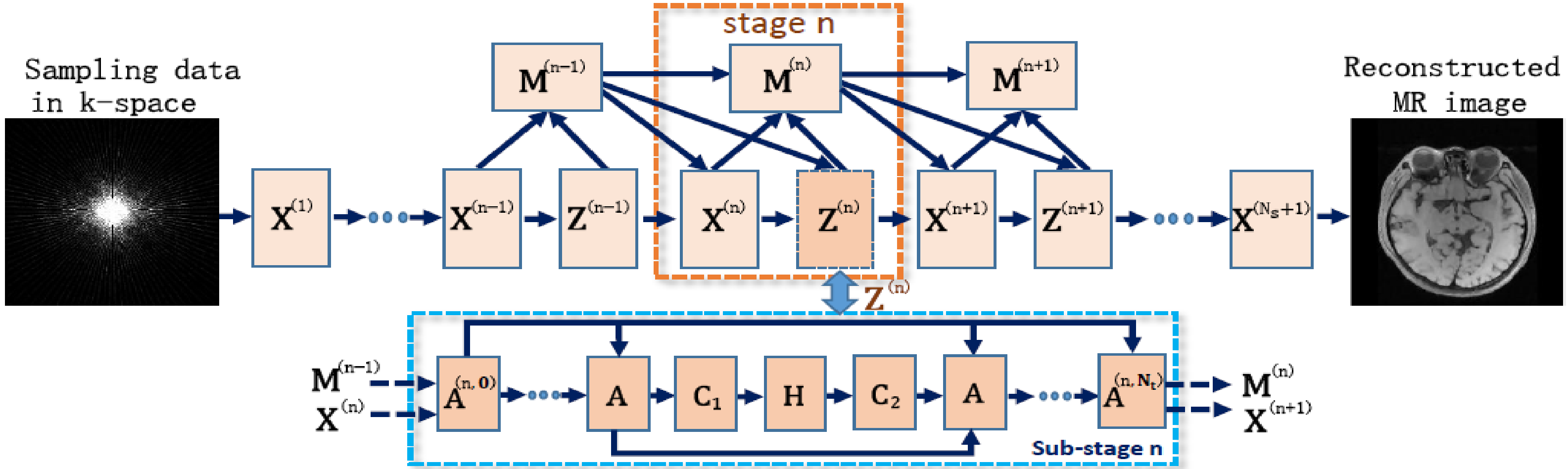
Application In Medical Imaging

UNROLLING REVISITED

Unrolled Dynamics:
ADMM-Net

$$\min_{x,z} \frac{1}{2}\|Ax - y\|_2^2 + \sum_{l=1}^{L} \lambda_l g(D_l z)$$
$$s.t. \quad z = x.$$

$$\begin{cases} \mathbf{X^{(n)}} : x^{(n)} = F^T(P^T P + \rho I)^{-1} \\ \qquad [P^T y + \rho F(z^{(n-1)} - \beta^{(n-1)})], \\ \mathbf{Z^{(n)}} : z^{(n,k)} = \mu_1 z^{(n,k-1)} + \mu_2(x^{(n)} + \beta^{(n-1)}) \\ \qquad - \sum_{l=1}^{L} \tilde{\lambda}_l D_l^T \mathcal{H}(D_l z^{(n,k-1)}), \\ \mathbf{M^{(n)}} : \beta^{(n)} = \beta^{(n-1)} + \tilde{\eta}(x^{(n)} - z^{(n)}), \end{cases}$$



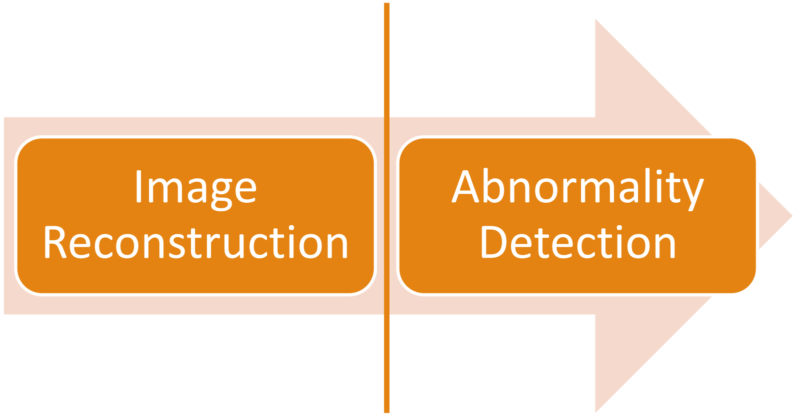
Sun, Li, and Xu. Deep ADMM-net for compressive sensing MRI. NIPS 2016.

Unrolled Dynamics: ADMM-Net

Method	10%		20%		30%		40%		50%		Test Time
	NMSE	PSNR	NMSE	PSNR	NMSE	PSNR	NMSE	PSNR	NMSE	PSNR	CPU \ GPU
Zero-filling [46]	0.2624	26.35	0.1700	29.96	0.1247	32.59	0.0968	34.76	0.0770	36.73	0.001s\--
TV [3]	0.1539	30.90	0.0929	35.20	0.0673	37.99	0.0534	40.00	0.0440	41.69	0.739s\--
RecPF [11]	0.1498	30.99	0.0917	35.32	0.0668	38.06	0.0533	40.03	0.0440	41.71	0.311s\--
SIDWT [3]	0.1564	30.81	0.0885	35.66	0.0620	38.72	0.0484	40.88	0.0393	42.67	7.864s\--
PBDW [24]	0.1290	32.45	0.0814	36.34	0.0627	38.64	0.0518	40.31	0.0437	41.81	35.364s\--
PANO [7]	0.1368	31.98	0.0800	36.52	0.0592	39.13	0.0477	41.01	0.0390	42.76	53.478s\--
FDLCP [29]	0.1257	32.63	0.0759	36.95	0.0592	39.13	0.0500	40.62	0.0428	42.00	52.222s\--
BM3D-MRI [30]	0.1132	33.53	0.0674	37.98	0.0515	40.33	0.0426	41.99	0.0359	43.47	40.911s\--
Init-Net$_{10}$	0.2589	26.17	0.1737	29.64	0.1299	32.16	0.1025	34.21	0.0833	36.01	3.827s\0.644s
ADMM-Net$_{10}$	**0.1082**	**33.88**	**0.0620**	**38.72**	**0.0480**	**40.95**	**0.0395**	**42.66**	**0.0328**	**44.29**	3.827s\0.644s

Further Application of Unrolling – Task-Based Image Reconstruction

Two-step approach: imaging and diagnosis

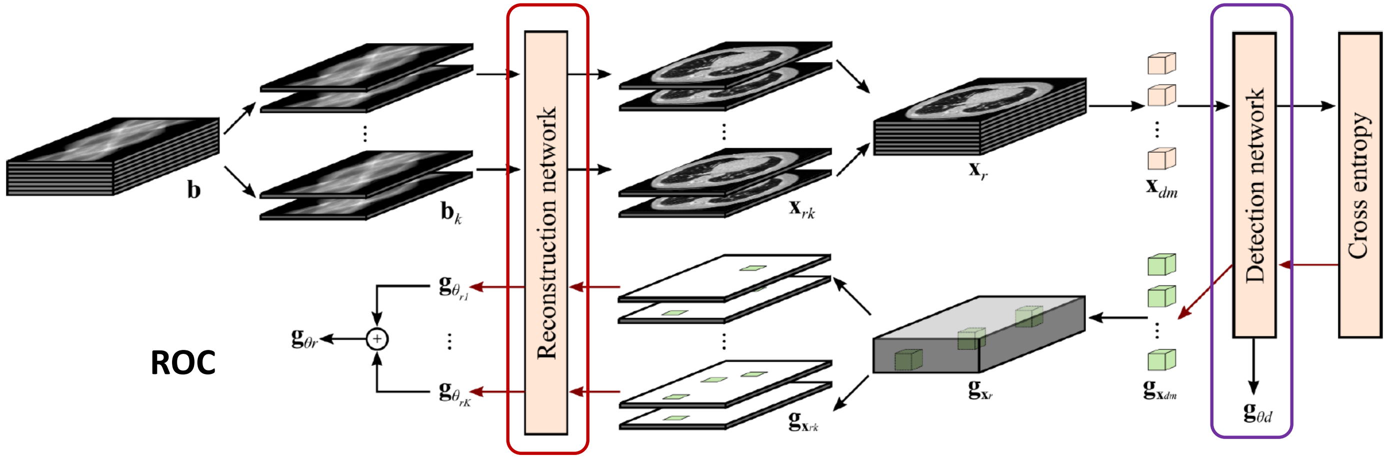


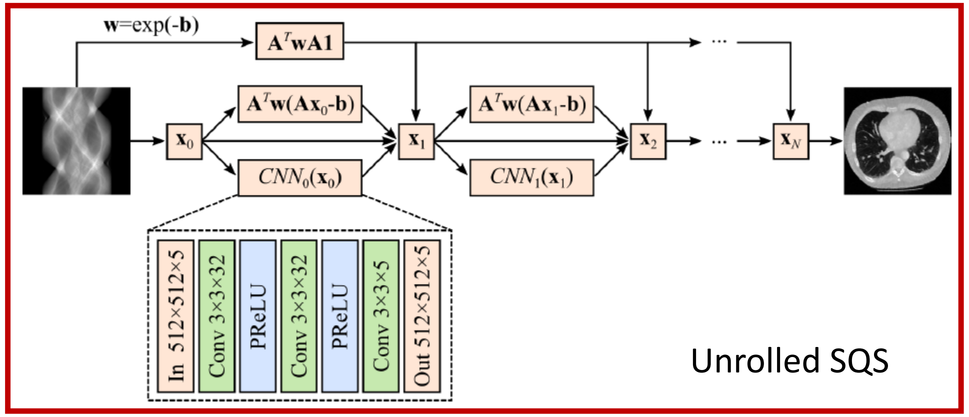
Problems of the two-step approach:

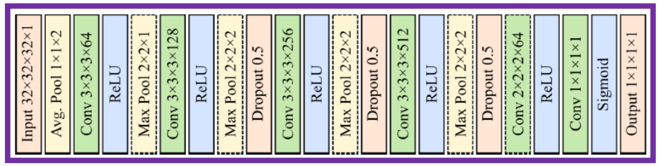
- Evaluation of the reconstructed image quality.
- Redundancy in data for a specific task.

Can we make it end-to-end, and does it help?

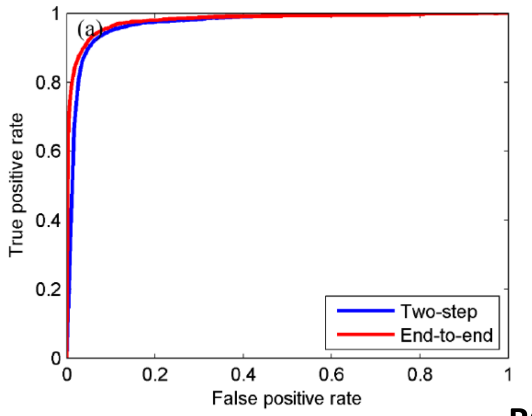
Further Application of Unrolling – Task-Based Image Reconstruction



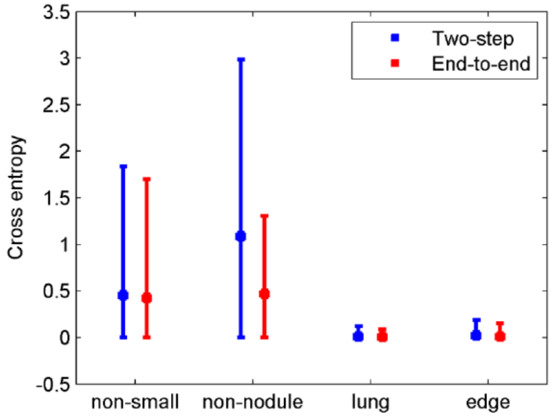




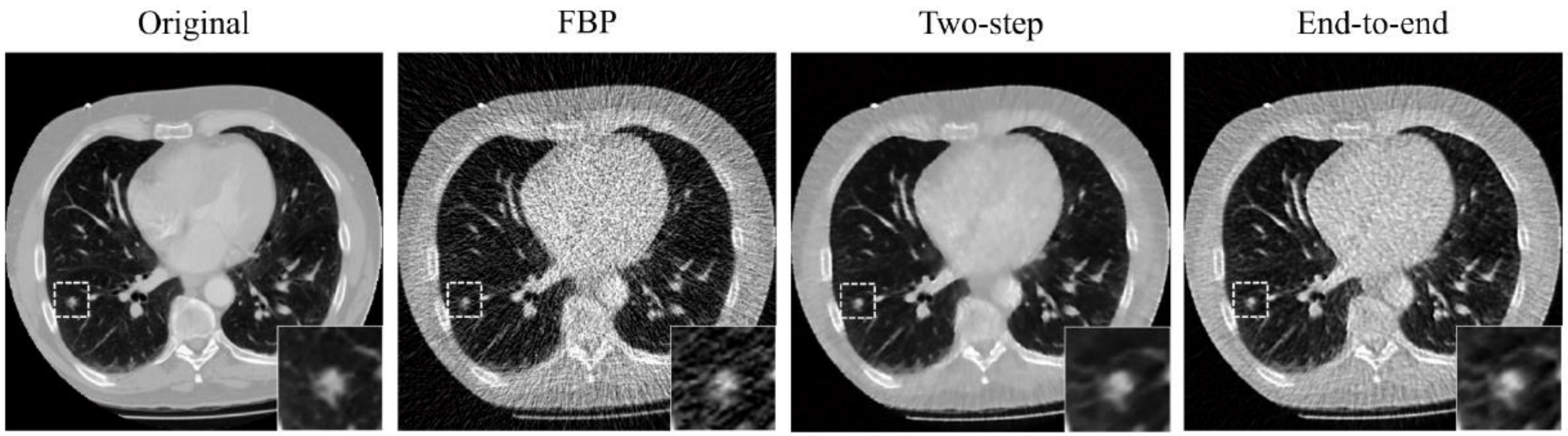
Further Application of Unrolling – Task-Based Image Reconstruction



ROC



Cross Entropy



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

Medical Physics
The International Journal of Medical Physics Research and Practice

Explore this journal >

Point/Counterpoint

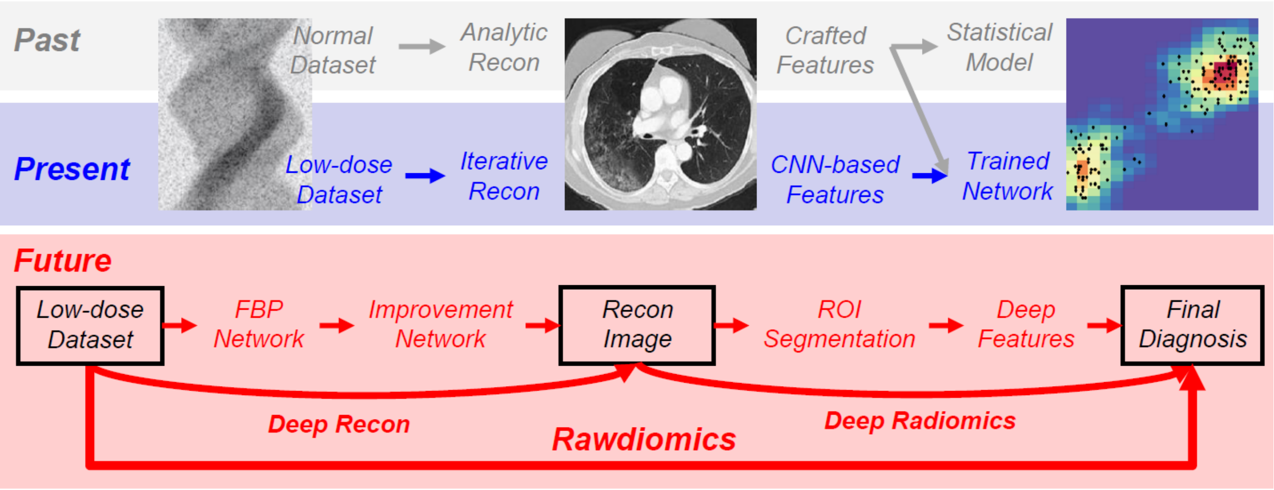
Radiomics in lung cancer: Its time is here

Mannudeep Kalra M.D., Ge Wang Ph.D., Colin G. Orton Ph.D.

First published: 12 December 2017 Full publication history

DOI: 10.1002/mp.12685 View/save citation

Cited by (CrossRef): 0 articles Check for updates Citation tools ▾



arXiv.org > cs > arXiv:1706.04284

Computer Science > Computer Vision and Pattern Recognition

When Image Denoising Meets High-Level Vision Tasks: A Deep Learning Approach

Ding Liu, Bihan Wen, Xianming Liu, Zhangyang Wang, Thomas S. Huang

(Submitted on 14 Jun 2017 (v1), last revised 16 Apr 2018 (this version, v3))

arXiv.org > cs > arXiv:1809.01826

Computer Science > Computer Vision and Pattern Recognition

Connecting Image Denoising and High-Level Vision Tasks via Deep Learning

Ding Liu, Bihan Wen, Jianbo Jiao, Xianming Liu, Zhangyang Wang, Thomas S. Huang

(Submitted on 6 Sep 2018)

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

Loss Function

Regularization

$H = p \cdot f - L$

Original ODE

$$\min_{\theta \in \mathcal{U}} \sum_{i=1}^{K} \Phi_i(X_T^i) + \int_0^T L(\theta_t) dt,$$

$$\dot{X}_t^i = f(t, X_t^i, \theta_t), \quad X_0^i = x^i, \quad 0 \le t \le T, \quad i = 1, \dots, K, \tag{1}$$

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 ess $\sup_{t \in [0,T]} \|\theta_t^*\|_\infty < \infty$. Then, there exists an absolutely continuous co-state process $P^* : [0, T] \to \mathbb{R}^d$ such that the Hamilton's equations*

$$\dot{X}_t^* = \nabla_p H(t, X_t^*, P_t^*, \theta_t^*), \qquad X_0^* = x, \tag{2}$$

$$\dot{P}_t^* = -\nabla_x H(t, X_t^*, P_t^*, \theta_t^*), \qquad P_T^* = -\nabla \Phi(X_T^*), \tag{3}$$

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

$$H(t, X_t^*, P_t^*, \theta_t^*) \ge H(t, X_t^*, P_t^*, \theta) \text{ for all } \theta \in \Theta \tag{4}$$

Optimization: Solving The "KKT" Condition

@Maximum Principle Based Algorithms

Loss Function

Regularization

$H = p \cdot f - L$

$$\min_{\theta \in \mathcal{U}} \sum_{i=1}^{K} \Phi_i(X_T^i) + \int_0^T L(\theta_t)dt,$$

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

Costate

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 ess $\sup_{t\in[0,T]} \|\theta_t^*\|_\infty < \infty$. Then, there exists an absolutely continuous co-state process $P^* : [0,T] \to \mathbb{R}^d$ such that the Hamilton's equations*

$$\dot{X}_t^* = \nabla_p H(t, X_t^*, P_t^*, \theta_t^*), \qquad X_0^* = x, \qquad (2)$$

$$\dot{P}_t^* = -\nabla_x H(t, X_t^*, P_t^*, \theta_t^*), \qquad P_T^* = -\nabla\Phi(X_T^*), \qquad (3)$$

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

$$H(t, X_t^*, P_t^*, \theta_t^*) \ge H(t, X_t^*, P_t^*, \theta) \text{ for all } \theta \in \Theta \qquad (4)$$

Optimization: Solving The "KKT" Condition

@Maximum Principle Based Algorithms

Loss Function

Regularization

$H = p \cdot f - L$

$$\min_{\theta \in \mathcal{U}} \sum_{i=1}^{K} \Phi_i(X_T^i) + \int_0^T L(\theta_t)dt,$$

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

Maximum Principle

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 ess $\sup_{t\in[0,T]} \|\theta_t^*\|_\infty < \infty$. Then, there exists an absolutely continuous co-state process $P^* : [0,T] \to \mathbb{R}^d$ such that the Hamilton's equations*

$$\dot{X}_t^* = \nabla_p H(t, X_t^*, P_t^*, \theta_t^*), \qquad X_0^* = x, \tag{2}$$

$$\dot{P}_t^* = -\nabla_x H(t, X_t^*, P_t^*, \theta_t^*), \qquad P_T^* = -\nabla\Phi(X_T^*), \tag{3}$$

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

$$H(t, X_t^*, P_t^*, \theta_t^*) \ge H(t, X_t^*, P_t^*, \theta) \text{ for all } \theta \in \Theta \tag{4}$$

Optimization: Solving The “KKT” Condition

@Maximum Principle Based Algorithms

$$\dot{X}_t^* = \nabla_p \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \qquad X_0^* = x, \tag{8}$$

$$\dot{P}_t^* = -\nabla_x \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \qquad P_T^* = -\nabla_x \Phi(X_T^*), \tag{9}$$

$$\tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*) \geq \tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*), \qquad \theta \in \Theta, t \in [0, T]. \tag{10}$$

Solving it via **Gauss-Seidel** Iteration

Algorithm 2 Extended MSA (E-MSA)

1: Initialize: $\theta^0 \in \mathcal{U}$. Hyper-parameter: ρ
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 $t \in [0, T]$

Optimization: Solving The "KKT" Condition

@Maximum Principle Based Algorithms

$$\dot{X}_t^* = \nabla_p \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \qquad X_0^* = x, \tag{8}$$

$$\dot{P}_t^* = -\nabla_x \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \qquad P_T^* = -\nabla_x \Phi(X_T^*), \tag{9}$$

$$\tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*) \geq \tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*), \qquad \theta \in \Theta, t \in [0, T]. \tag{10}$$

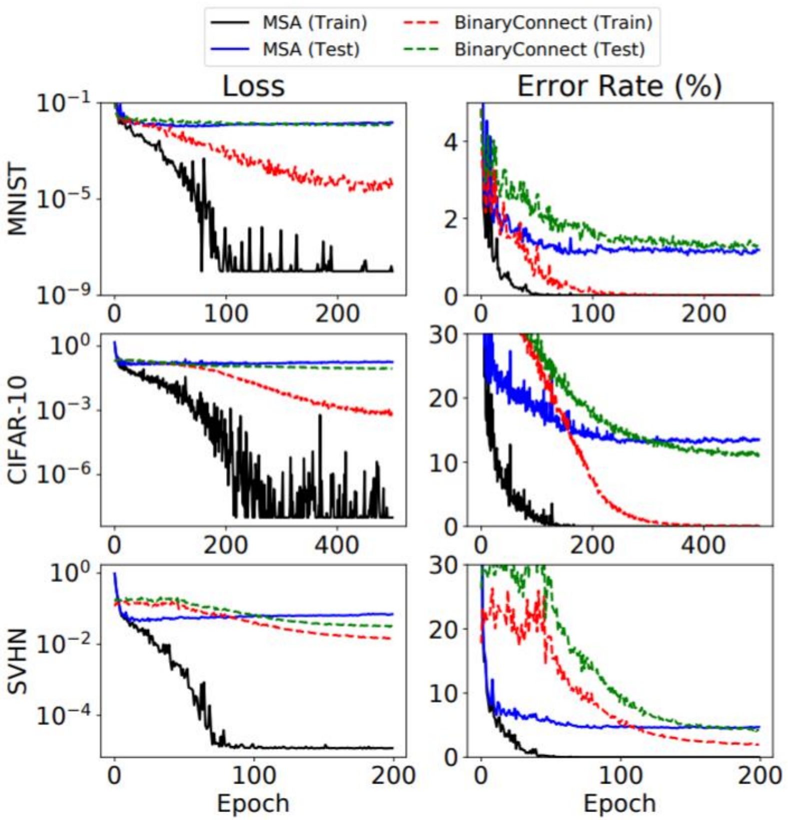
Solving it via **Gauss-Seidel** Iteration

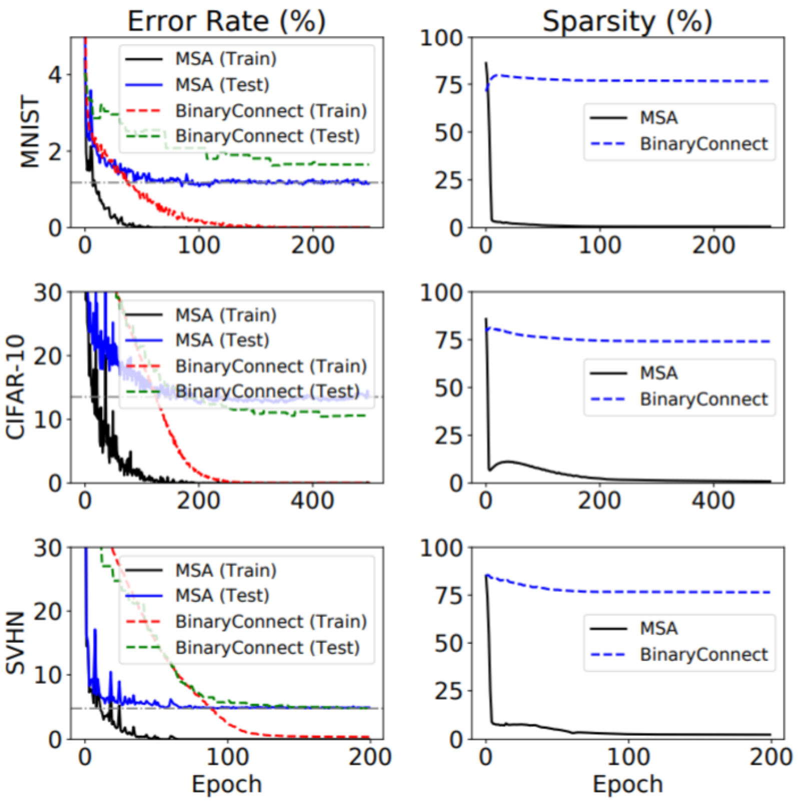
Algorithm 2 Extended MSA (E-MSA)

1: Initialize: $\theta^0 \in \mathcal{U}$. Hyper-parameter: ρ
2: **for** $k = 0$ to #Iterations **do**
3: Solve $\dot{X}_t^{\theta^k} = f(t, X_t^{\theta^k}, \theta_t^k), \quad X_0^{\theta^k} = x$
4: Solve $\dot{P}_t^{\theta^k} = -\nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k), \quad P_T^{\theta^k} = -\nabla\Phi(X_T^{\theta^k})$
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 $t \in [0, T]$

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

NODE

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

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

$\frac{\partial L}{\partial t_1} = \frac{\partial L}{\partial \mathbf{z}(t_1)}^{\mathsf{T}} f(\mathbf{z}(t_1), t_1, \theta)$ ▷ Compute gradient w.r.t. t_1

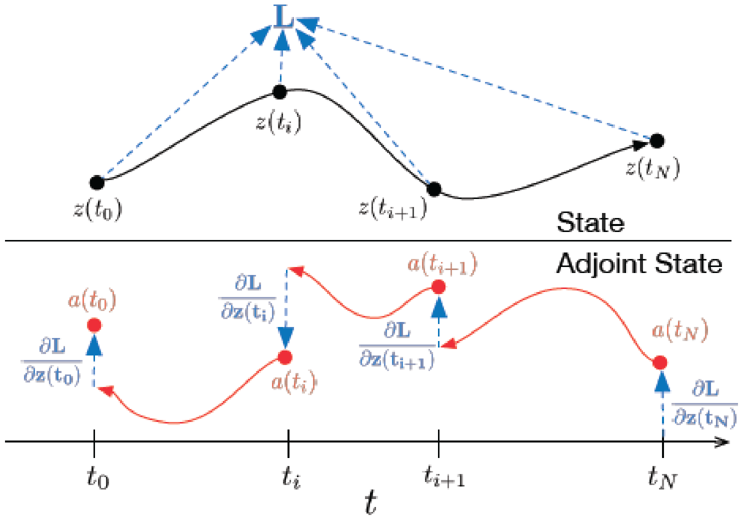
$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}, -\frac{\partial L}{\partial t_1}]$ ▷ Define initial augmented state

def aug_dynamics($[\mathbf{z}(t), \mathbf{a}(t), -, -], t, \theta$): ▷ Define dynamics on augmented state

return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^{\mathsf{T}} \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^{\mathsf{T}} \frac{\partial f}{\partial \theta}, -\mathbf{a}(t)^{\mathsf{T}} \frac{\partial f}{\partial t}]$ ▷ Concatenate time-derivatives

$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}, \frac{\partial L}{\partial t_1}$ ▷ Return all gradients



Recall the PMP

$$\dot{X}_t^* = \nabla_p H(t, X_t^*, P_t^*, \theta_t^*), \qquad X_0^* = x,$$
$$\dot{P}_t^* = -\nabla_x H(t, X_t^*, P_t^*, \theta_t^*), \qquad P_T^* = -\nabla \Phi(X_T^*).$$

VAE and Normalizing Flow

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

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$$\geq -\mathbb{D}_{\mathrm{KL}}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] + \mathbb{E}_q[\log p_\theta(\mathbf{x}|\mathbf{z})] = -\mathcal{F}(\mathbf{x}),$$

M. I. Jordan, et al., An introduction to variational methods for graphical models. Machine Learning, 37(2):183–233, 1999.

VAE and Normalizing Flow

Variational Autoencoders

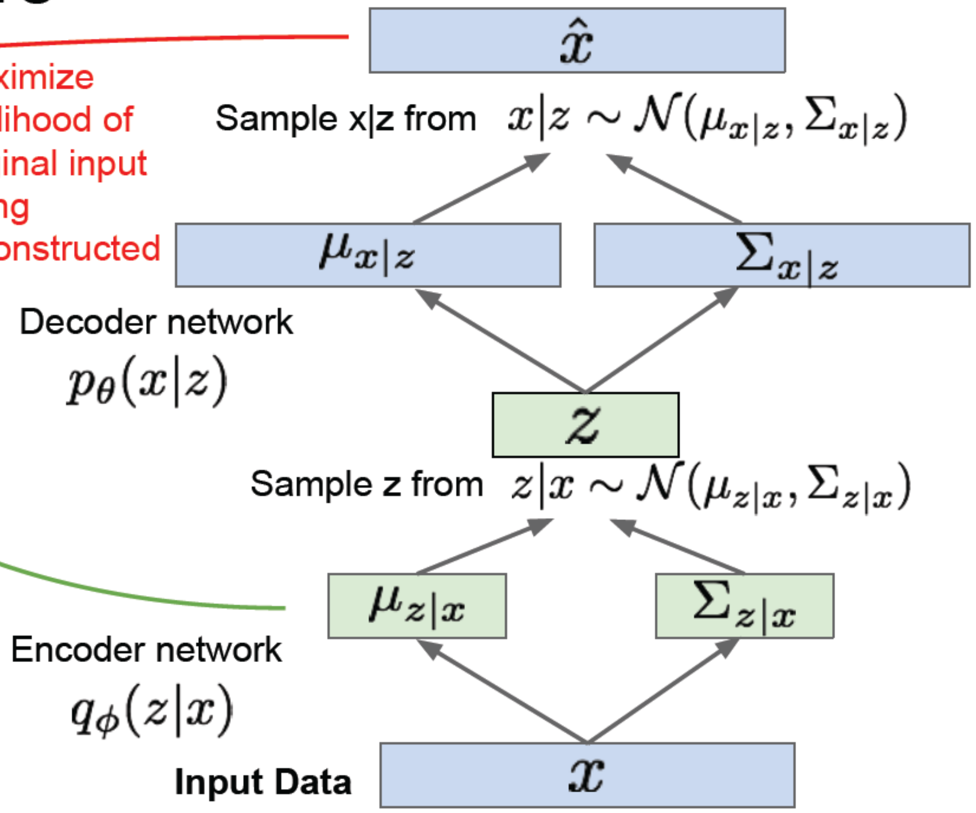
Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

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!



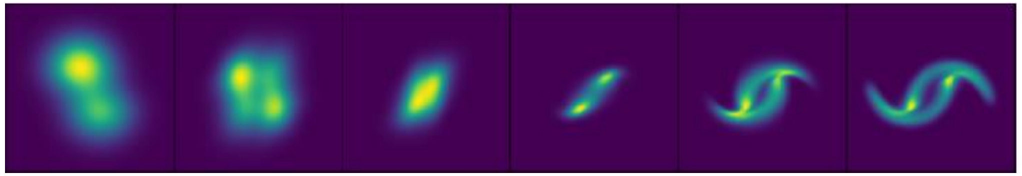
VAE and Normalizing Flow

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

$$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$

where f_j are smooth invertible maps



Algorithm 1 Variational Inf. with Normalizing Flows

Parameters: ϕ variational, $\boldsymbol{\theta}$ generative
while not converged **do**
  $\mathbf{x} \leftarrow \{\text{Get mini-batch}\}$
  $\mathbf{z}_0 \sim q_0(\bullet|\mathbf{x})$
  $\mathbf{z}_K \leftarrow f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z}_0)$
  $\mathcal{F}(\mathbf{x}) \approx \mathcal{F}(\mathbf{x}, \mathbf{z}_K)$
  $\Delta\boldsymbol{\theta} \propto -\nabla_{\theta}\mathcal{F}(\mathbf{x})$
  $\Delta\phi \propto -\nabla_{\phi}\mathcal{F}(\mathbf{x})$
end while

NODE for Normalizing Flow

Use the change of variables theorem to compute exact changes in probability if samples are transformed through a bijective function f:

$$\mathbf{z}_1 = f(\mathbf{z}_0) \implies \log p(\mathbf{z}_1) = \log p(\mathbf{z}_0) - \log \left| \det \frac{\partial f}{\partial \mathbf{z}_0} \right|$$

Use NODE:

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)} \right)$$

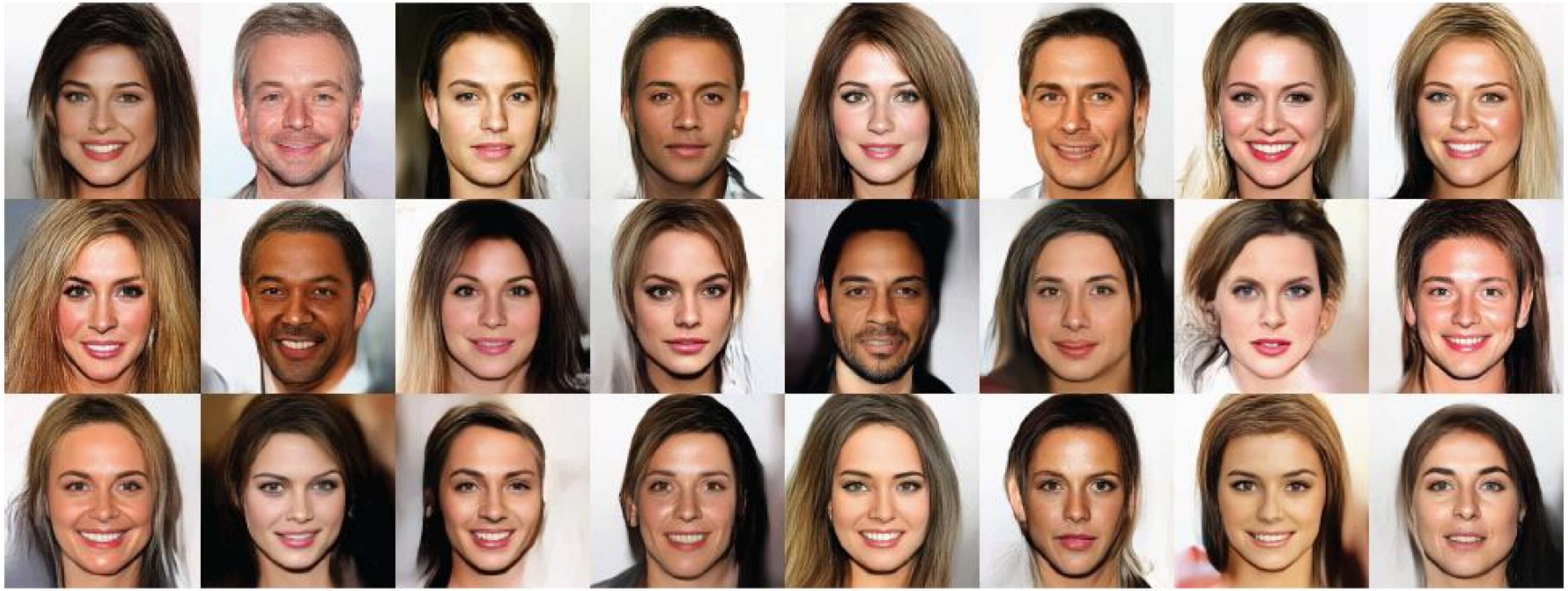
Reducing the calculation cost of gradient from $O(d^3)$ to $O(d)$

VAE and Normalizing Flow

Normalizing flow for image synthesis:

VAE and Normalizing Flow

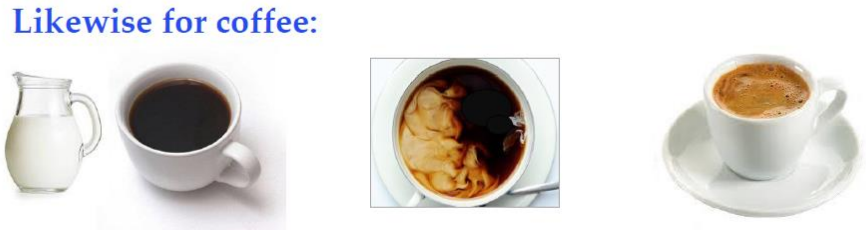
Normalizing flow for image synthesis:



Applied Math Perspective on Deep Learning

Take home message:

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



From David Wipf's Slide@ICASSP2018

Thanks and Questions?