Deep Residual Learning: A PDE Perspective

Yiping Lu

Peiking University, School of Mathematic Science

1500010638@pku.edu.cn

June 29, 2017

Overview

1. Deep Residual Learning
   - Ensemble learning
   - Unrolled Iterative Estimation
   - ResNet,RNN and Visual Cortex
2. PDE And Residual Learning
3. Understanding Deep Learning
   - Generalization Gap

HighWay Net

There is plenty of theoretical and empirical evidence that depth of neural networks is a crucial ingredient for their success. The purpose of **HighWay Net** is to **go depper**.
Instead of the every layer processing in the traditional networks which can be written as

$$y = H(x, W_H)$$

Every block of Highway Net can be written as:

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C)$$

HighWay Net

Sricastava ect.(2015) suggested that the shortcut in the Highway Net may written as

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$$

Idea: parametrized skip-connections that are referred to information high ways.

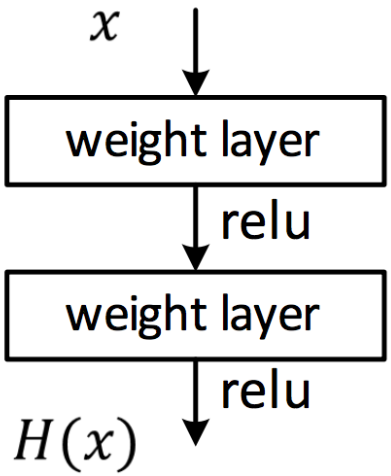
The transform gate is defined as $T = \sigma(W_T^T x + b_T)$, suggest that b_T can initialization with a negative value such taht the network is initially biased towars carry behavior. Similar with **LSTM**.

Residual Learning(CVPR2016)

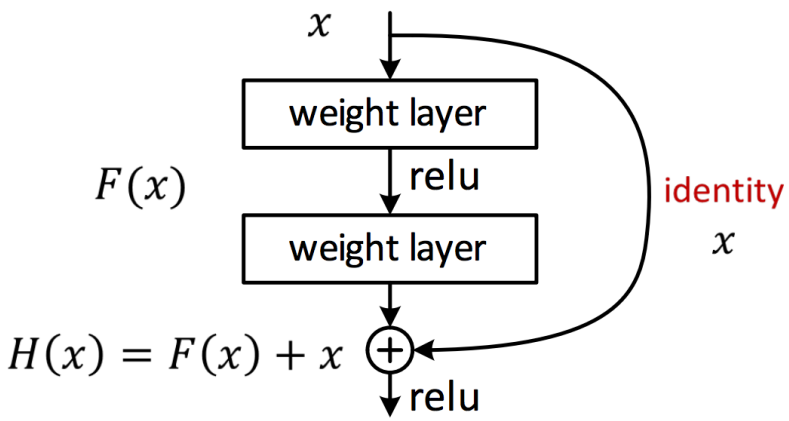
Champion of ImageNet,COCO challenge 2015

- Plaint net

any two stacked layers



- Residual net



Every block of ResNet runs as

$$H(x)=F(x)+x$$

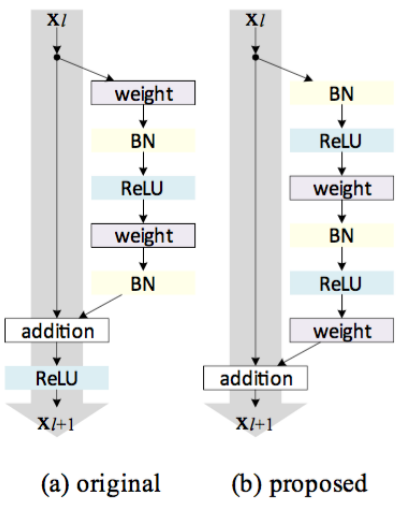
hope the 2 weight layer fit the residual $F(x)$

Identity Mappings in Deep Residual Networks

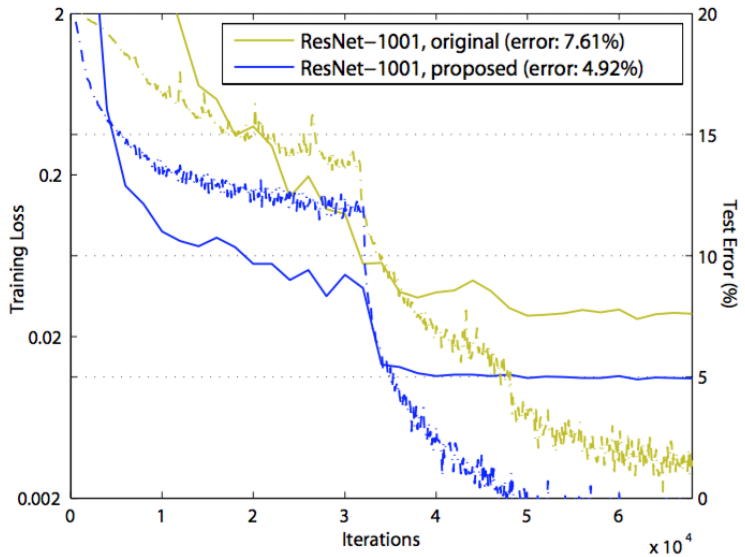
Creating a 'direct' path for propagating information - not only within a residual unit, but through the entire network. That is to say in the following formula

$$y_l = h(x_l) + F(x_l + W_l)$$
$$x_{l+1} = f(y_l)$$

f and h are both identity mappings.



(a) original (b) proposed



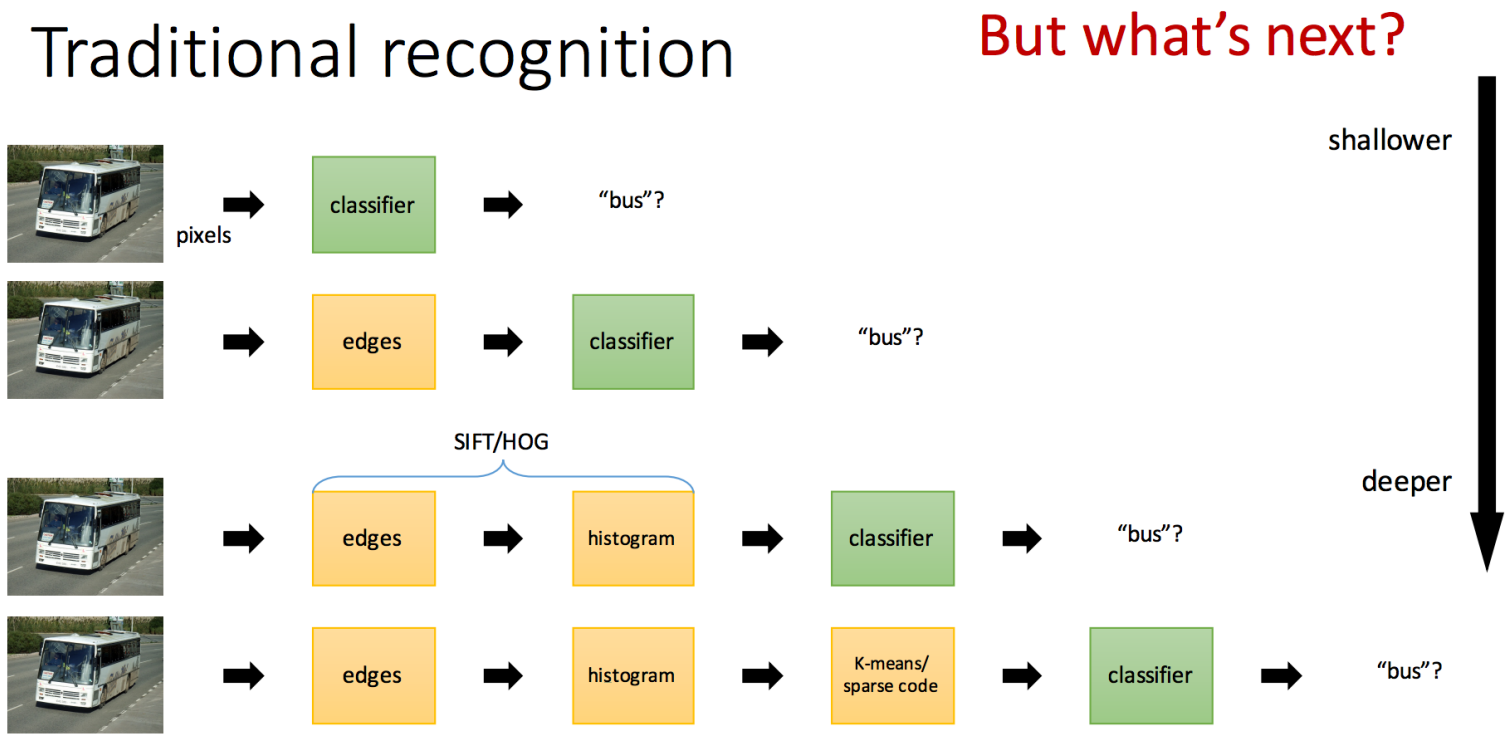
ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

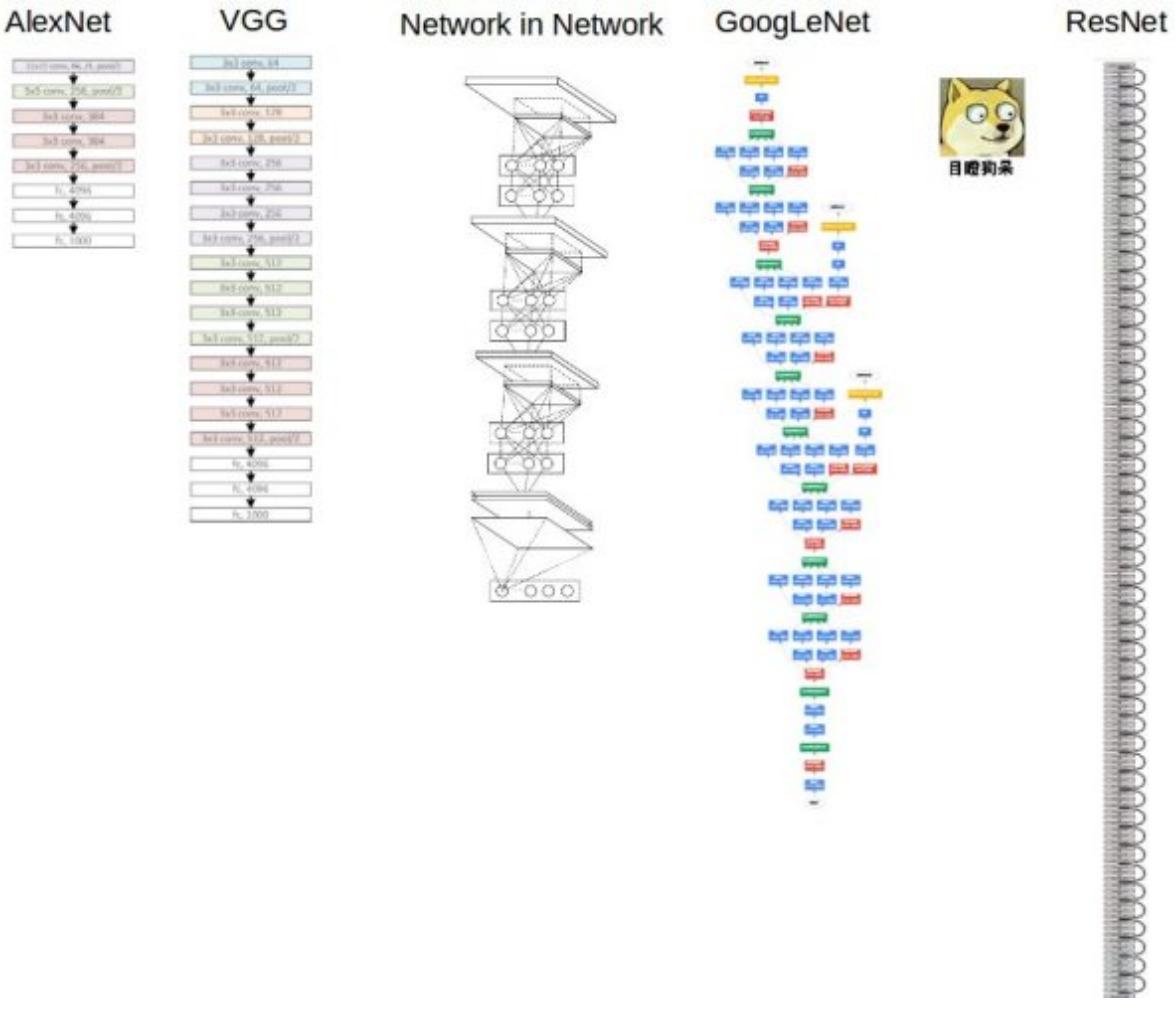
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Evolution in depth

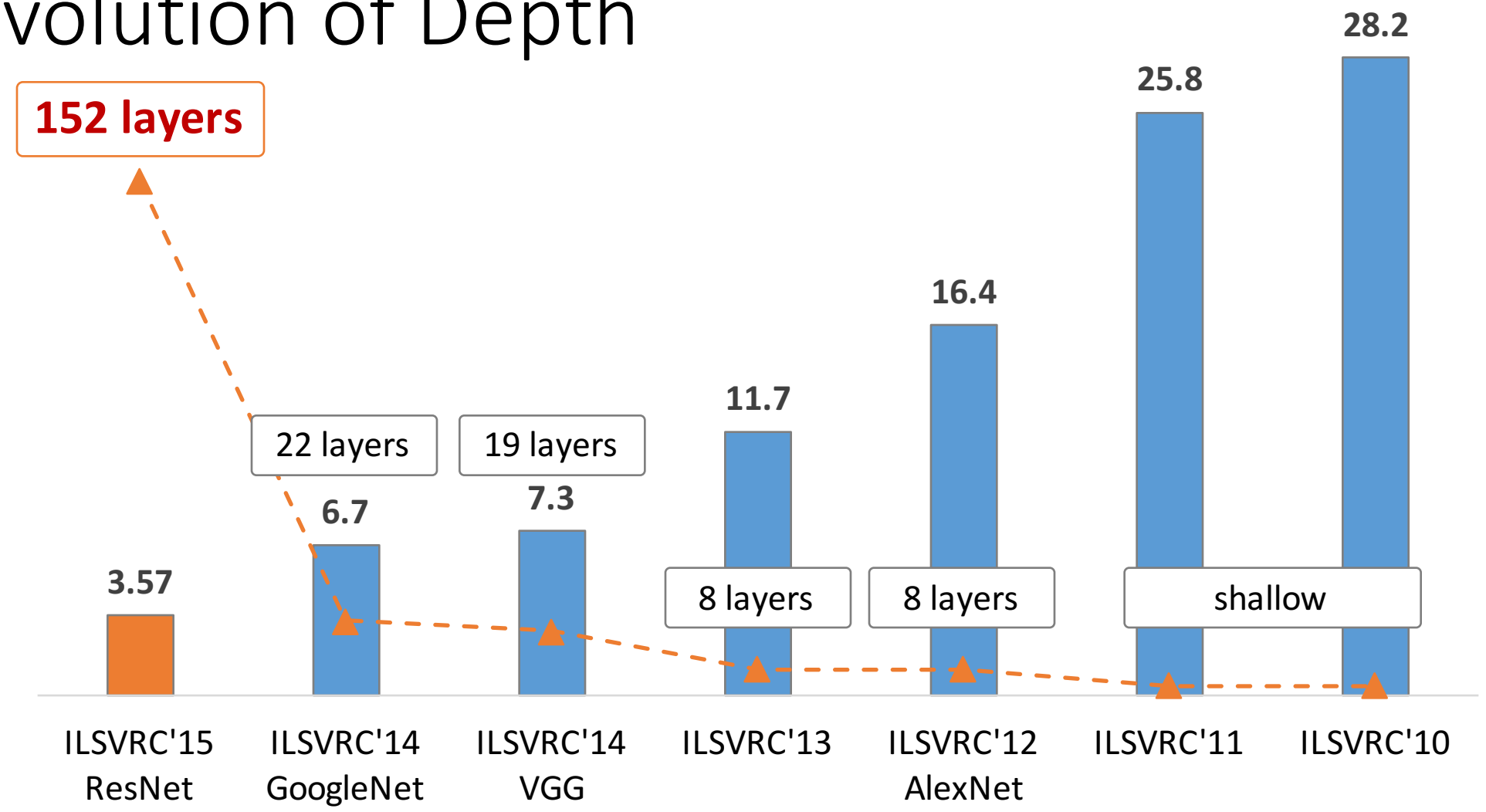


Evolution in depth

ResNet can become very very deep.



Revolution of Depth



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Highway and Residual Network

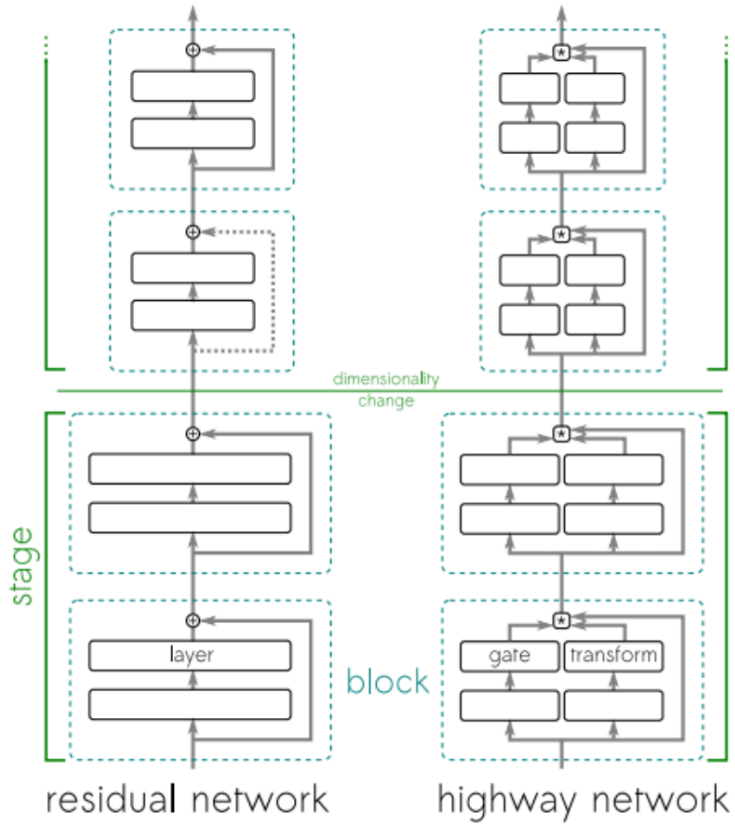
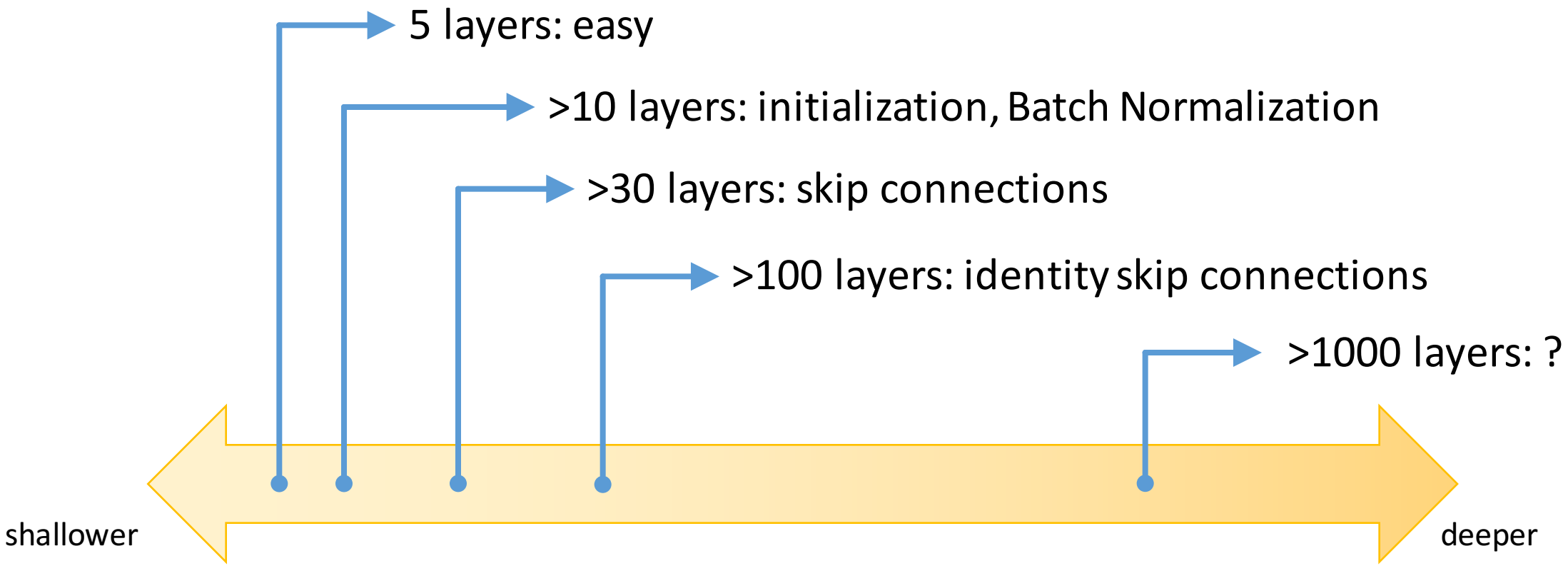


Figure 1: Illustrating our usage of *blocks* and *stages* in Highway and Residual networks. Note also that blocks and layers refer to the same thing in a classic neural network.

Spectrum of Depth



Training Relu Network

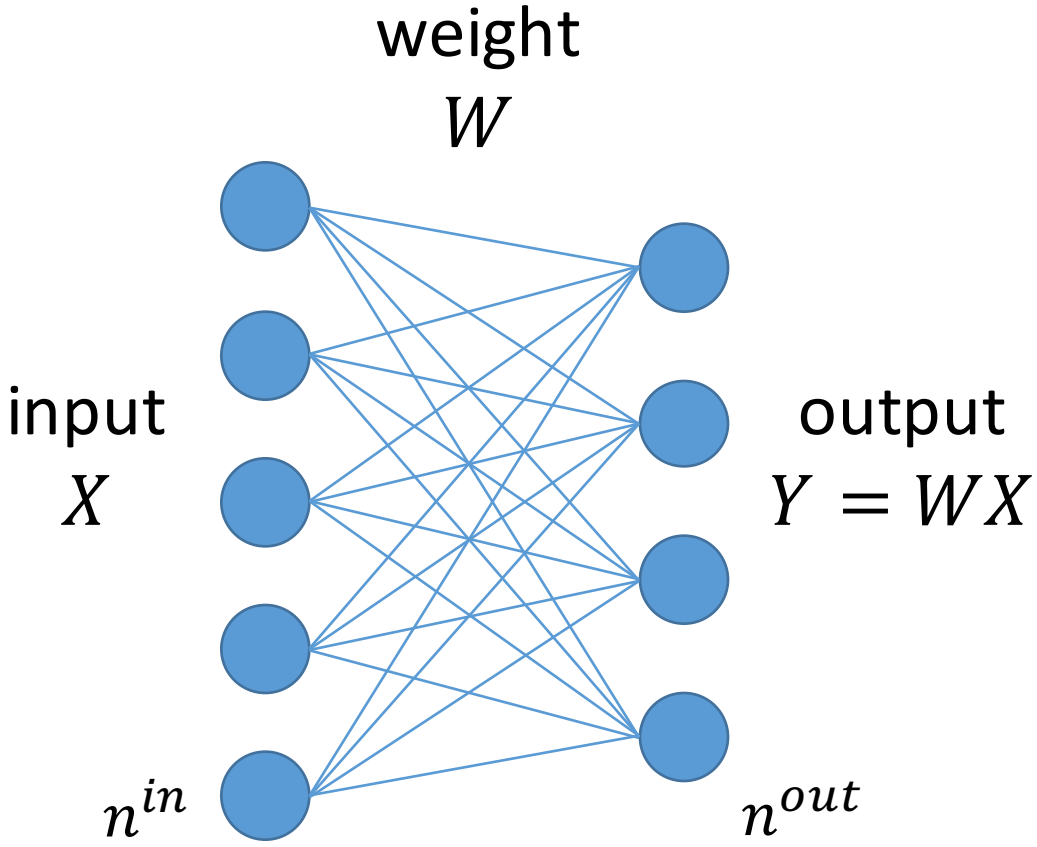
Rectifier networks are easier to train compared with traditional sigmoid-like activation networks. But a bad initialization can still hamper the learning of a highly non-linear system.

$$y_l = f(W_l x_l + b_l)$$

we need

$$\Pi_d \frac{1}{2} n_d^{in} Var[w_d] = 1$$

Initialization



If:

- Linear activation
- x, y, w: independent

Then:

1-layer:

$$Var[y] = (n^{in} Var[w]) Var[x]$$

Multi-layer:

$$Var[y] = (\prod_d n_d^{in} Var[w_d]) Var[x]$$

LeCun et al 1998 “Efficient Backprop”

Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

Initialization

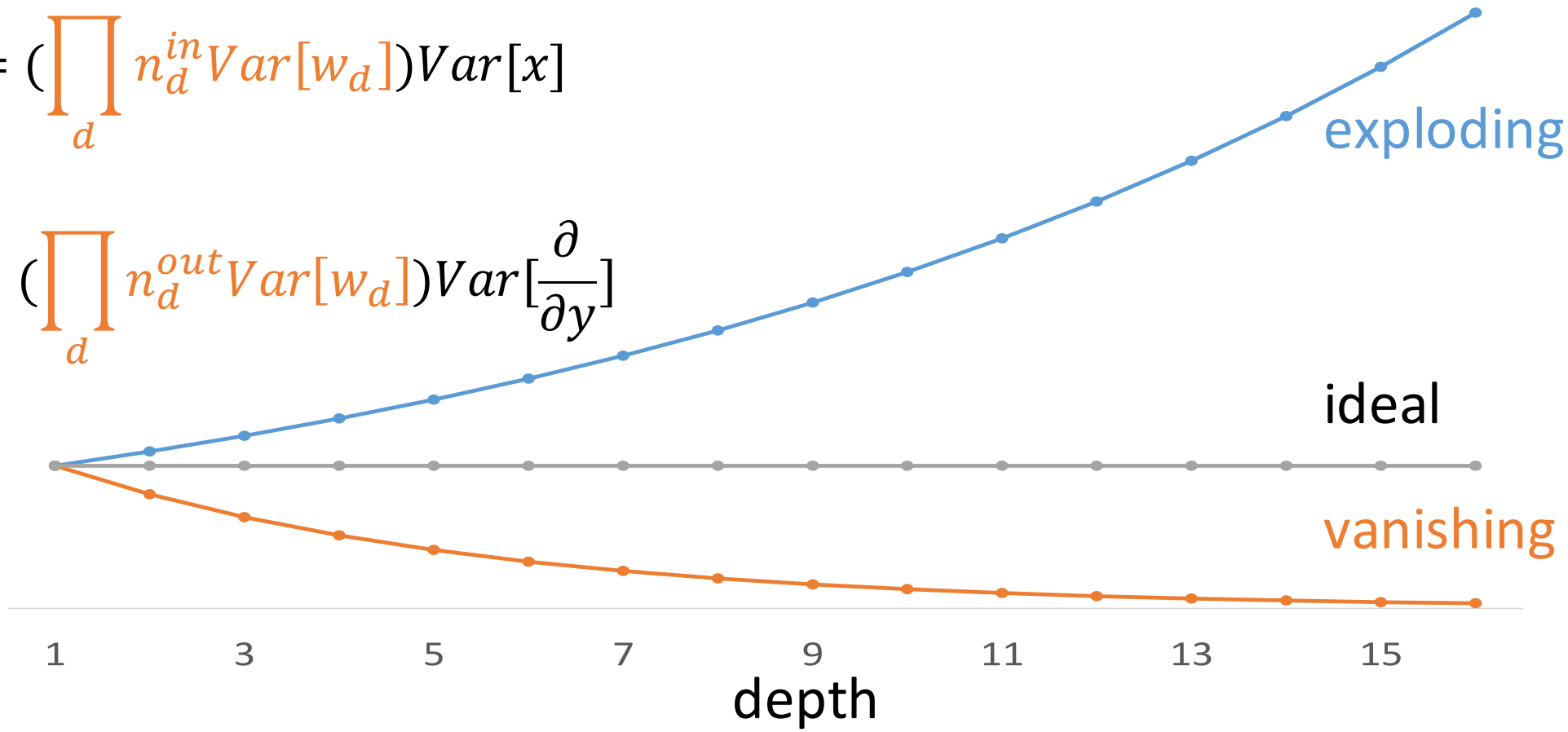
Both forward (response) and backward (gradient) signal can vanish/explode

Forward:

$$Var[y] = (\prod_d n_d^{in} Var[w_d])Var[x]$$

Backward:

$$Var\left[\frac{\partial}{\partial x}\right] = (\prod_d n_d^{out} Var[w_d])Var[\frac{\partial}{\partial y}]$$



LeCun et al 1998 “Efficient Backprop”

Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

Initialization

- Initialization under **linear** assumption

$$\prod_d n_d^{in} Var[w_d] = const_{\text{fw}} \text{ (healthy forward)}$$

and

$$\prod_d n_d^{out} Var[w_d] = const_{\text{bw}} \text{(healthy backward)}$$

➡ $\boxed{\begin{array}{c} n_d^{in} Var[w_d] = 1 \\ \text{or*} \\ n_d^{out} Var[w_d] = 1 \end{array}}$

*: $n_d^{out} = n_{d+1}^{in}$, so $\frac{const_{\text{bw}}}{const_{\text{fw}}} = \frac{n_{\text{last}}^{out}}{n_{\text{first}}^{in}} < \infty$.

It is sufficient to use either form.

"*Xavier*" init in Caffe

LeCun et al 1998 "Efficient Backprop"

Glorot & Bengio 2010 "Understanding the difficulty of training deep feedforward neural networks"

Initialization

- Initialization under **ReLU** activation

$$\prod_d \frac{1}{2} n_d^{in} Var[w_d] = const_{\text{fw}} \text{ (healthy forward)}$$

and

$$\prod_d \frac{1}{2} n_d^{out} Var[w_d] = const_{\text{bw}} \text{(healthy backward)}$$

➡

$$\frac{1}{2} n_d^{in} Var[w_d] = 1$$

or

$$\frac{1}{2} n_d^{out} Var[w_d] = 1$$

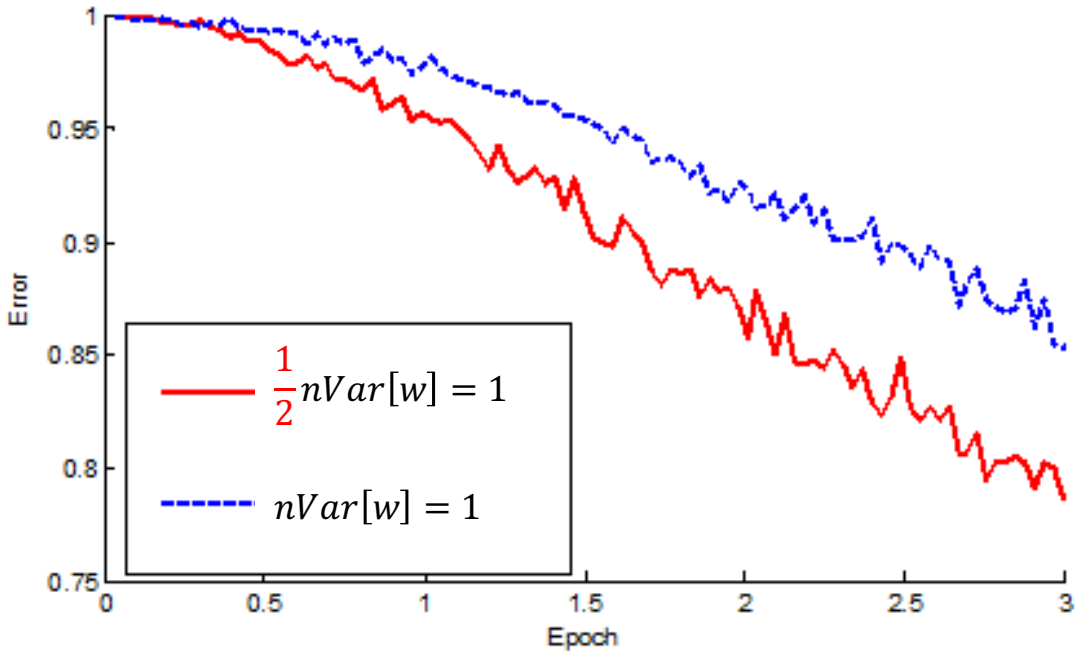
"*MSRA*" init in Caffe

With D layers, a factor of 2 per layer has exponential impact of 2^D

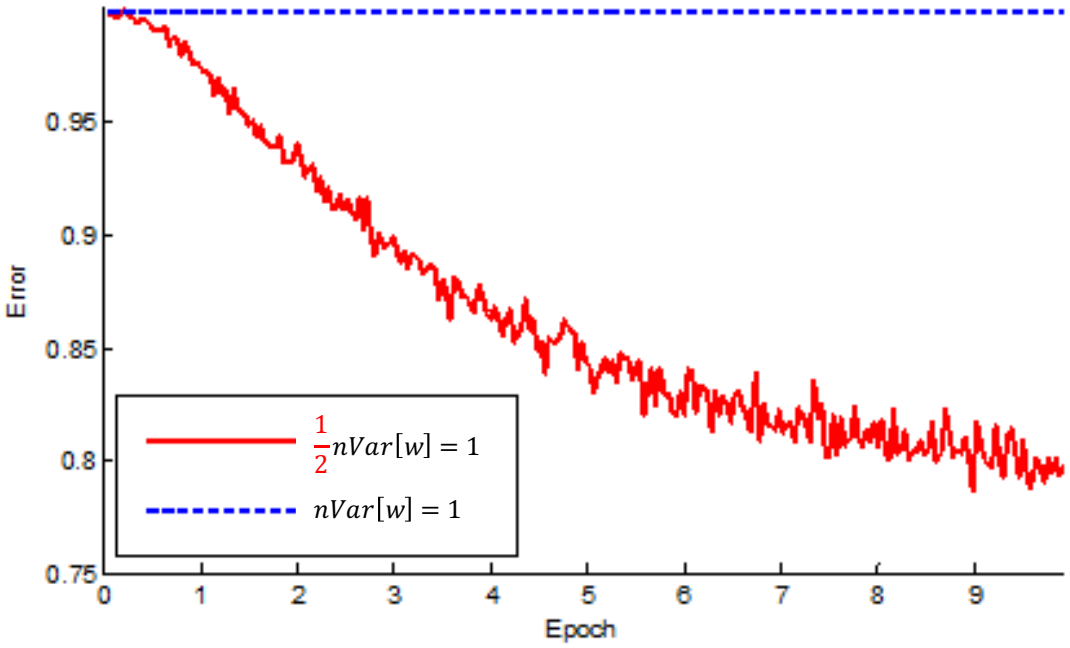
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". ICCV 2015.

Initialization

22-layer ReLU net:
good init converges faster



30-layer ReLU net:
good init is able to converge



*Figures show the beginning of training

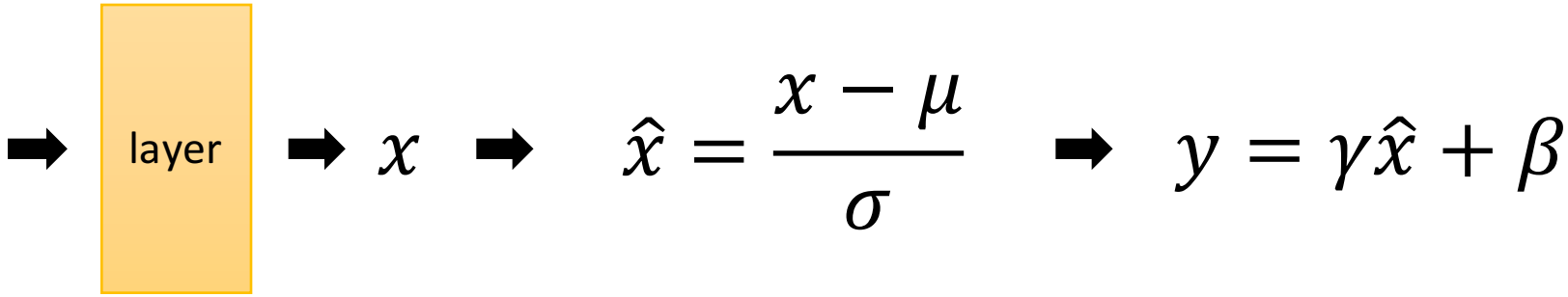
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". ICCV 2015.

Batch Normalization (BN)

- Normalizing input (LeCun et al 1998 “Efficient Backprop”)
- BN: normalizing each layer, for each mini-batch
- Greatly accelerate training
- Less sensitive to initialization
- Improve regularization

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Batch Normalization (BN)

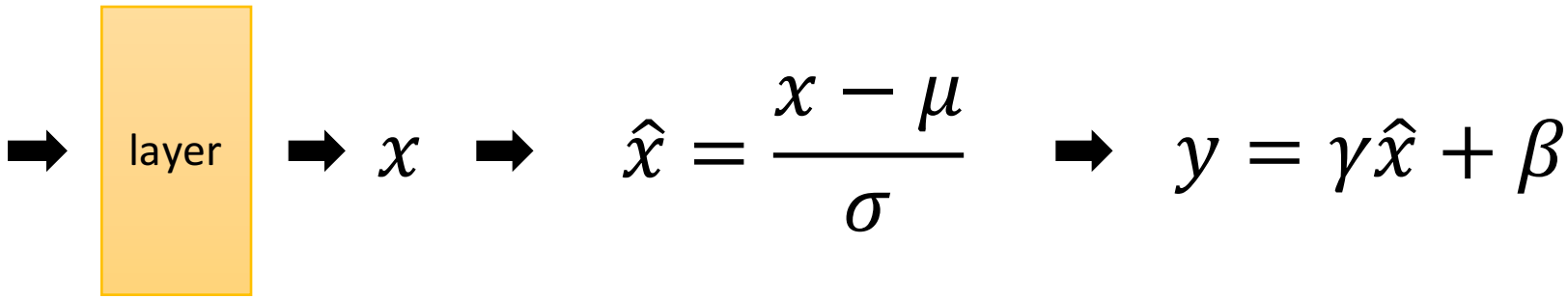


- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

- μ, σ: functions of x, analogous to responses
- γ, β: parameters to be learned, analogous to weights

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Batch Normalization (BN)



2 modes of BN:

- Train mode:
  - μ, σ are functions of x; backprop gradients
- Test mode:
  - μ, σ are pre-computed* on training set

Caution: make sure your BN is in a correct mode

*: by running average, or post-processing after training

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Batch Normalization (BN)

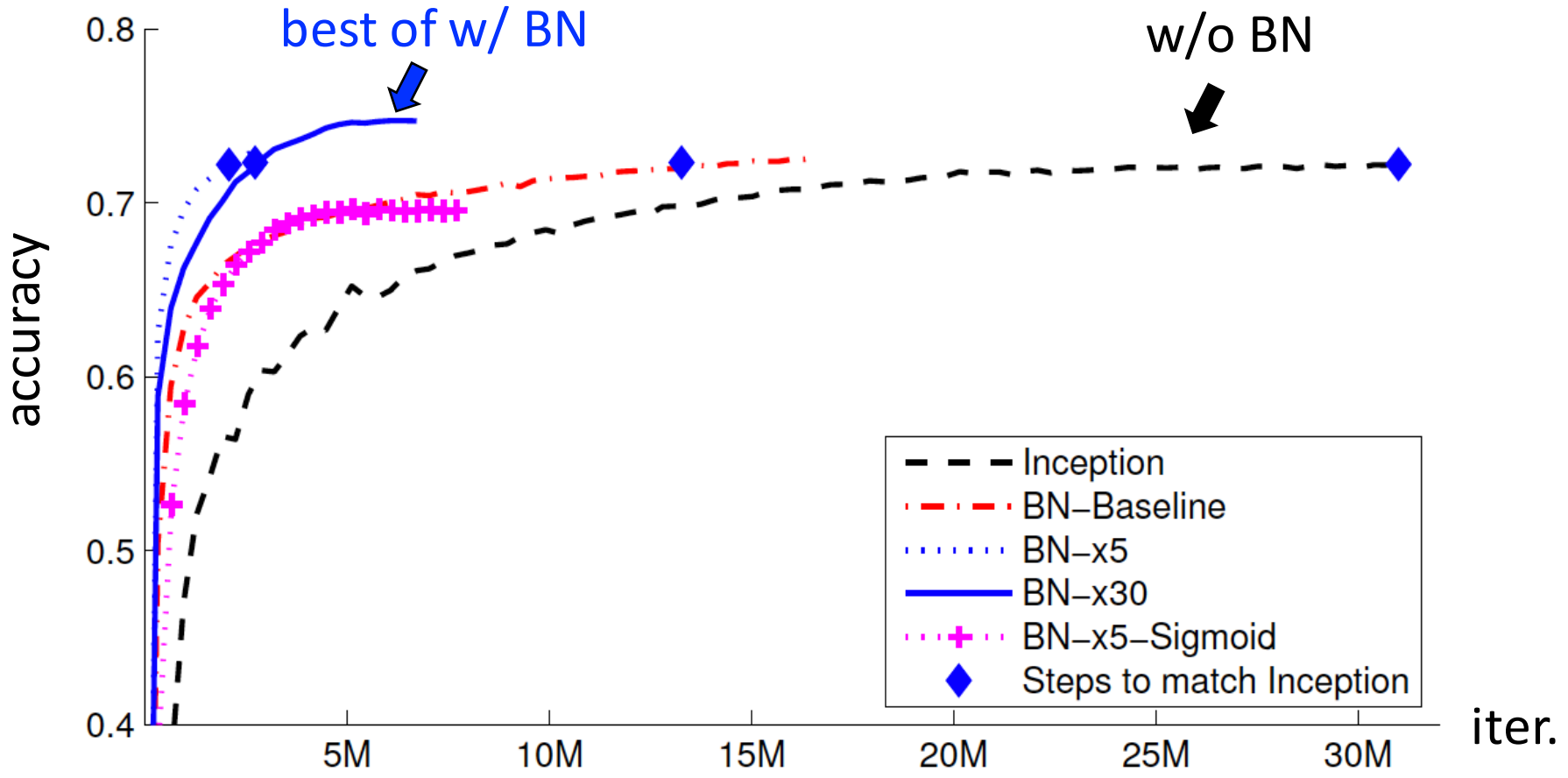


Figure taken from [S. Ioffe & C. Szegedy]

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Deep Networks with Stochastic Depth

Aim: to shrink the depth of a network during training, while keeping it unchanged during testing

The survival probability p_l is a new hyper-parameter of our training procedure. Set it to a smooth function of l: $p_l = 1 - \frac{l}{L}(1 - p_L)$

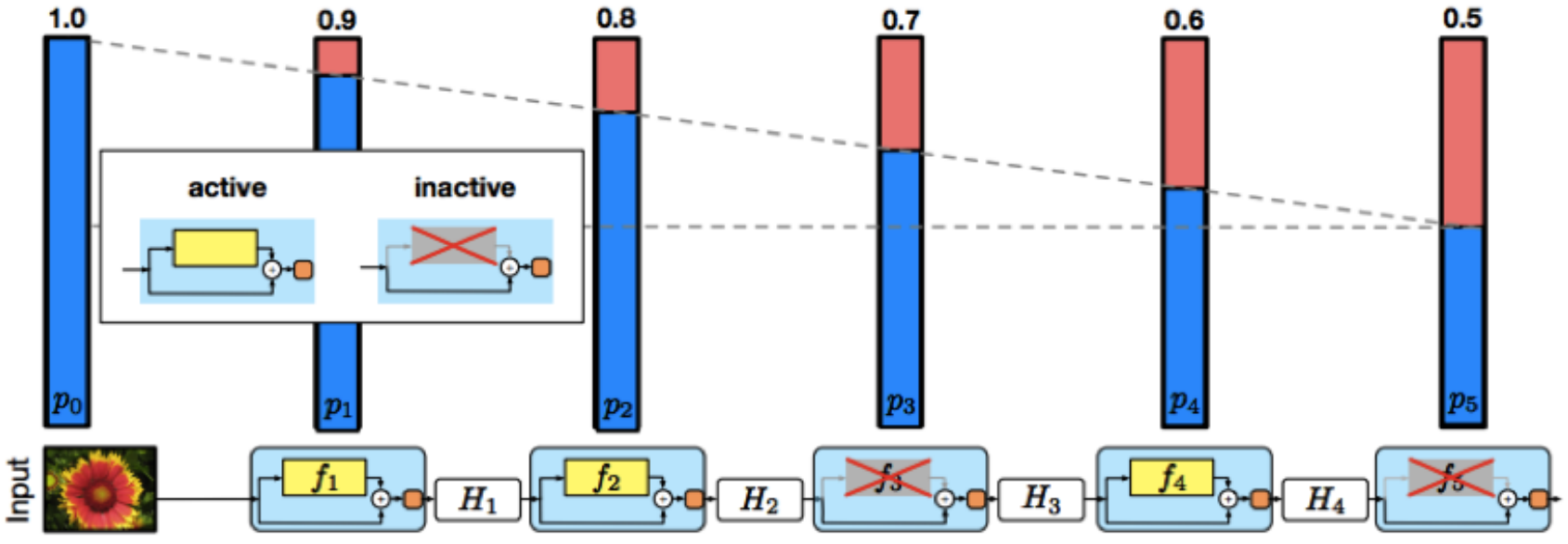
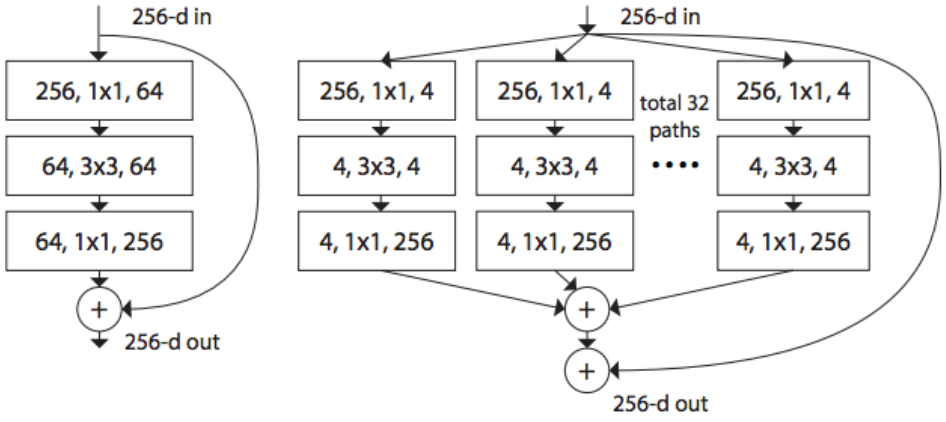


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

Aggregated Residual Transformations(CVPR2017)

ResNetX

1. Wide
2. Depth



Aggregated Residual Transformations

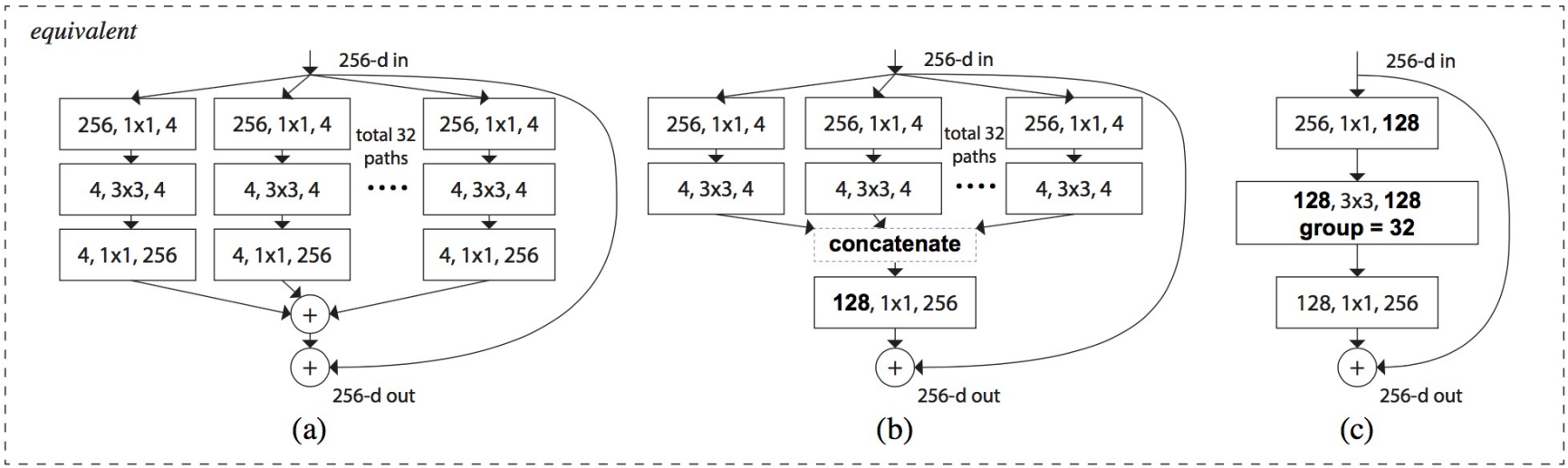


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [23]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

Multi-Residual Networks

Increase the multiplicity of the residual network.

$$x_{l+1} = x_l + f^1_{l+1}(x_l) + f^2_{l+1}(x_l) + \cdots + f^k_{l+1}(x_l)$$

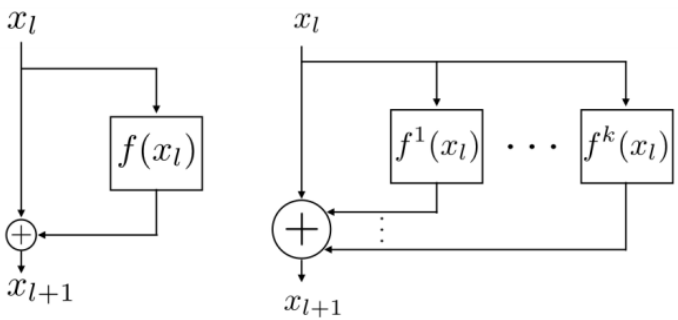


Figure 2: A residual block (left) versus a multi-residual block (right).

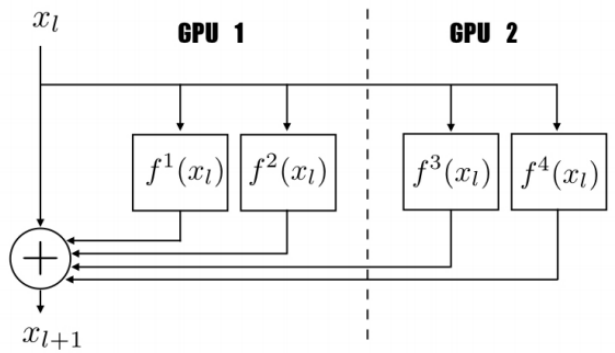
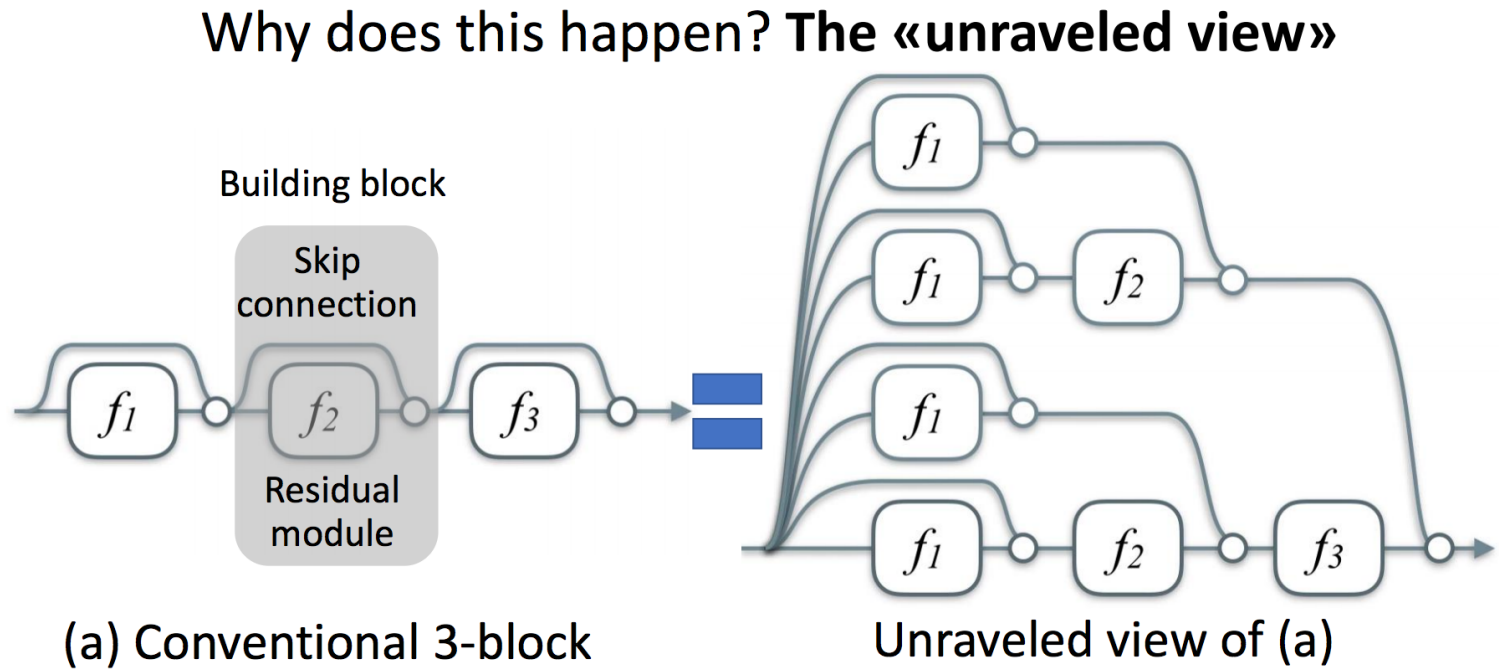


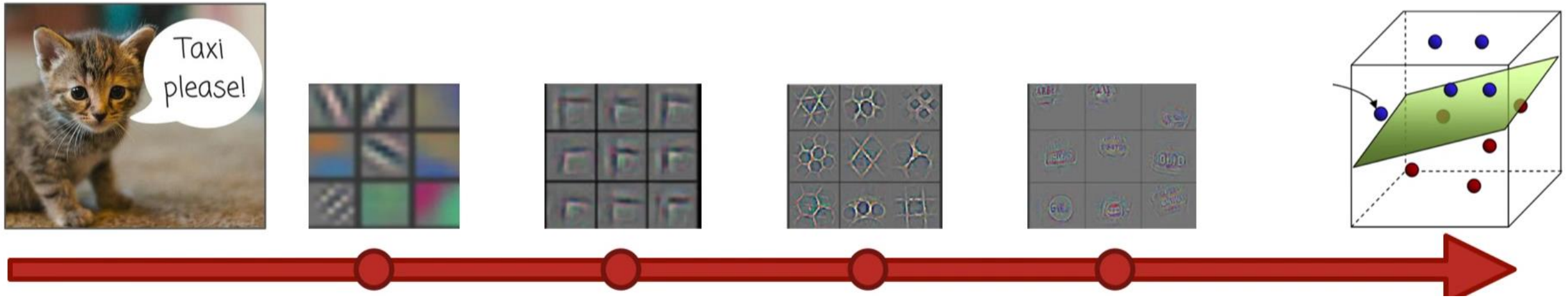
Figure 4: Model parallelization of a multi-residual block with four residual functions on two GPUs.

Residual Learning: An exponential ensemble(NIPS2016)

Residual Networks Behave Like Ensembles of Relatively Shallow Networks



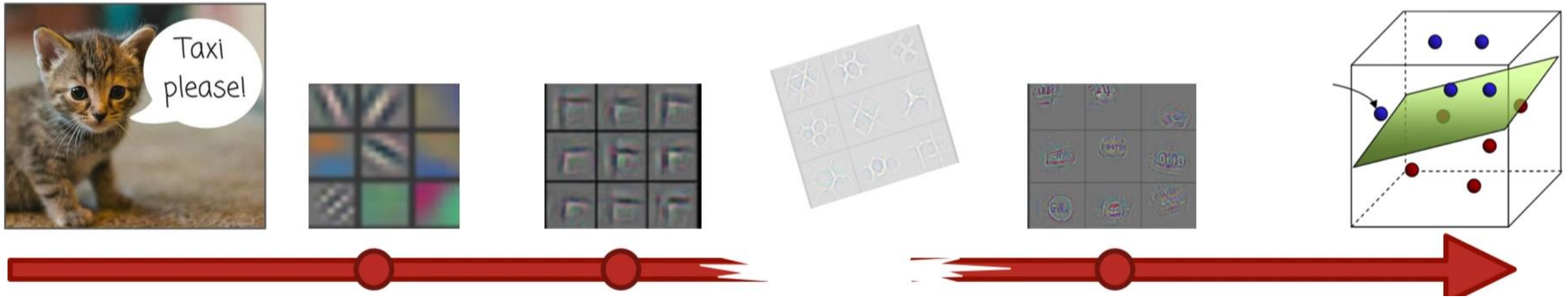
Existing systems are feed-forward, with only one path



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Existing systems are feed-forward, with only one path



What happens when we delete a step?

Slide from NIPS 2016 Spotlight Video

Existing systems are feed-forward, with only one path

Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

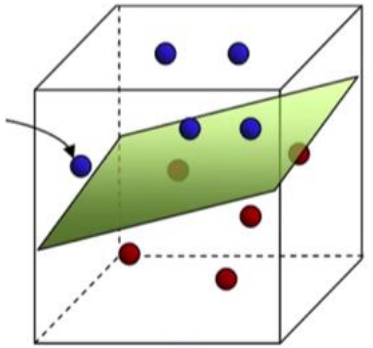
Existing systems are feed-forward, with only one path

Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Existing systems are feed-forward, with only one path

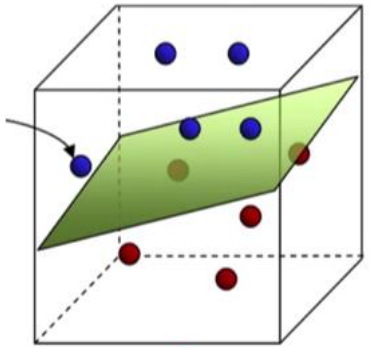
Any alternatives?



Slide adapted from NIPS 2016 Spotlight Video

Existing systems are feed-forward, with only one path

Any alternatives?
ResNets!



Slide adapted from NIPS 2016 Spotlight Video

Previous investigations:
What do we know about neural networks?

- Shown by Bengio et al. 1994 and Hochreiter 1991:
  - Length of paths affect magnitude of the gradient during backpropagation.

- Lesion studies on AlexNet by Yosinski et al. 2014:
  - Early layers little co-adaptation: General, applicable to many datasets and tasks
  - Later layers have more co-adaptation: Specific

generality -> specificity

Key takeaways

Residual networks contain many paths.

Previous networks have a single path.

Only short paths contribute gradient during training.

Vanishing gradient suppresses gradient from long paths.

Slide from NIPS 2016 Spotlight Video

Key takeaways

Residual networks contain many paths.

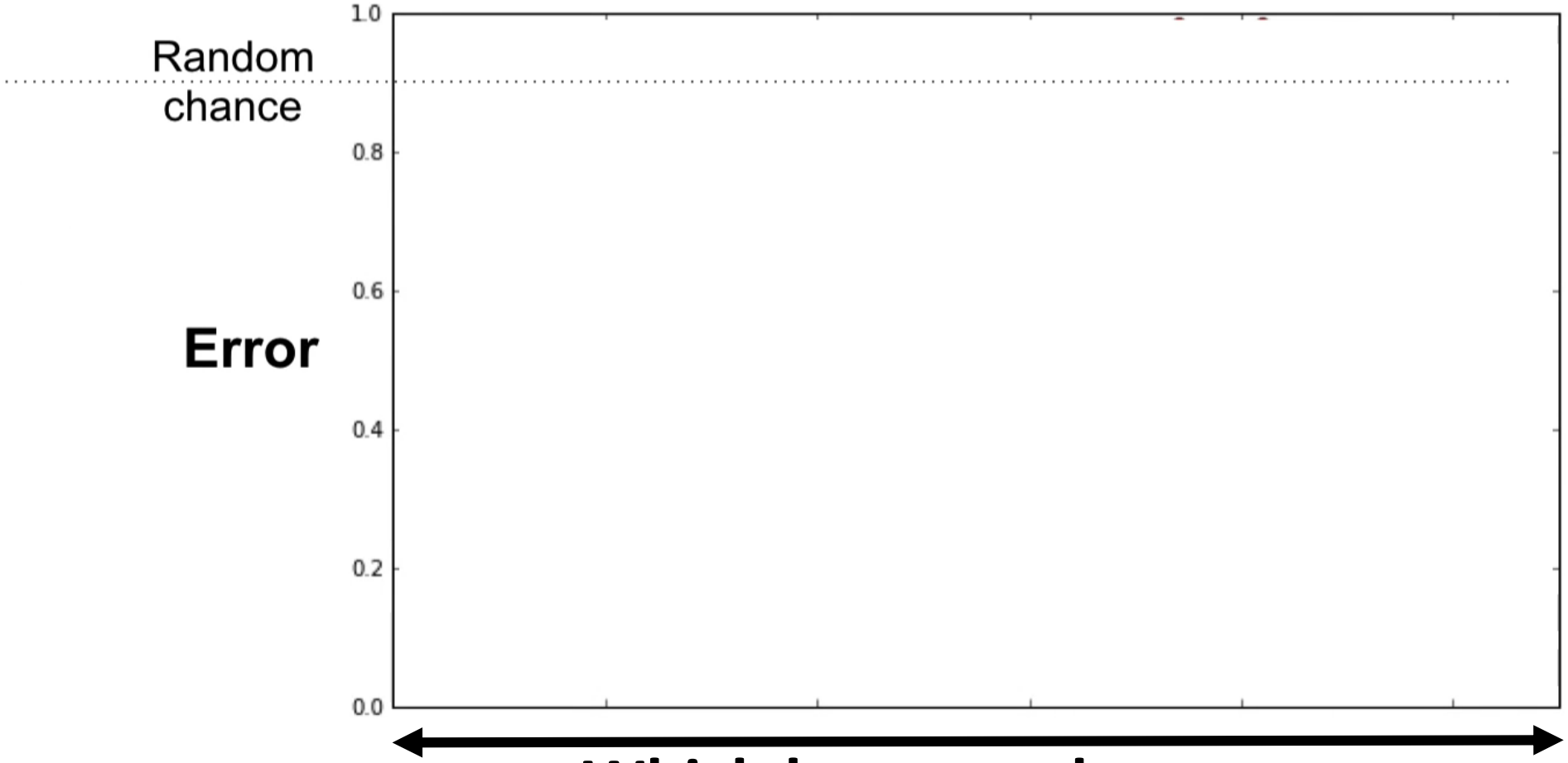
Previous networks have a single path.

Only short paths contribute gradient during training.

Vanishing gradient suppresses gradient from long paths.

Slide from NIPS 2016 Spotlight Video

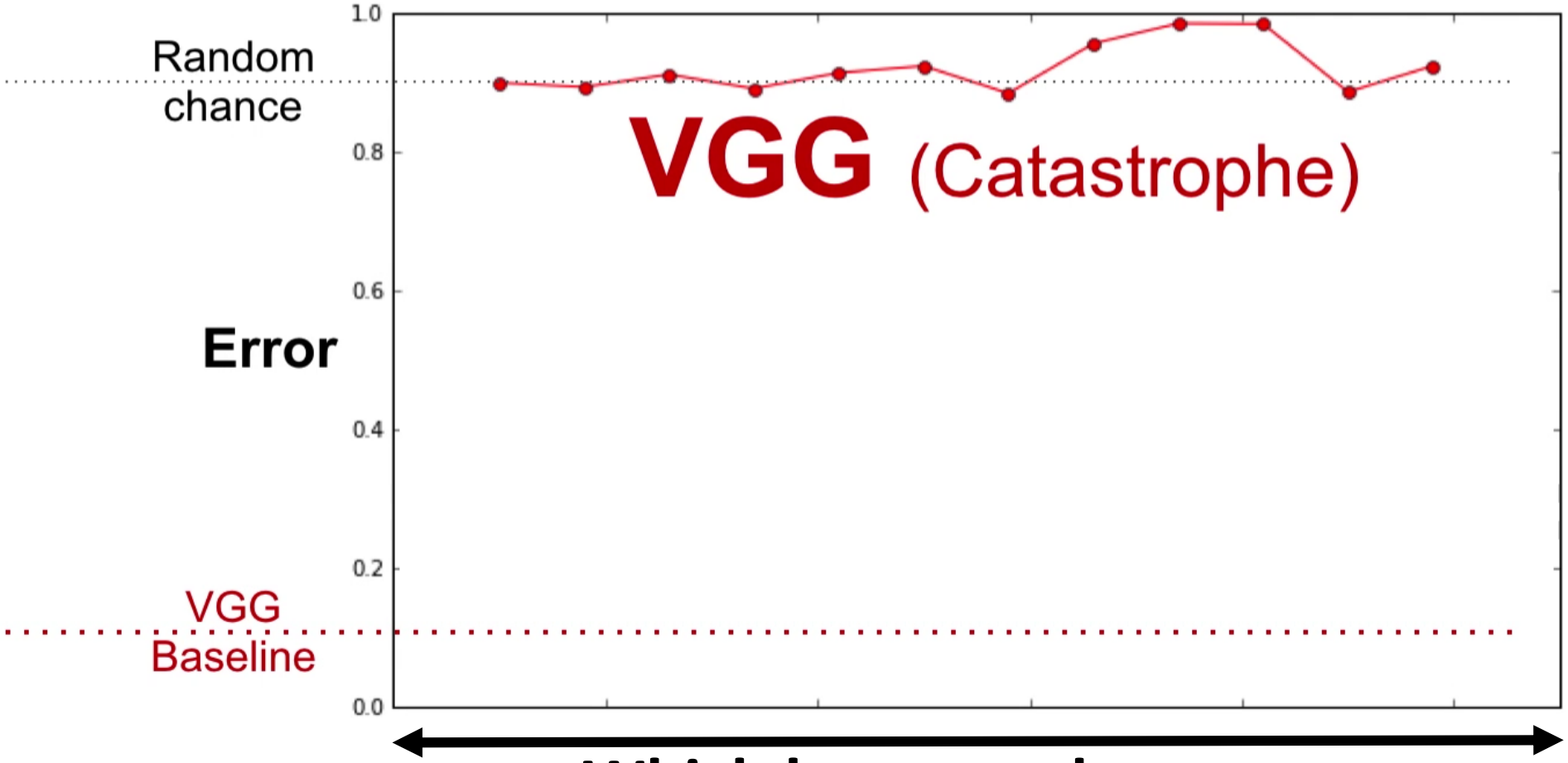
For example, **what happens when we delete layers at test time?**



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

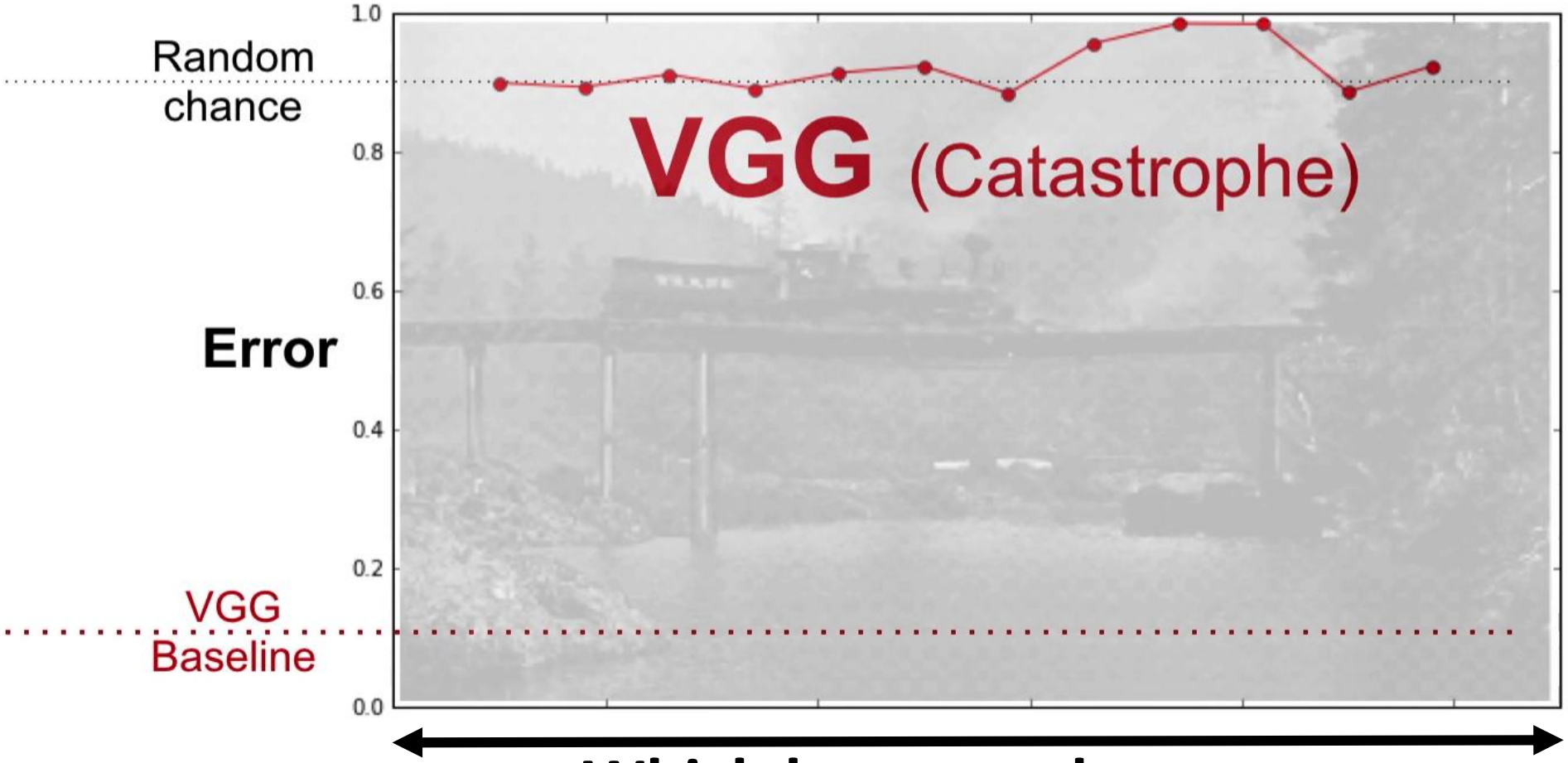
For example, **what happens when we delete layers at test time?**



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

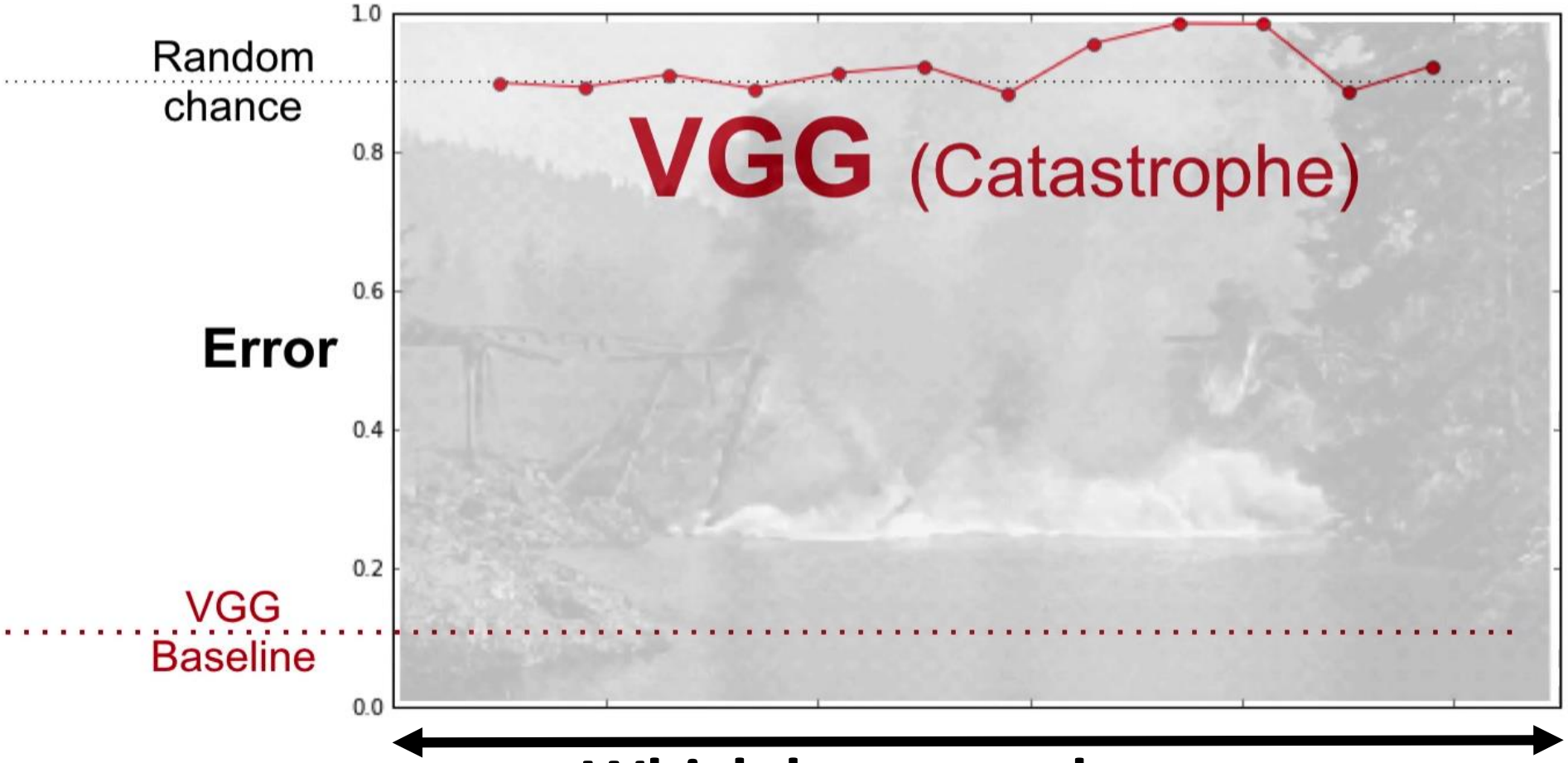
For example, **what happens when we delete layers at test time?**



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

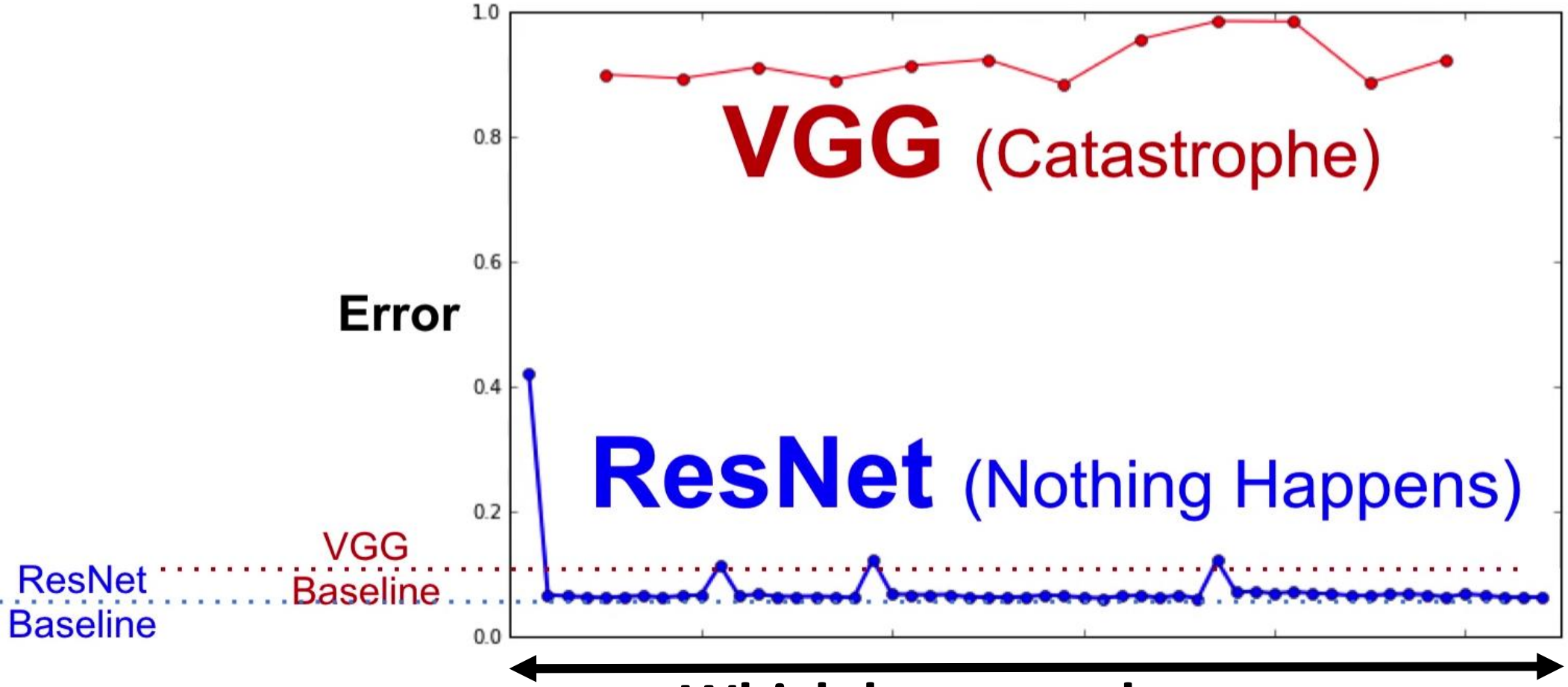
For example, **what happens when we delete layers at test time?**



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

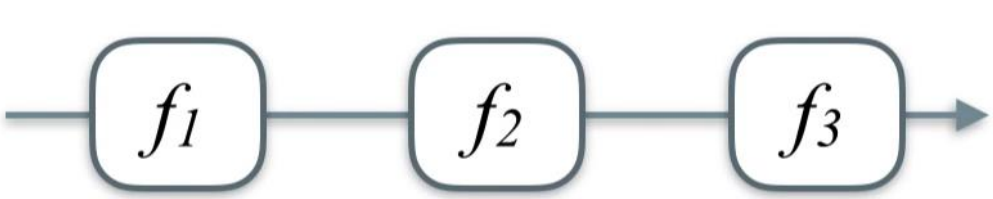
For example, **what happens when we delete layers at test time?**



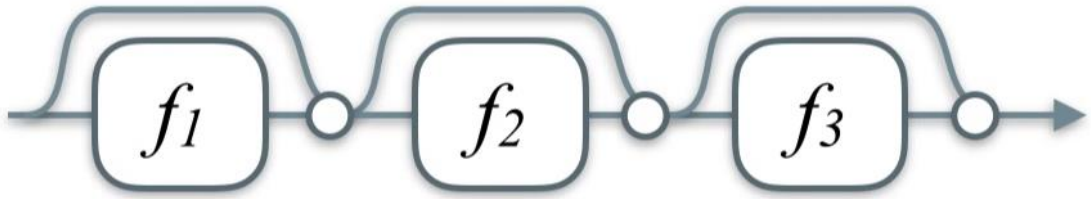
Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Why does this happen? **The «unraveled view»**



VGG

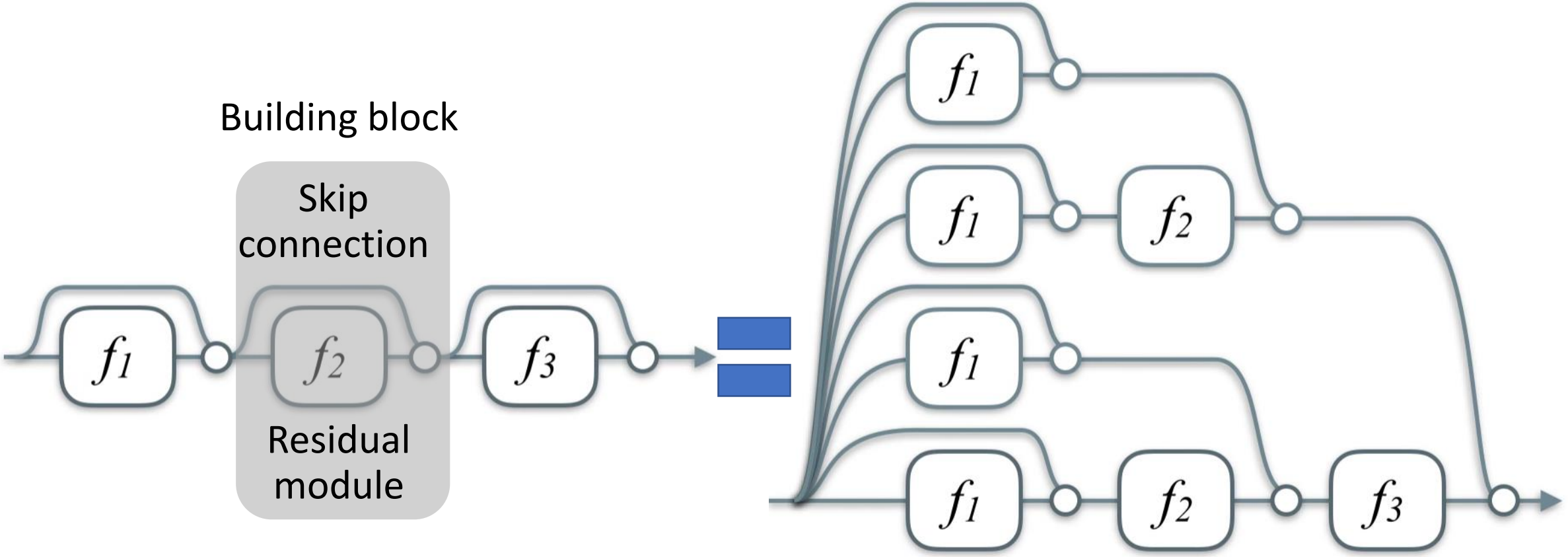


ResNet

Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Why does this happen? **The «unraveled view»**

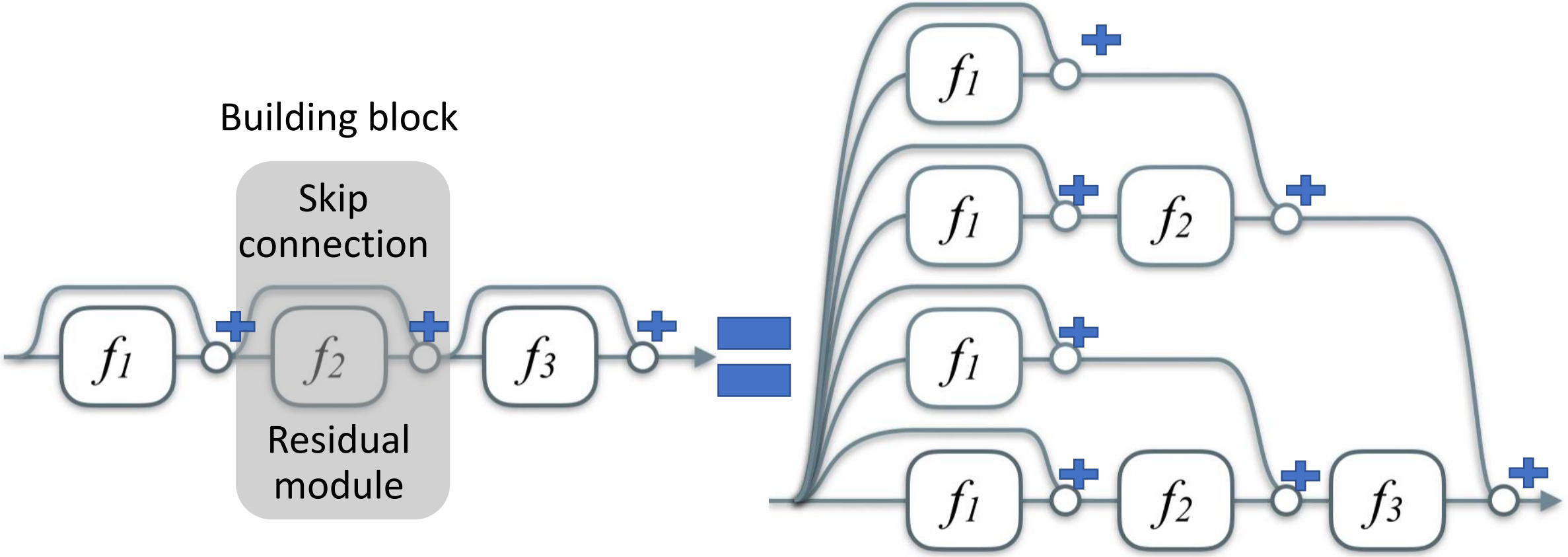


(a) Conventional 3-block residual network

Unraveled view of (a)

Slide adapted from NIPS 2016 Spotlight Video

Why does this happen? **The «unraveled view»**



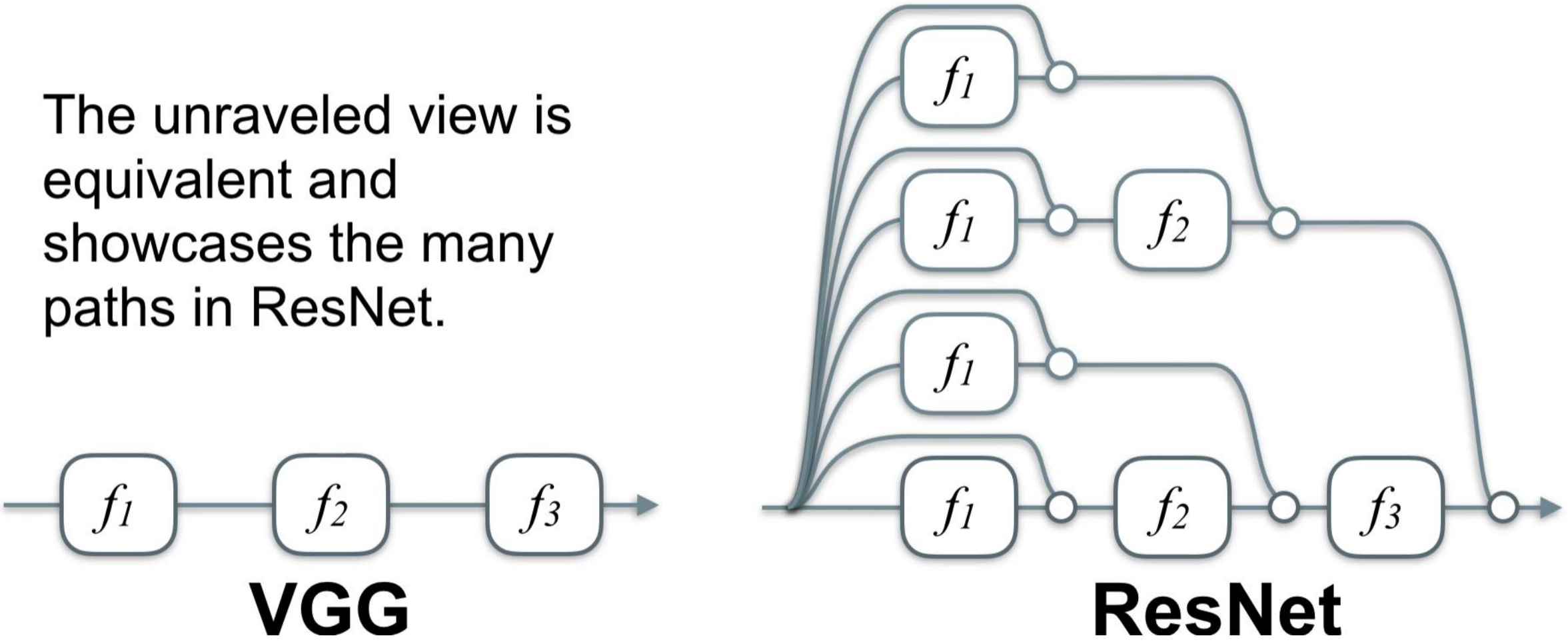
(a) Conventional 3-block residual network

Unraveled view of (a)

Slide adapted from NIPS 2016 Spotlight Video

Why does this happen? **The «unraveled view»**

The unraveled view is equivalent and showcases the many paths in ResNet.



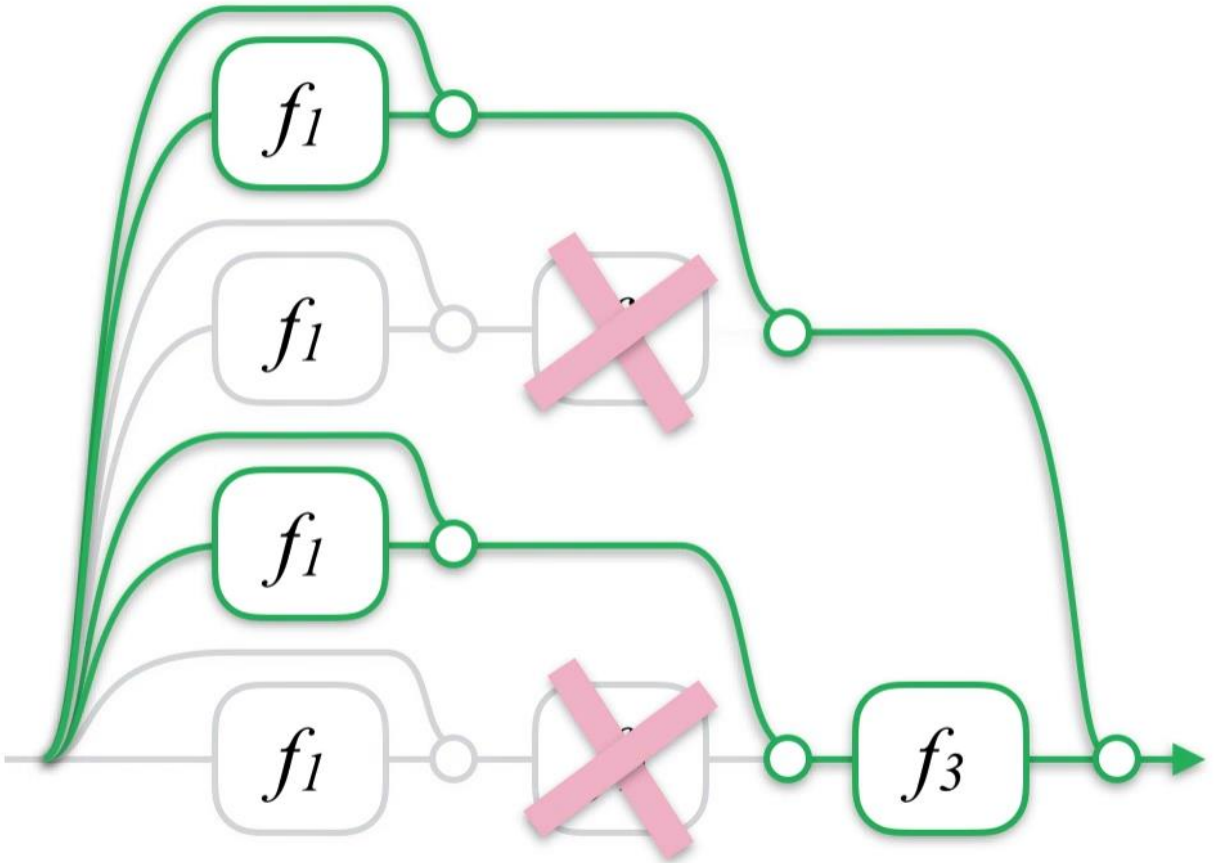
Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Deletion of one layer

VGG

All paths are affected



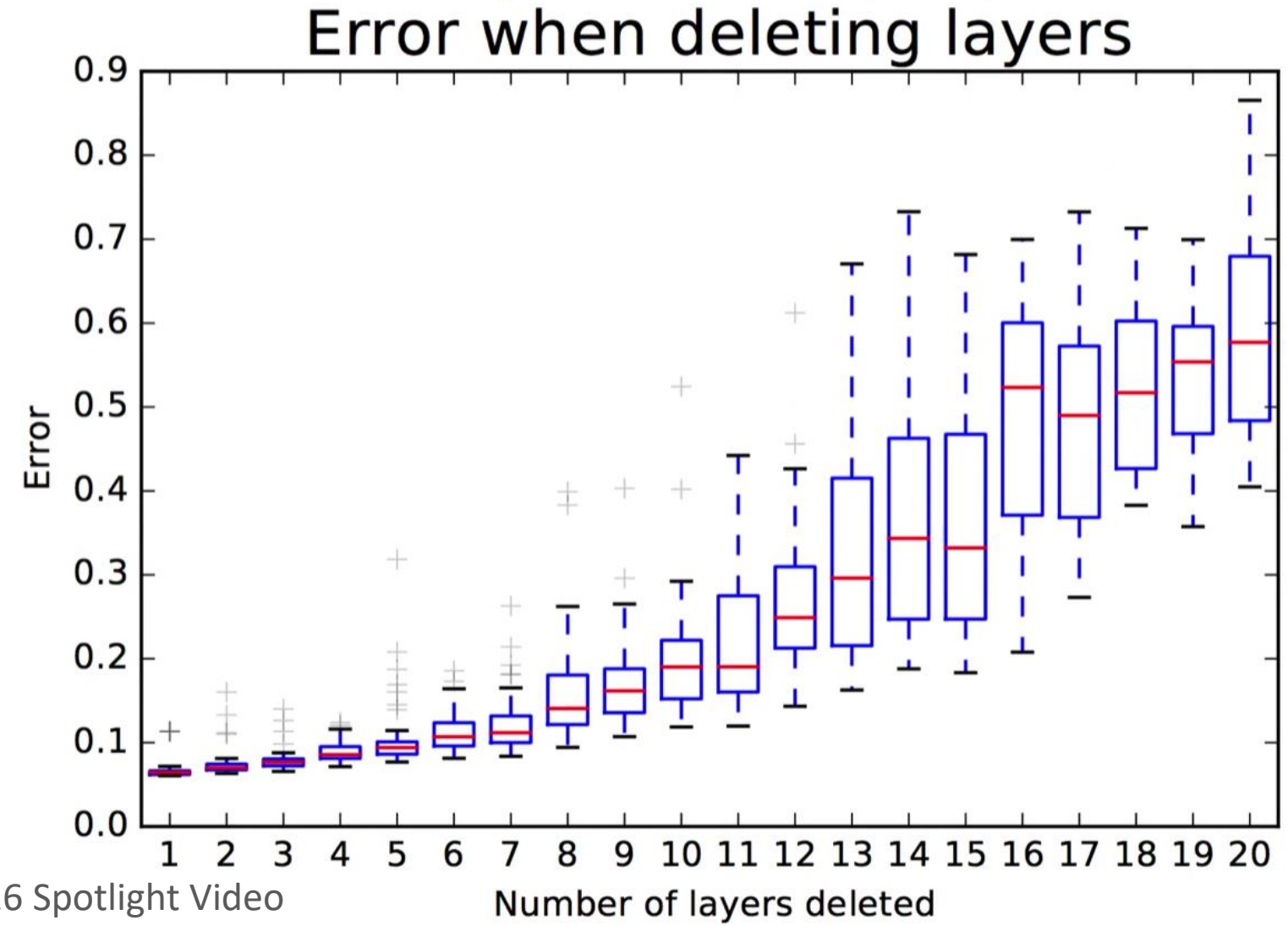
ResNet

Only **half** of the paths are affected

Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

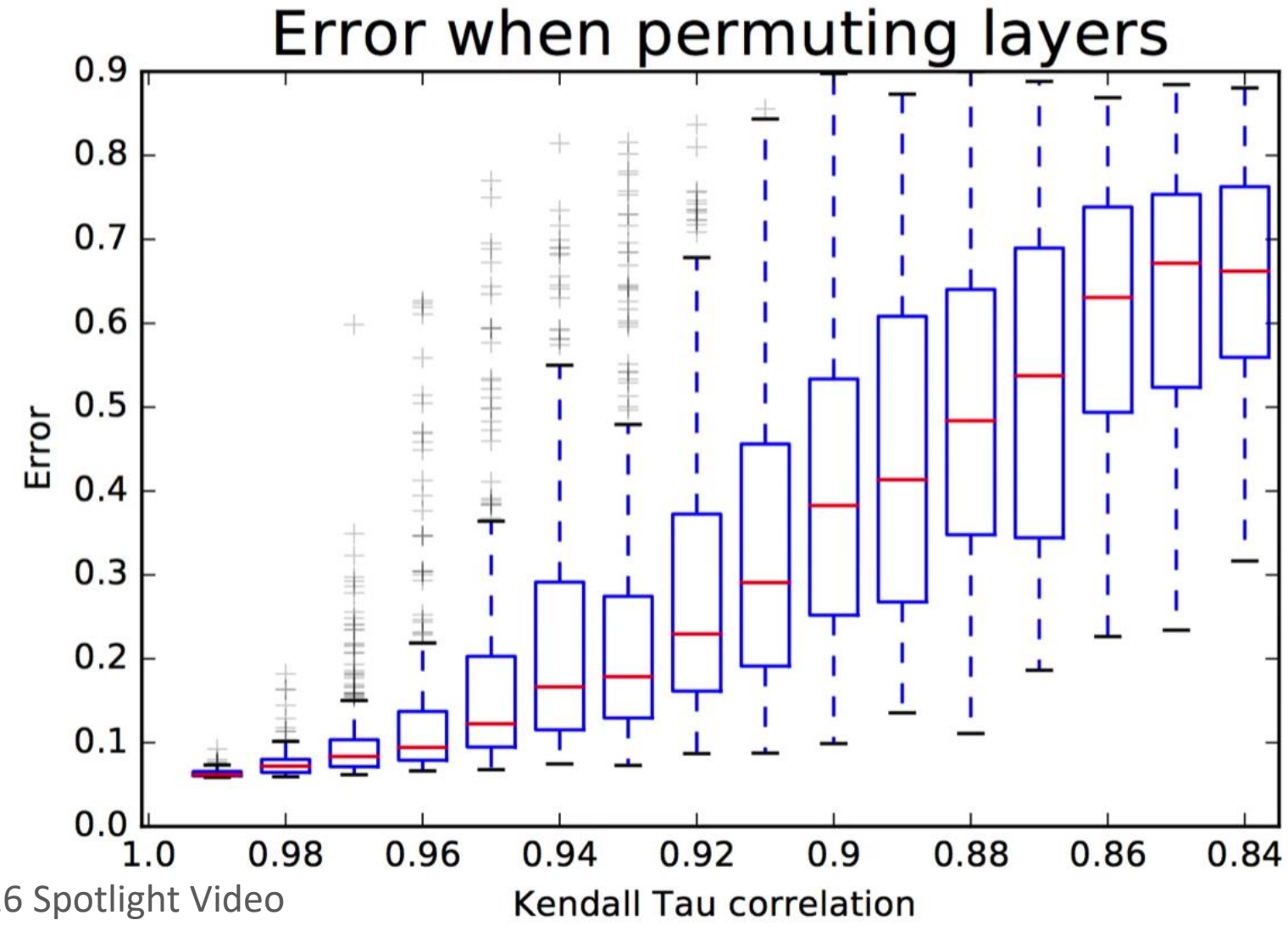
Performance varies smoothly when deleting **several** layers.



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Performance varies smoothly when **re-ordering** layers.



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Conclusion 1:

- Residual Networks consist of many paths.

- Although trained jointly, they do not strongly depend on each other: Ensemble-like behavior

Slide adapted from NIPS 2016 Spotlight Video

Key takeaways

Residual networks contain many paths.

Previous networks have a single path.

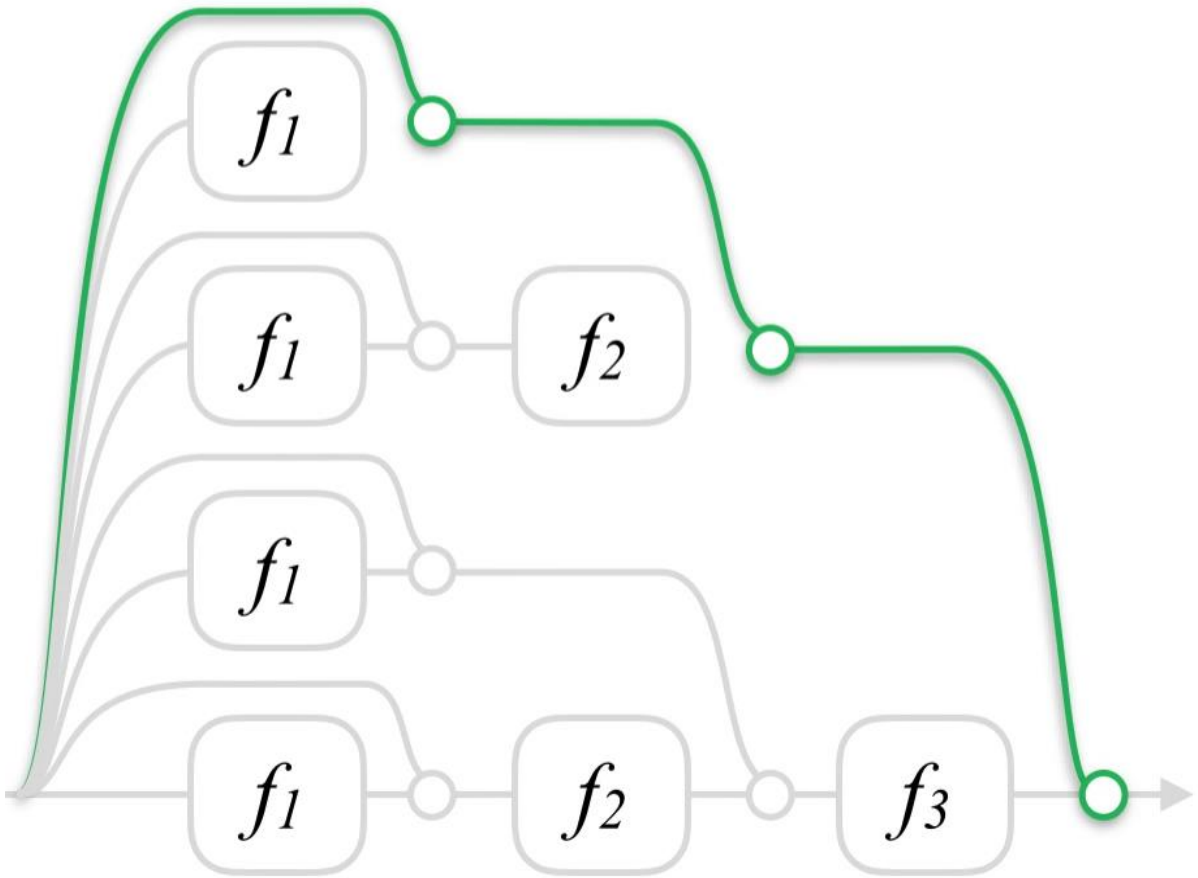
Only short paths contribute gradient during training.

Vanishing gradient suppresses gradient from long paths.

Slide from NIPS 2016 Spotlight Video

Distribution of path length

There are very few **short paths…**



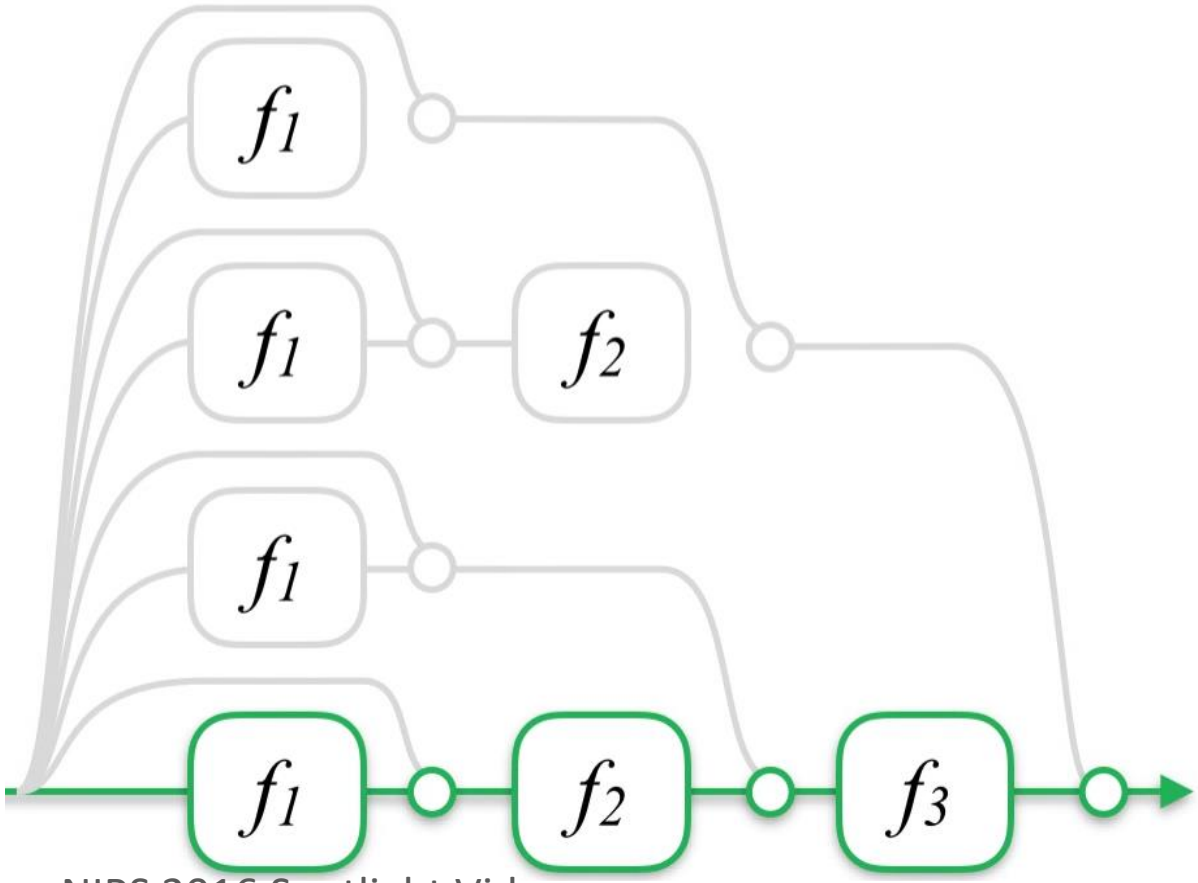
Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Distribution of path length

There are very few **short paths…**

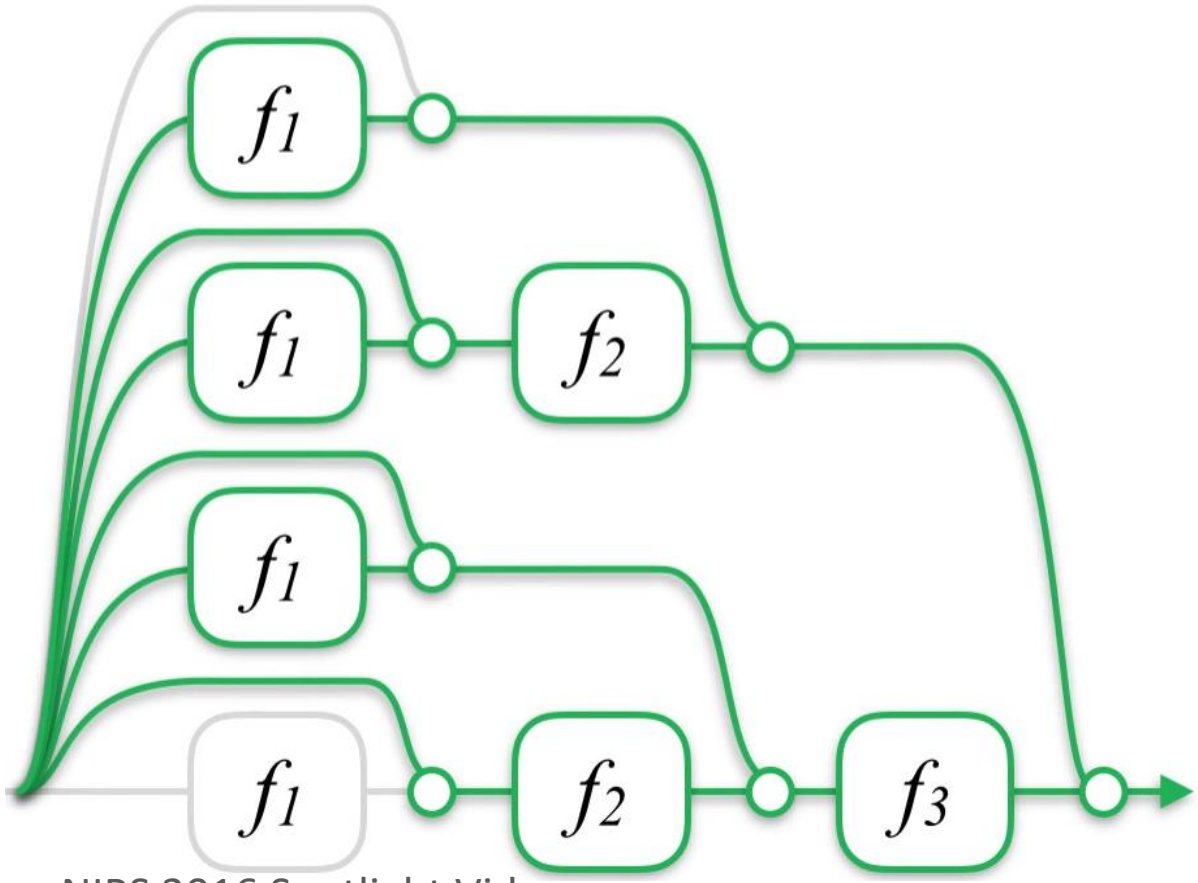
And very few **long paths…**



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Distribution of path length



There are very few **short paths…**

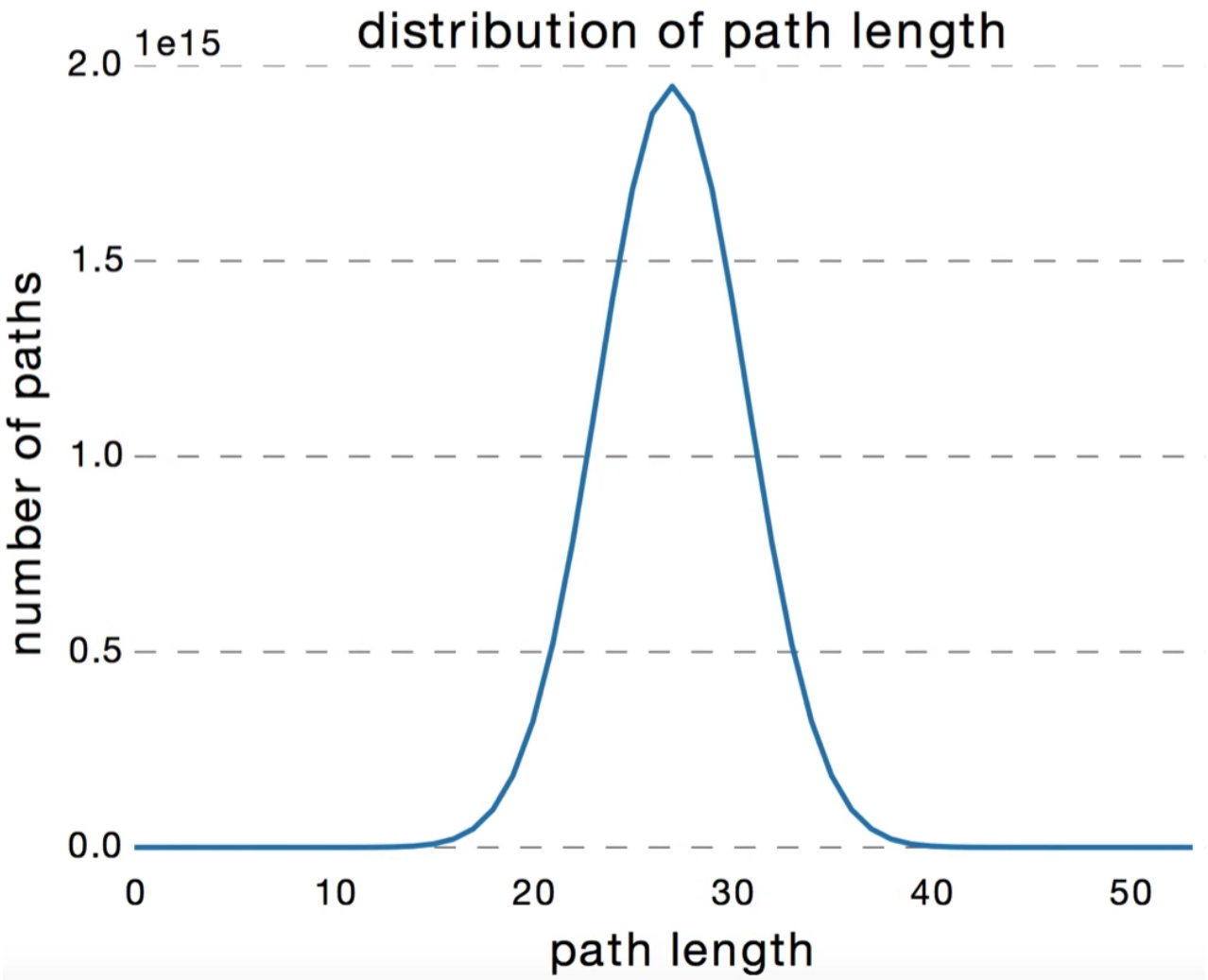
And very few **long paths…**

Most paths are **medium length!**

Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Distribution of path length



There are very few **short paths…**

And very few **long paths…**

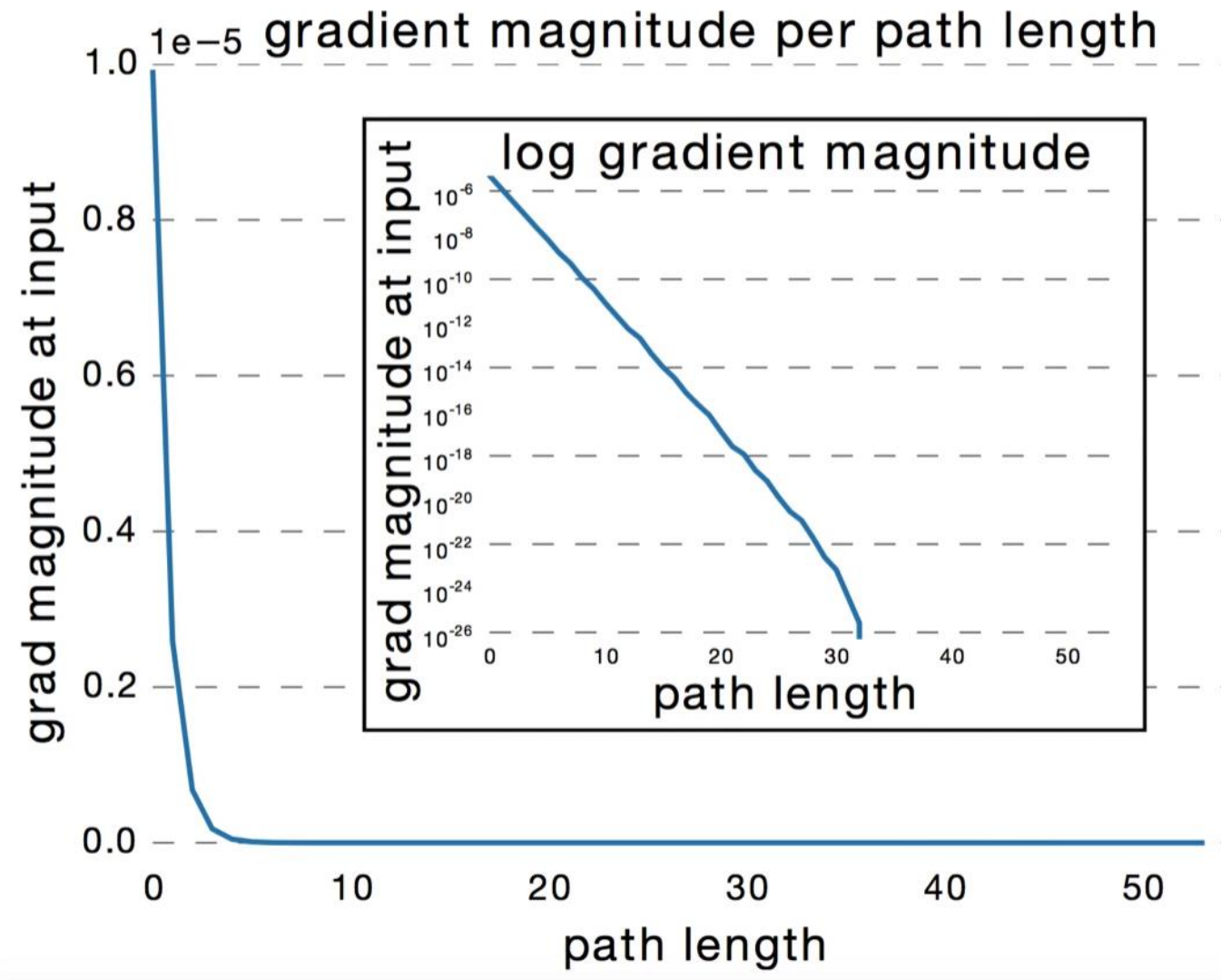
Most paths are **medium length!**

Paths length follows a **binomial distribution.**

Slide from NIPS 2016 Spotlight Video

Vanishing gradient

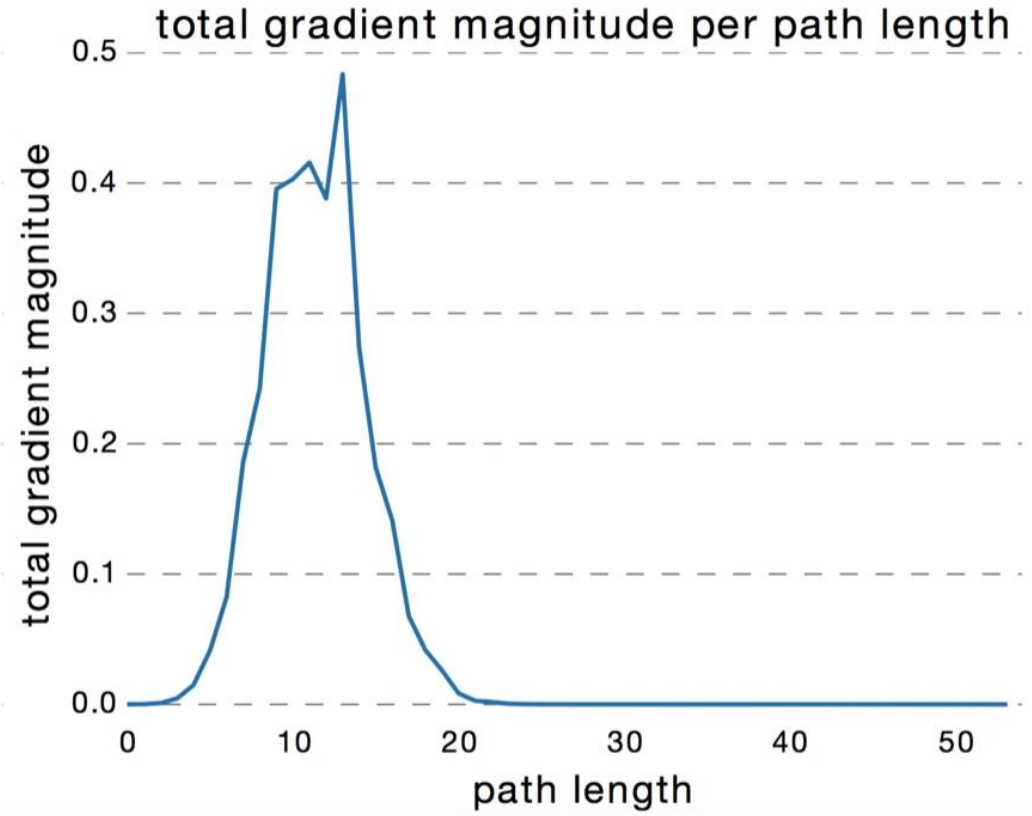
The gradient magnitude **decreases exponentially** with increasing path length.



Slide from NIPS 2016 Spotlight Video

Andreas Veit, Michael Wilber & Serge Belonie. NIPS 2016.

Gradient during training with respect to path lengths



Combining the path length distribution and the vanishing gradients, one can observe that most of the gradient comes from relatively short paths.

Slide from NIPS 2016 Spotlight Video

Conclusion 2:

- Most paths through a ResNet are relatively short.
- During training, gradients only flow through short paths.

Slide from NIPS 2016 Spotlight Video

Highway and Residual Network: An Unrolled Iterative Estimation

Idea:

- The level of representation stays the same within each stages!
- The level of representation is changed by the use of a projection to change dimensionality.

Suppose $E[A - C] = E[B - C] = 0$ and
$Var[A - C] = \sigma_A^2$, $Var[B - C] = \sigma_B^2$ and $cov[A, B] = \sigma_{A,B}^2$

Looking for $q(A, B) = q_1 A + q_2 B (q_1 + q_2 = 2)$(so that $E[q - C] = 0$) minimize the variance

$$\min_{q_1, q_2} Var[q_1 A + q_2 B - C]$$

$$subject.to. q_1 + q_2 = 1$$

An Unrolled Iterative Estimation

Considering the Lagrangian multipliers:

$$Var[q_1 A + q_2 B - C] - \lambda(q_1 + q_2 - 1)$$

Then we have

$$2q_1\sigma_A^2 + 2q_2\sigma_{AB}^2 - \lambda = 0$$
$$2q_2\sigma_B^2 + 2q_1\sigma_{AB}^2 - \lambda = 0$$

That is to say

$$q_1 = \frac{\sigma_B^2 - \sigma_{AB}^2}{\sigma_A^2 - 2\sigma_{AB}^2 + \sigma_B^2}$$
$$q_2 = \frac{\sigma_A^2 - \sigma_{AB}^2}{\sigma_A^2 - 2\sigma_{AB}^2 + \sigma_B^2}$$

An Unrolled Iterative Estimation

We rewrite the estimation formula

$$q = \frac{\alpha_1}{\alpha_1 + \alpha_2} A + \frac{\alpha_2}{\alpha_1 + \alpha_2} B$$

Here $\alpha = \sigma_B^2 - \sigma_A^2, \alpha_2 = \sigma_A^2 - \sigma_{AB}^2$, which is similar to the highway net. For Residual Nets $E[a_i^k - A_i] - E[a_i^{k-1} - A_i] = 0$ Then $E[residual] = 0$ which can be consider reasonable when using batch normalization.

An Unrolled Iterative Estimation

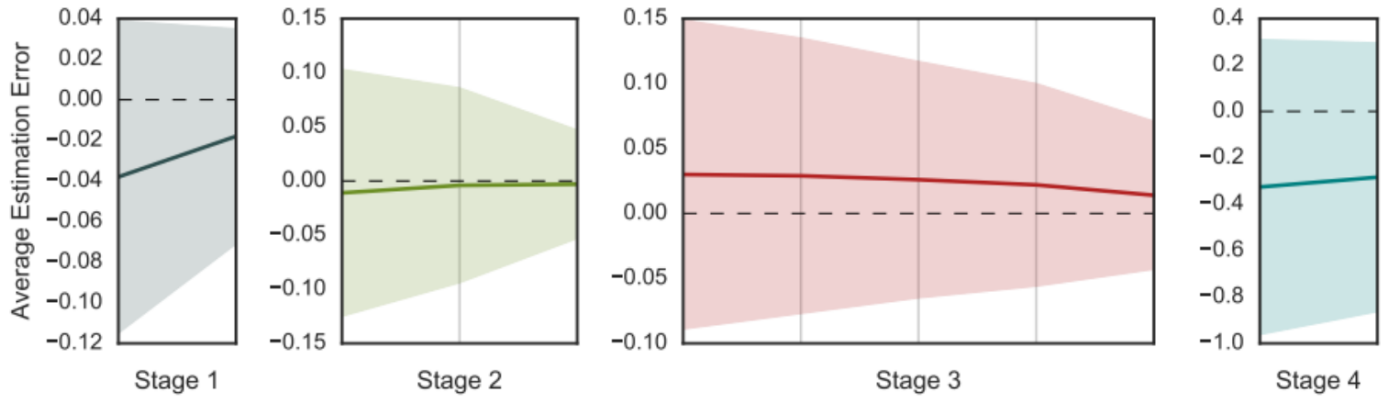
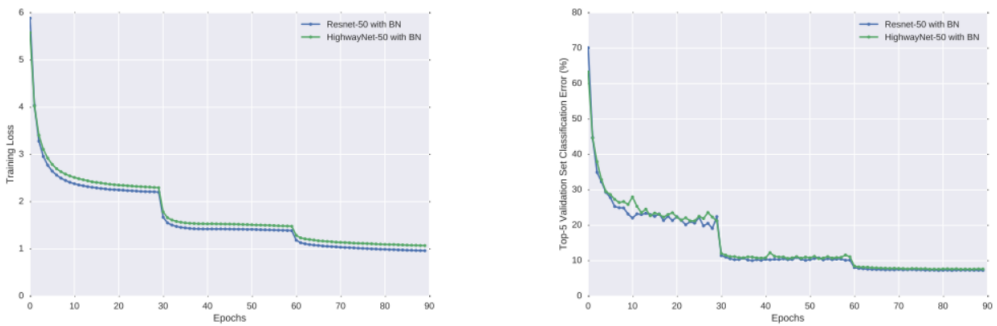


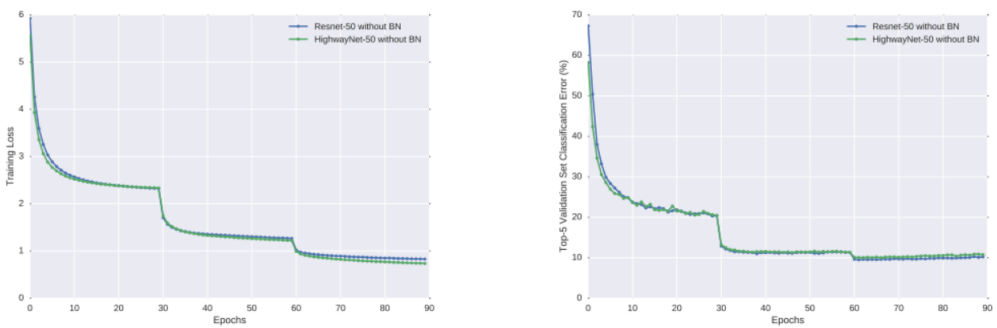
Figure 3: Experimental corroboration of Equation 1. The average estimation error – an empirical estimate of the LHS in Equation 1 – for each block of each stage (x-axis). It stays close to zero in all stages of a 50-layer ResNet trained on the ILSVRC-2015 dataset. The standard deviation of the estimation error decreases as depth increases in each stage (left to right), indicating iterative refinement of the representations.

BN

We find that without BN both networks reach an even lower training error than before while performing worse on the validation set indicating increased overfitting for both. This shows that BN is not necessary for training these networks and does not speed up learning. Interestingly, the effect is more pronounced for the Highway network, which now fits the data better than the ResNet.

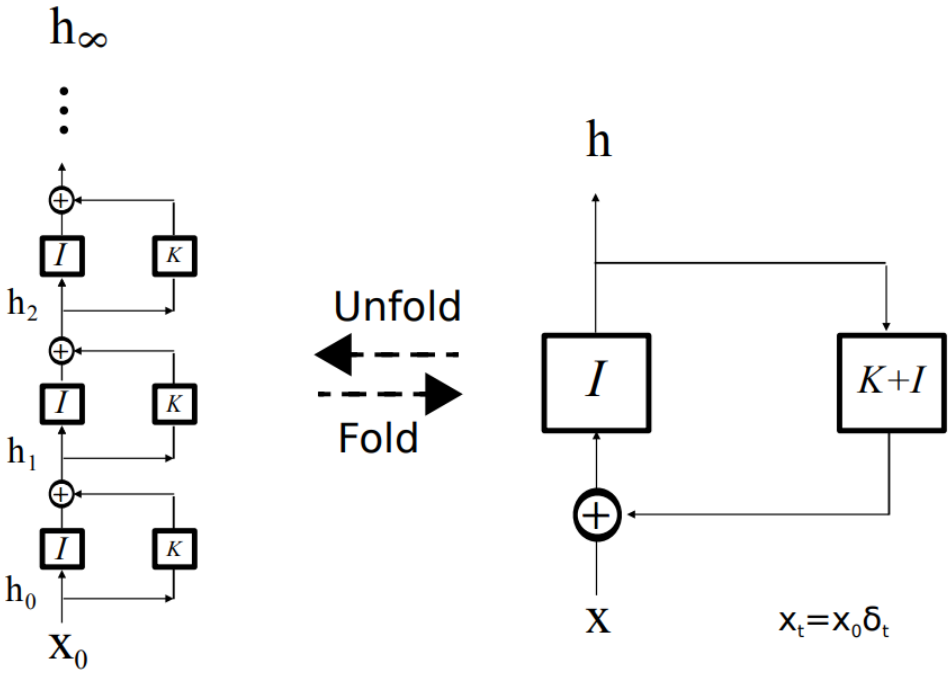


(a) with batch normalization



(b) without batch normalization

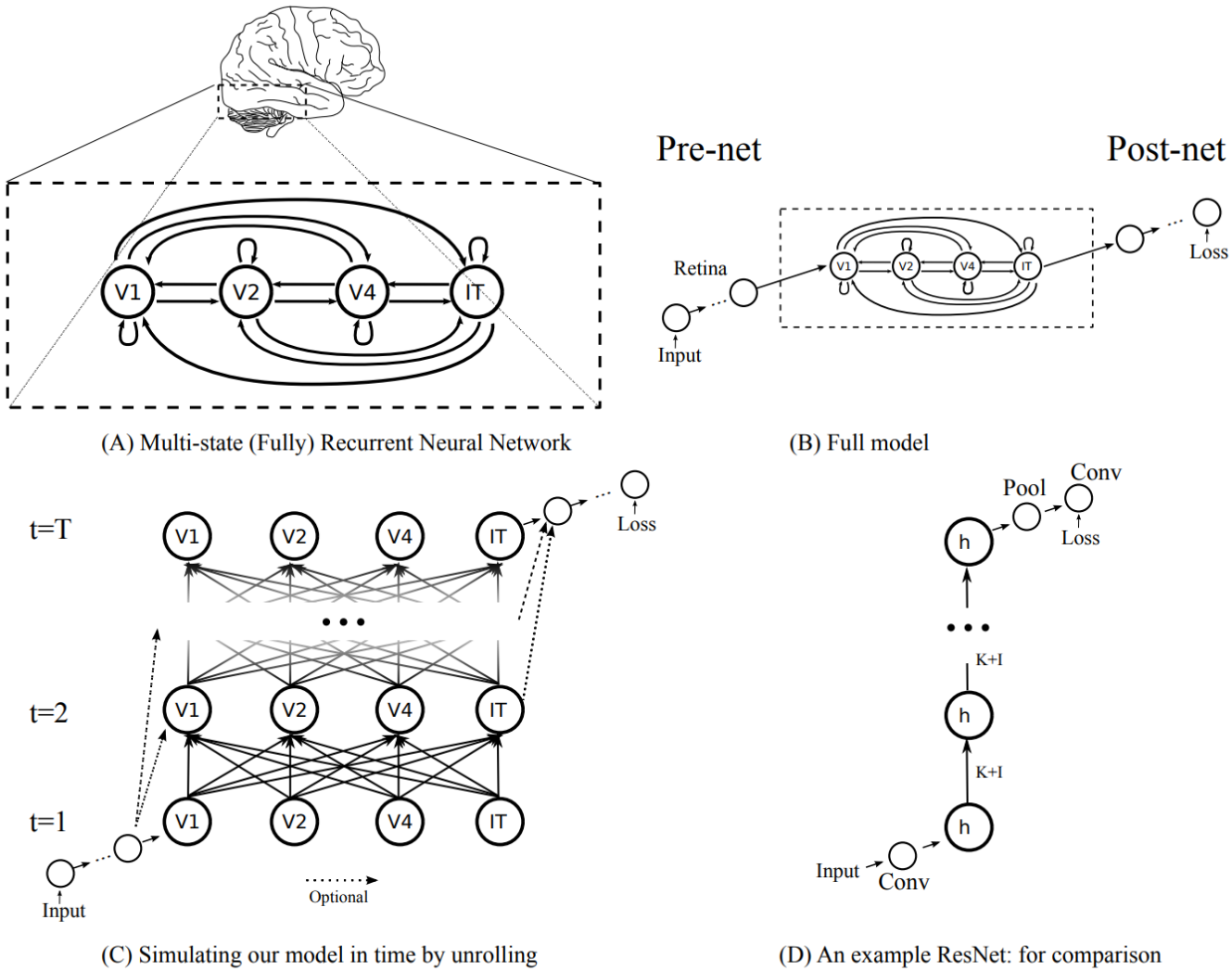
ResNet with shared weights



(A) ResNet with shared weights (B) ResNet in recurrent form

Figure 1: A formal equivalence of a ResNet (A) with weight sharing and a RNN (B). I is the identity operator. K is an operator denoting the nonlinear transformation called f in the main text. x_t is the value of the input at time t. δ_t is a Kronecker delta function.

Visual Cortex



(A) Multi-state (Fully) Recurrent Neural Network

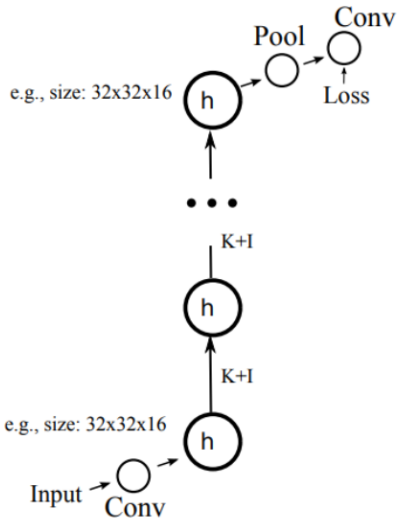
(B) Full model

(C) Simulating our model in time by unrolling

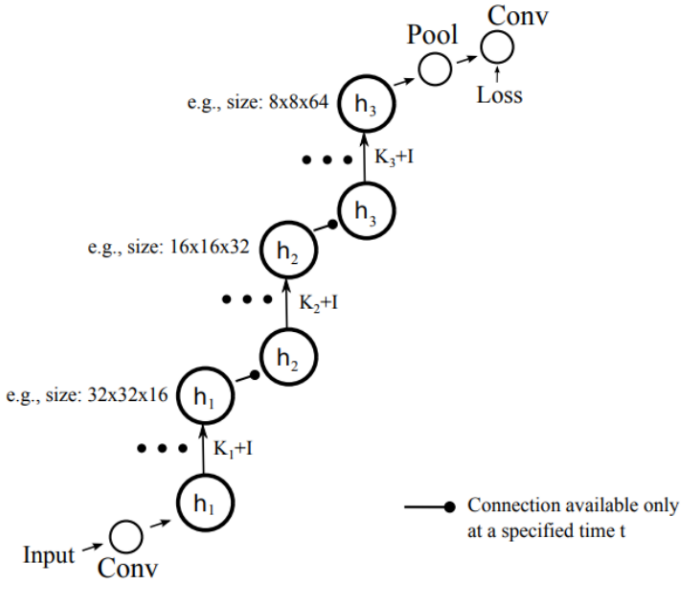
(D) An example ResNet: for comparison

Figure 2: Modeling the ventral stream of visual cortex using a multi-state fully recurrent neural network

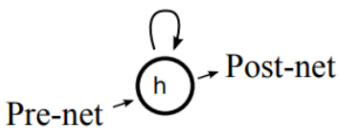
Shared Vs. Non-shared Weights



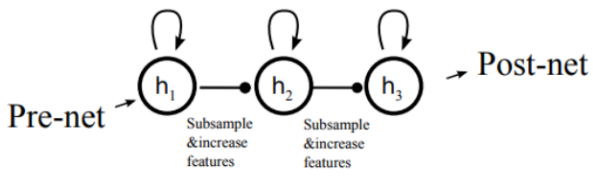
(A) ResNet without changing spacial&feature sizes



(B) ResNet with changes of spacial&feature sizes (He et. al.)



(C) Recurrent form of A



(D) Recurrent form of B

Experiment

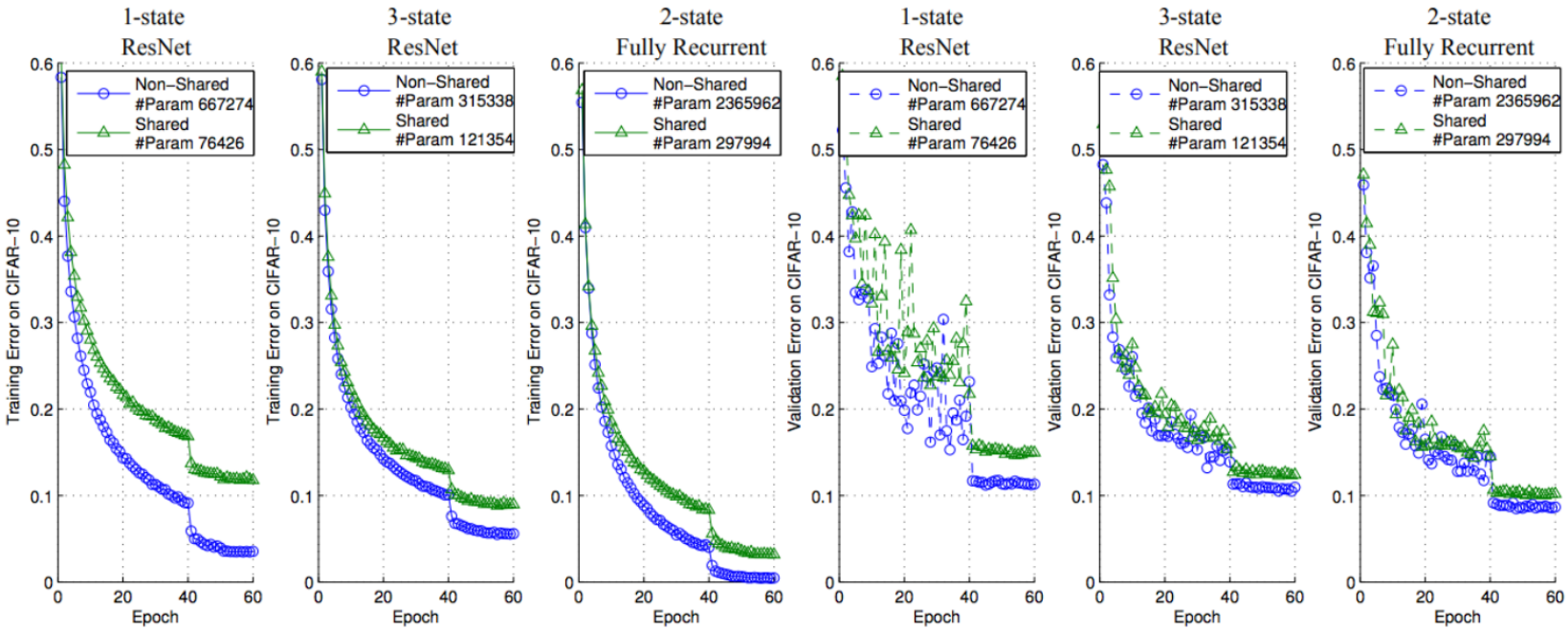


Figure 6: All models are robust to sharing weights across time. This supports our conjecture that deep networks can be well approximated by shallow/moderately-deep RNNs. The transition matrices of all models are shown in Figure 4. "#Param" denotes the number of parameters. The 1-state ResNet has a single state of size 32x32x64 (height × width × #features). It was trained and tested with readout time t=10. The 3-state ResNet has 3 states of size 32x32x16, 16x16x32 and 8x8x64 – there is a transition (via a simple convolution) at time 4 and 8 — each state has a self-transition unrolled 3 times. The 2-state fully recurrent NN has 2 states of the same size: 32x32x64. It was trained and tested with readout time t=10 (same as 1-state ResNet). It is a generalization of and directly comparable with 1-state ResNet, showing the benefit of having more states. The 2-state fully recurrent NN with shared weights and fewer parameters outperforms 1-state and 3-state ResNet with non-shared weights.

PDEs For Computer Vision

PDE is a classical method in low level computer vision.

PM(**Perona and Malik**) Equation is a traditional PDE to processing image.

$$u_t = div(g(|\nabla u|)\nabla u)$$

$g(x)$ here is always taken as $g(x) = \frac{1}{1+kx^2}$

Consider ResNet as PDE

Understanding ResNet: A PDE Perspective

Main Observe

- The similarity between the forward difference scheme and the shortcut in PDE.
- An FIR filter can be consider as a finite difference scheme of a differential operator.

Main Idea

Under this observe, we may write every iterate of ResNet as

$$u_t = F(u) = \sum_{\alpha} f_\alpha(D^\alpha u)$$

Filter And Differential Operator

Identify Differential Operator via **Vansing Moment**

Vanishing Moment

An FIR highpass filter $\mathbf{q}$ have vanishing moments of order $\alpha = (\alpha_1, \alpha_2)$, where $\alpha \in Z_+^2$ provided that

$$\sum_{k \in Z_+^2} k^\beta \mathbf{q}[k] = i^{|\beta|} \frac{\partial^\beta}{\partial \omega^\beta} \hat{\mathbf{q}}(\omega)|_{\omega=0} = 0$$

FIR filters

Theorem (FIR filters as Differential Operator)

Let $\mathbf{q}$ *be an FIR highpass filter with vanishing moments of oder* $\alpha \in Z_+^2$. *Then for a smooth function* $F(x)$ *on* R^2, *we have*

$$\frac{1}{\delta^{|\alpha|}} \sum_{k \in Z^2} \mathbf{q}[k] F(x + \epsilon k) = C_\alpha \frac{\partial^\alpha}{\partial x^\alpha} F(x) + O(\epsilon)(\epsilon \to 0)$$

Here $C_\alpha = \frac{1}{\alpha!} \sum_k k^\alpha \mathbf{q}[k]$

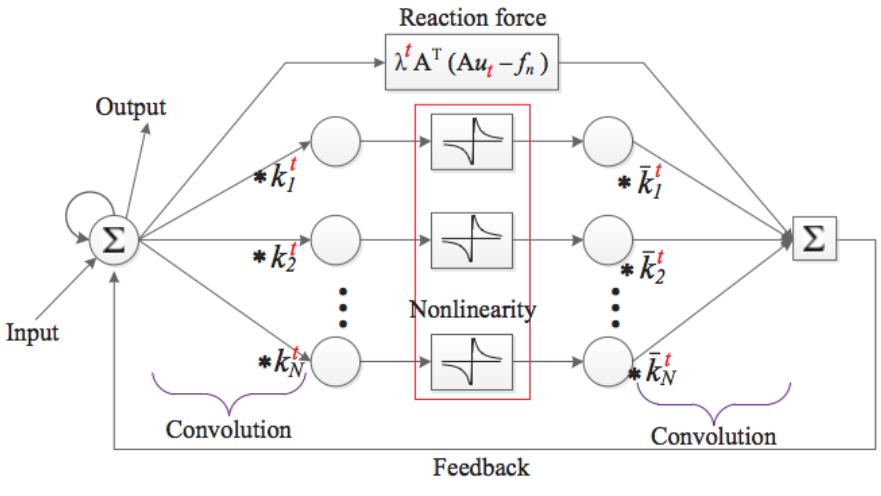
Proof.

The proof is by straightforward calculation based on Taylor's expansion. □

Learning Optimized Reaction Diffusion

Ignoring the coupled relation between $\nabla_x u$ and $\nabla_y u$, the P-M Equation can be written as:

$$\frac{u_t - u_{t-1}}{\Delta t} = -\sum_{i=1}^{N_k} (K_i^t)^T \psi_i^t (K_i^t u_{t-1}) - \phi(u_{t-1}, f_n)$$

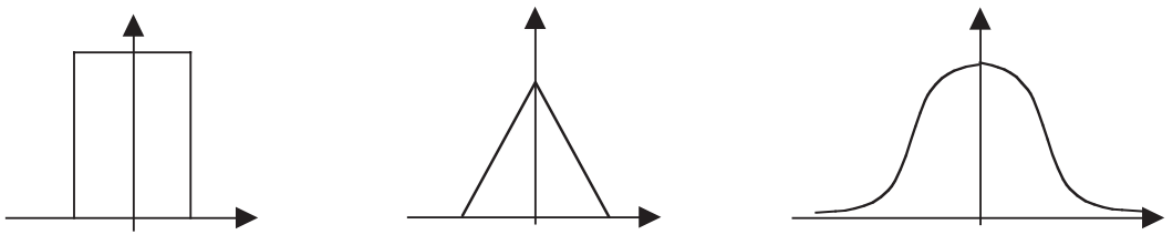


Can handler PDEs from conservation law like burgers equation:

$$u_t + div(F(u)) = 0$$

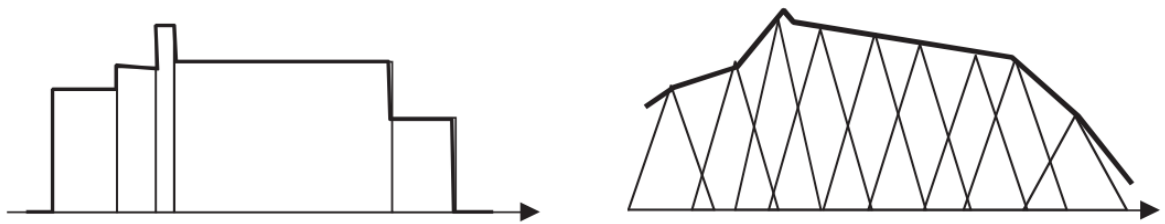
How to approximate ψ

Radial Basis Networks(RBN)! A radial basis network is a feed-forward neural network using the radial basis activation function. Some Examples are depicted in the Fig below.



1.10 Three examples of one-dimensional radial basis functions.

We can use RBN to approximate functions by $\hat{\psi}(x) = \sum_{i=1}^{c} \omega_i \phi(\frac{||x-m_i||}{\sigma_i})$!



1.11 Two examples illustrating radial basis function approximation.

Experiment

Fig. Gaussian Noise Denoising Test:(a) clean image(b) noised image (c) BM3D (d) EPLL (e) SRCNN (f) WNNM (g)5×5 filters (h) 7×7 filters (i) 5×5 filters with multi-scale learning (j)7×7 filters with multi-scale learning

Learning ResNet: A PDE Constrained Optimization Problem

The learning problem aims at estimating the parameter of the forward propagation and classifier so that the DNN accurately approximates the data label relation for the training data can be consider as

$$\min_{W,\mu,K_i,b_i,i\in 0,1,\cdots,N-1} \frac{S(h(Y_N W + e_s \mu^T), C)}{s} + \alpha R(W, \mu, K_{0,\cdots,N-1}, b_{0,\cdots,N-1})$$

$$subject.to. Y_{j+1} = Y_j + h\sigma(Y_j K_j + b_j), j = 0, 1, \cdots, N-1$$

Here S is the covex loss function, the regularizer R penalizes undersirable parameters.

Block coordinate descent

1. Weight (W, μ)
2. Forward propagation Network

Updating weight using Newton preconditioned conjugate gradient(PCG) method. The weight of the forward propagation are updated using a Gauss-Newton-PCG method.

Stability and well posedness of the forward propagation

It is well known that any parameter estimation problem requires a well-posed forward problem, i.e. a problem whose output is continuous with respect to its inputs. The forward network can be consider as an explicit Euler discretization of the nonlinear ODE

$$\dot{y}(t) = \sigma(K^T(t)y(t) + b(t)), y(0) = y_0$$

The ODE is stable if

$$\max(Re(\lambda(J(t)))) \leq 0, \forall t \in [0, T]$$

Here $J(t)$ is the Jacobian of the right hand side, i.e.

$$\begin{aligned} J(t) &= \left(\nabla_y(\sigma(K(t)^T y + b(t)))\right)^T \\ &= diag(\sigma'(K(t)^T y + b(t)))K(t)^T \end{aligned}$$

Stability and well posedness of the forward propagation

$$\max(Re(\lambda(J(t)))) \leq 0, \forall t \in [0, T]$$

Where

$$\begin{aligned} J(t) &= \left(\nabla_y(\sigma(K(t)^T y + b(t)))\right)^T \\ &= diag(\sigma'(K(t)^T y + b(t)))K(t)^T \end{aligned}$$

Since the activation function σ is typically monotonically non-decreasing, i.e. $\sigma'(\cdot) \geq 0$, then we only need

$$\max(Re(\lambda(K(t)))) \leq 0, \forall t \in [0, T]$$

Overall forward propagation stabiltiy

Theorem To ensure the stability of the overall forward propagation, we require

$$|1 + h \max(\lambda(K_j))| \leq 1, \forall j = 0, 1, \cdots, N-1$$

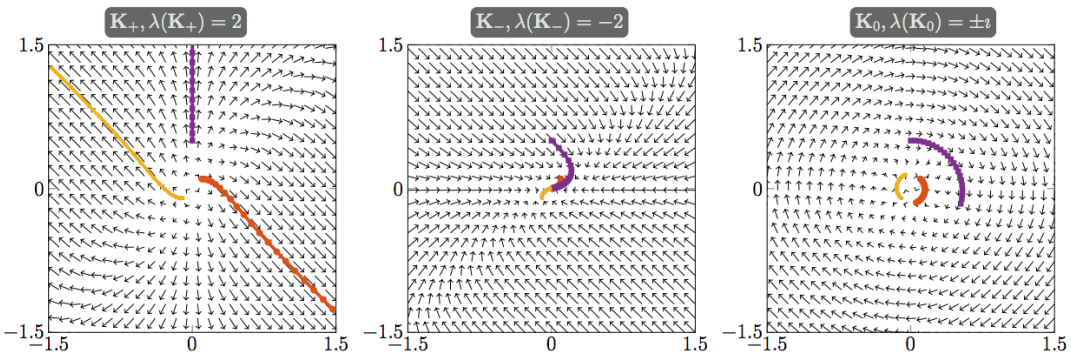


Figure 1: Phase plane diagrams for ResNets with $N = 10$ identical layers parameterized by the weight matrices in (3.11) starting with three different features. Colored lines indicate the values at hidden layers and arrows depict the force field. Left: Due to its positive eigenvalues, the features diverge using $\mathbf{K}_+$ leading to an unstable forward propagation. Center: $\mathbf{K}_-$ yields a contraction that annihilates differences in the features and renders the learning problem ill-posed. Right: The anti-symmetric matrix $\mathbf{K}_0$ leads to rotations, preserves distances between the features, and yields well-posed forward propagation and learning.

$$\text{we need:} Re(\lambda(K(t))) \approx 0$$

Stable Forward Propagation

Now we enforce Jacobians whose eignvalues have very small real parts. The simplest way is to enforce the Jacobians are anti-symmetric!

$$Y_{j+1} = Y_j + h\sigma(\frac{1}{2}Y_j(K_j - K_j^T) + b_j)$$

Another way is the recast forward propagation as a Hamiltonian system, which as structure:

$$\dot{y}(t) = -\nabla_z H(y, z, t), \dot{z}(t) = \nabla_y H(y, z, t), \forall t \in [0, T]$$

If H has the formula $H(y, z) = \frac{1}{2}z^T z + f(y)$, then the Hamiltonian system can be written as

$$\ddot{y}(t) = \nabla_y f(y(t)), \forall t \in [0, T]$$

Numerical Scheme

- **Verlet Method**

$$y_{j+1} = 2y_j - y_{j-1} + h^2\sigma(K_j y_j + b_j)$$

- **Leapforg**

$$z_{j+\frac{1}{2}} = z_{j-\frac{1}{2}} - h\sigma(K_j^T y_j + b_j)$$
$$y_{j+1} = y_j + h\sigma(K_j z_{j+\frac{1}{2}} + b_j)$$

- **Speed Verlet**

$$r_{n+1} = r_n + v_n\Delta t + \frac{1}{2}\frac{F_n}{m}\Delta t^2$$
$$v_{n+1} = v_n + \frac{1}{2}(\frac{F_{n+1}}{m} + \frac{F_n}{m})\Delta t$$

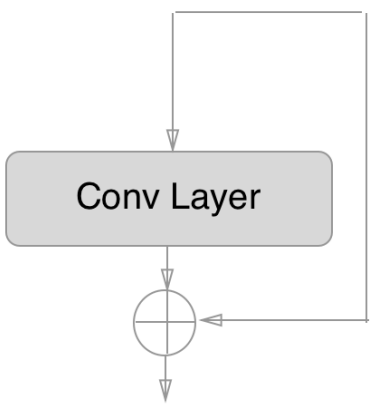
Among the three approaches, the Verlet method is the most flexible as it can handle non-square weighting matrices and does not require additional constraints on K

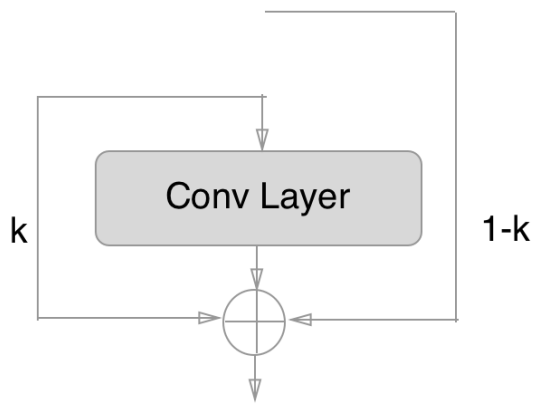
PDE And Network.

Primal Equation: $u_t = f(u)$

Accelerated Equation: $u_{tt} + cu_t = f(u)$

Turns into Network:





Related Works

DenseNet: For each layer, the feature maps of all preceding layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers.

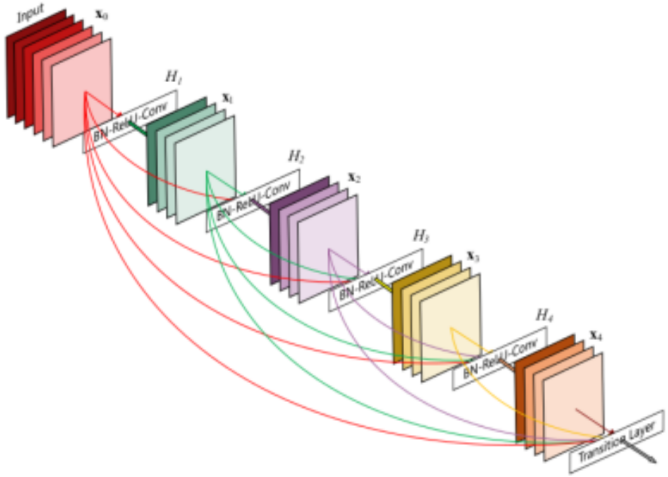
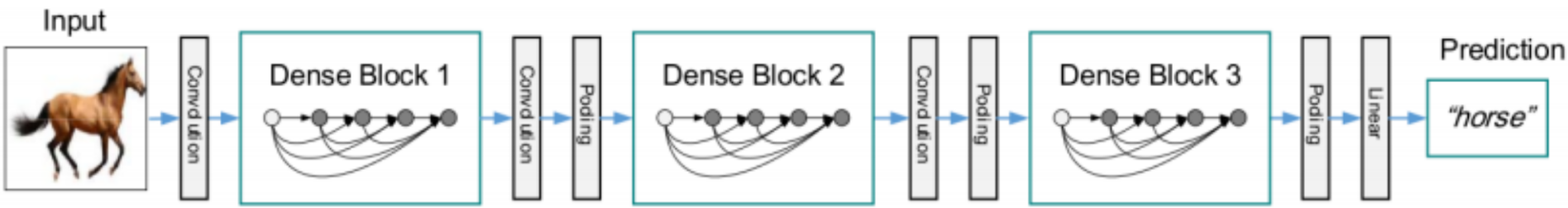


Figure 1. A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.



DenseNet

Never combine features through summation before they are passed into a layer, instead we provide them all as separate inputs.

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. No that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

DenseNet

Prior work has shown that there is **great redundancy within the feature maps of the individual layers in ResNets**. In DenseNets, all layers have direct access to every feature map from all preceding layers, which means that there is no need to re-learn redundant feature maps. Consequently, DenseNet layers are very narrow (on the order of 12 feature maps per layer) andonly add a small set of feature maps to the 'collective knowledge' of the whole network.

Multilevel learning

Goal: Transfer the fine scale convolution stencil into the coarse one–**multigrid**

- Use the parital differential equation representation on every possible mesh.
  Rediscretization:useful if the images are sufficient smooth.
- Algebraic multigrid approach.

Multilevel learning: Algebraic multigrid approach.

Assume the following connection holds between the fine mesh image y_h and coarse mesh image y_H

$$y_H = Ry_h, \hat{y}_h = Py_H, RP = I$$

P is a prolongation matrix and R is a restriction matrix. $\hat{y}_h$ is an interpolated coarse scale image to the fine mesh.
It is common to have P as a linear interpolation and R as a full weight restriction. For uniform nodal meshs, it is typical to have

$$P = \gamma R^T, \gamma = 2^d$$

Multilevel learning: Algebraic multigrid approach.

Using the prolongation and restriction we obtain that

$$K_H y_H = R K_h P y_H$$

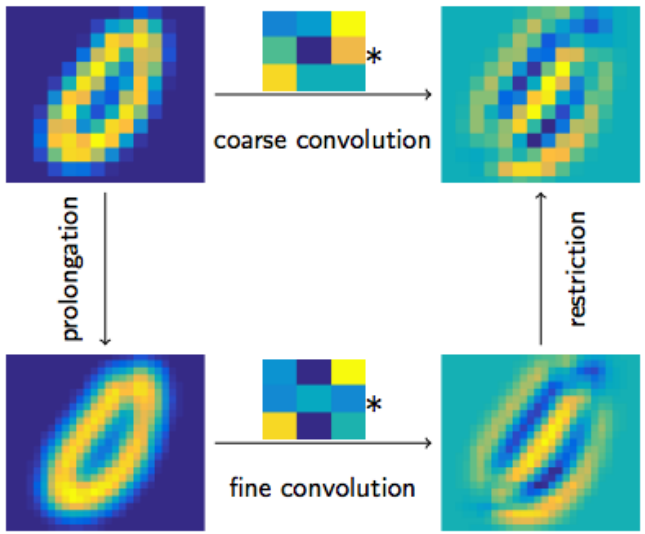


Figure 1: Illustration of a fine mesh vs. coarse mesh convolution.

Issues on learning deep models

- **Representation** ability
  - Ability of model to fit training data, if optimum could be found
  - If model A's solution space is a superset of B's, A should be better.
- **Optimization** ability
  - Feasibility of finding an optimum
  - Not all models are equally easy to optimize
- **Generalization** ability
  - Once training data is fit, how good is the test performance

How do ResNets address these issues?

- **Representation** ability
  - No explicit advantage on representation (only re-parameterization), but
  - Allow models to go deeper
- **Optimization** ability
  - Enable very smooth forward/backward prop
  - Greatly ease optimizing deeper models
- **Generalization** ability
  - Not explicitly address generalization, but
  - Deeper+thinner is good generalization

Understanding Deep Learning Requires Rethinking Generalization

Randomization Test

Deep Neural Networks easily fit random labels.

Example

There exists a two-layer neural network with ReLU activations and 2n+d weights that can represent any function on a sample of size n in d dimensions.

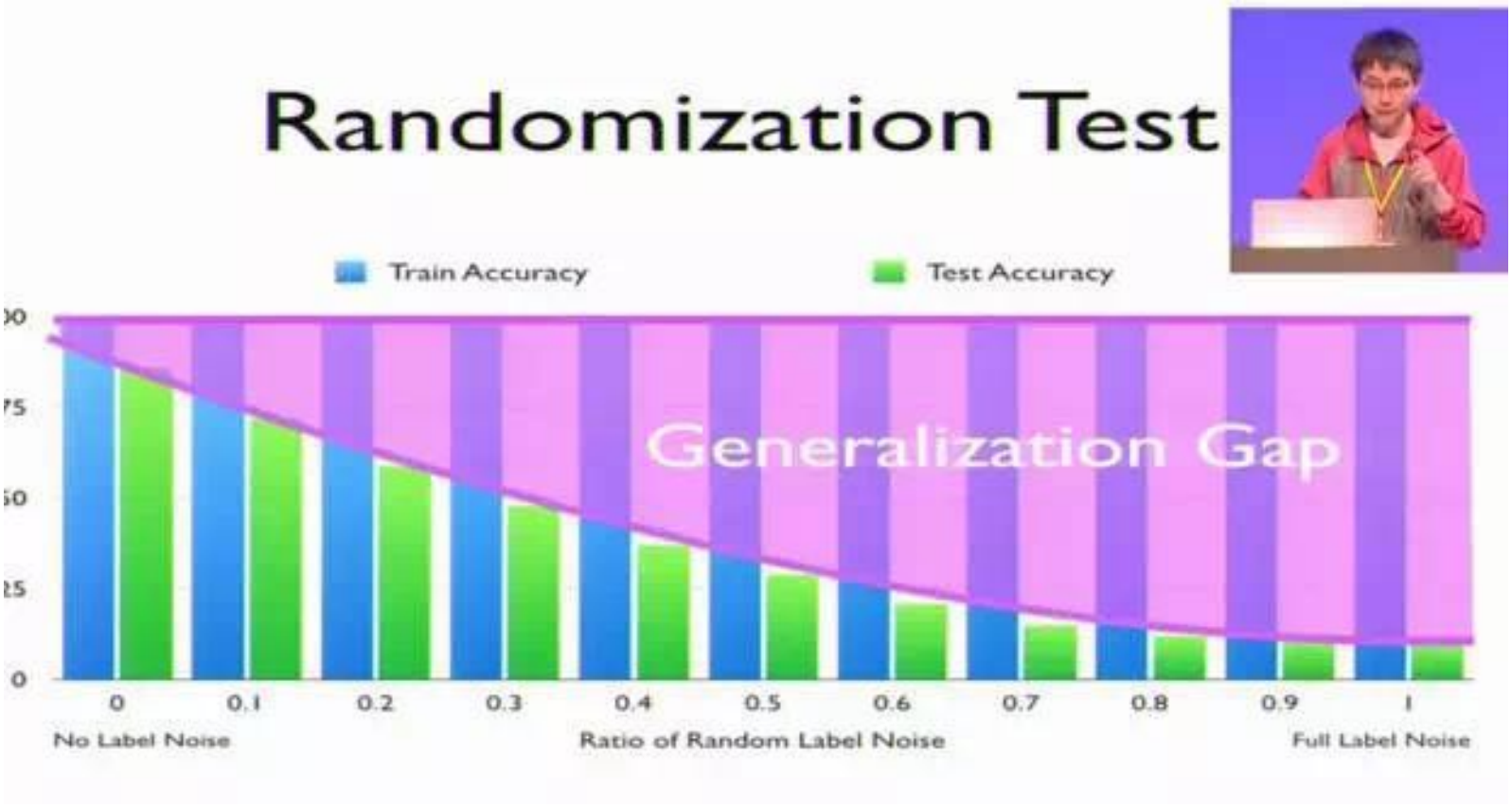
Proof.

For weight vectors ω, b, consider function$c : R^n \to R$:

$$c(x) = \sum_{j=1} \omega_j \max\{\langle a, x\rangle - b_j, 0\}$$

let $x_i = \langle a, z_i\rangle$, $b_1 < x_1 < b_2 < x_2 < \cdots < b_n < x_n$, then
$c(z_i) = A\omega, A = [\max\{x_i - b_j, 0\}]_{i,j}$ □

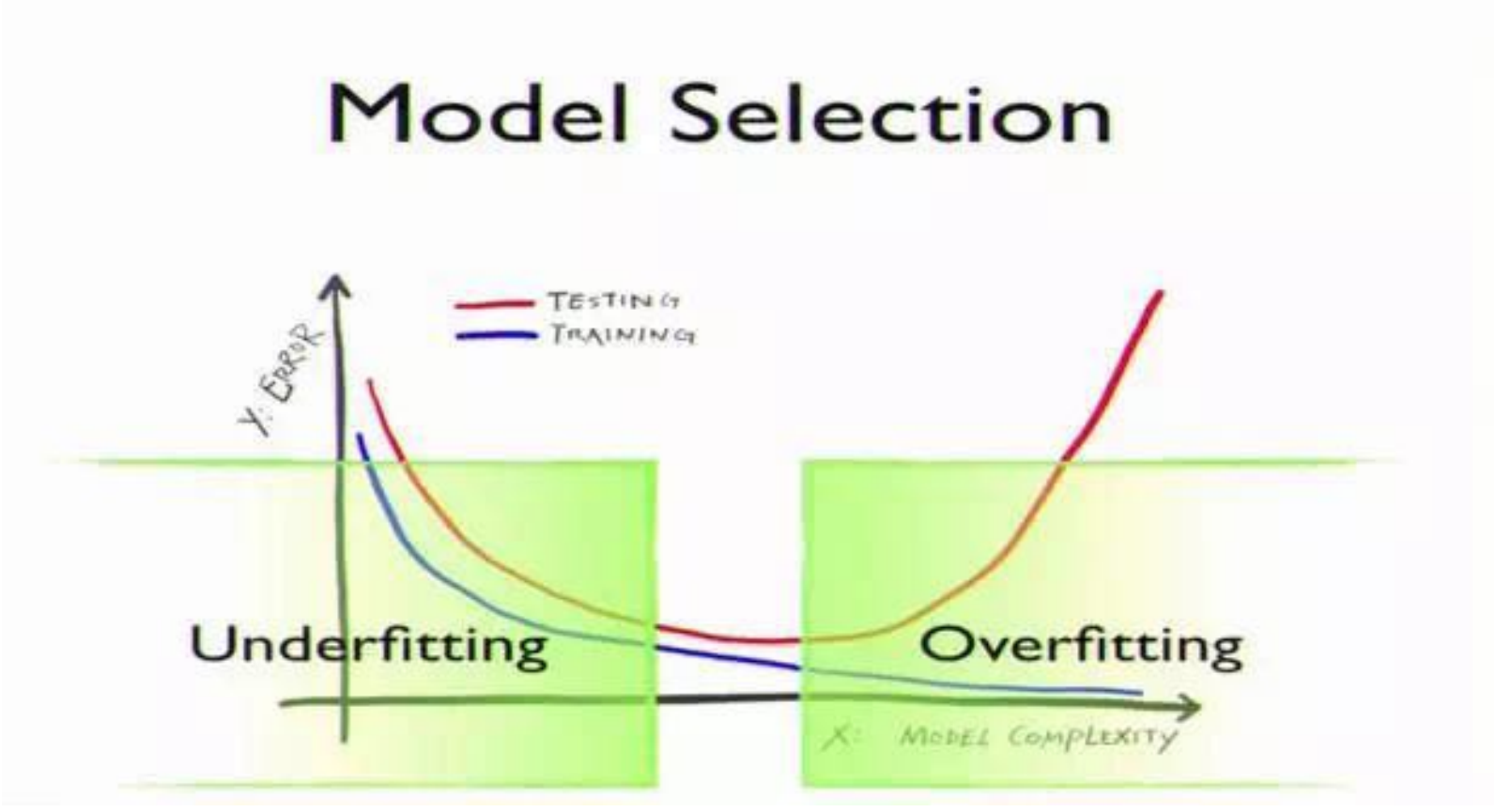
Randomization Test



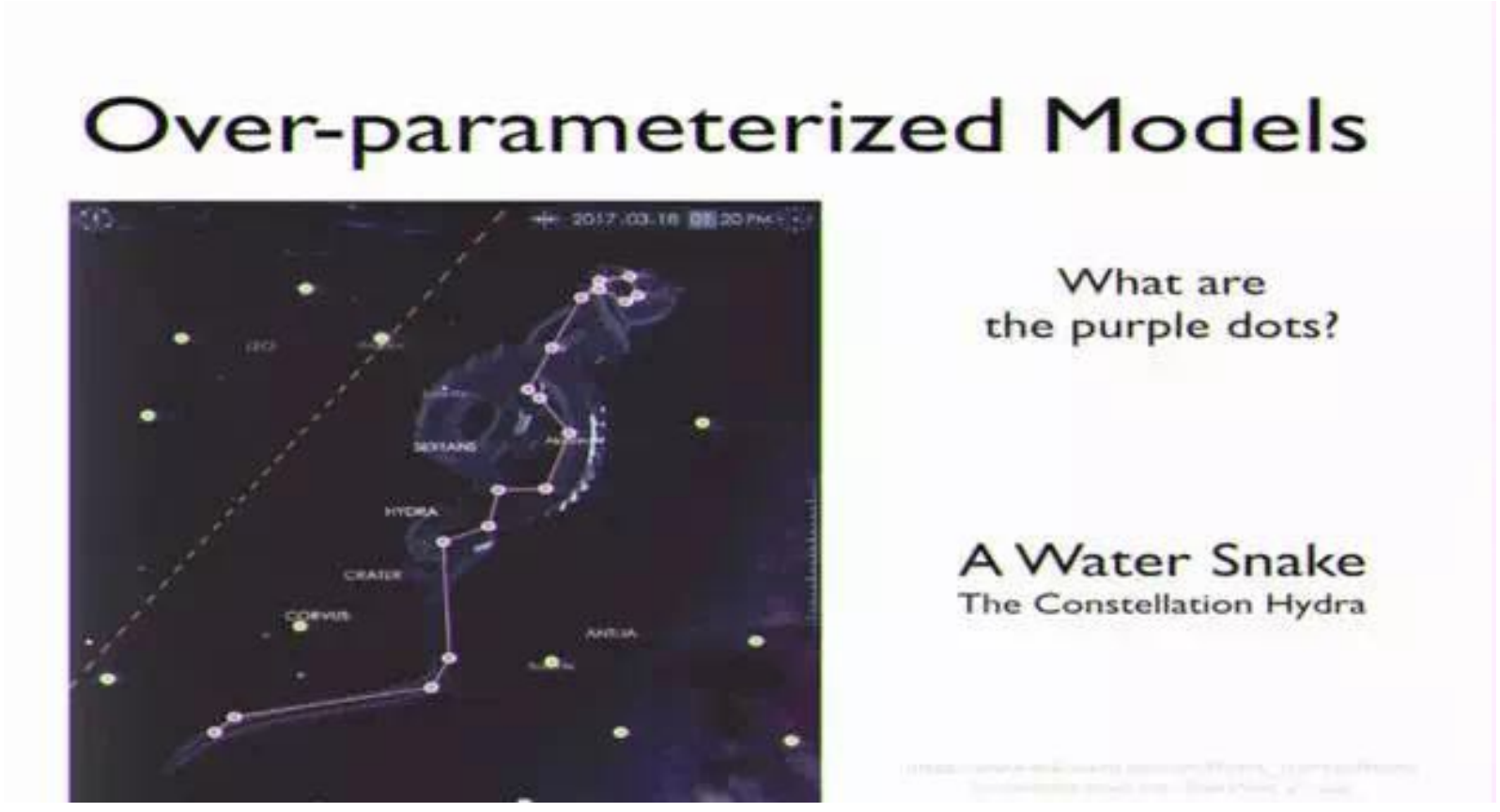
Randomization Test

- The effective capacity of neural networks is sufficient for memorizing the entire data set.
- Even optimization on random labels remains easy. In fact, training time increases only by a small constant factor compared with training on the true labels.
- Randomizing labels is solely a data transformation, leaving all other properties of the learning problem unchanged.

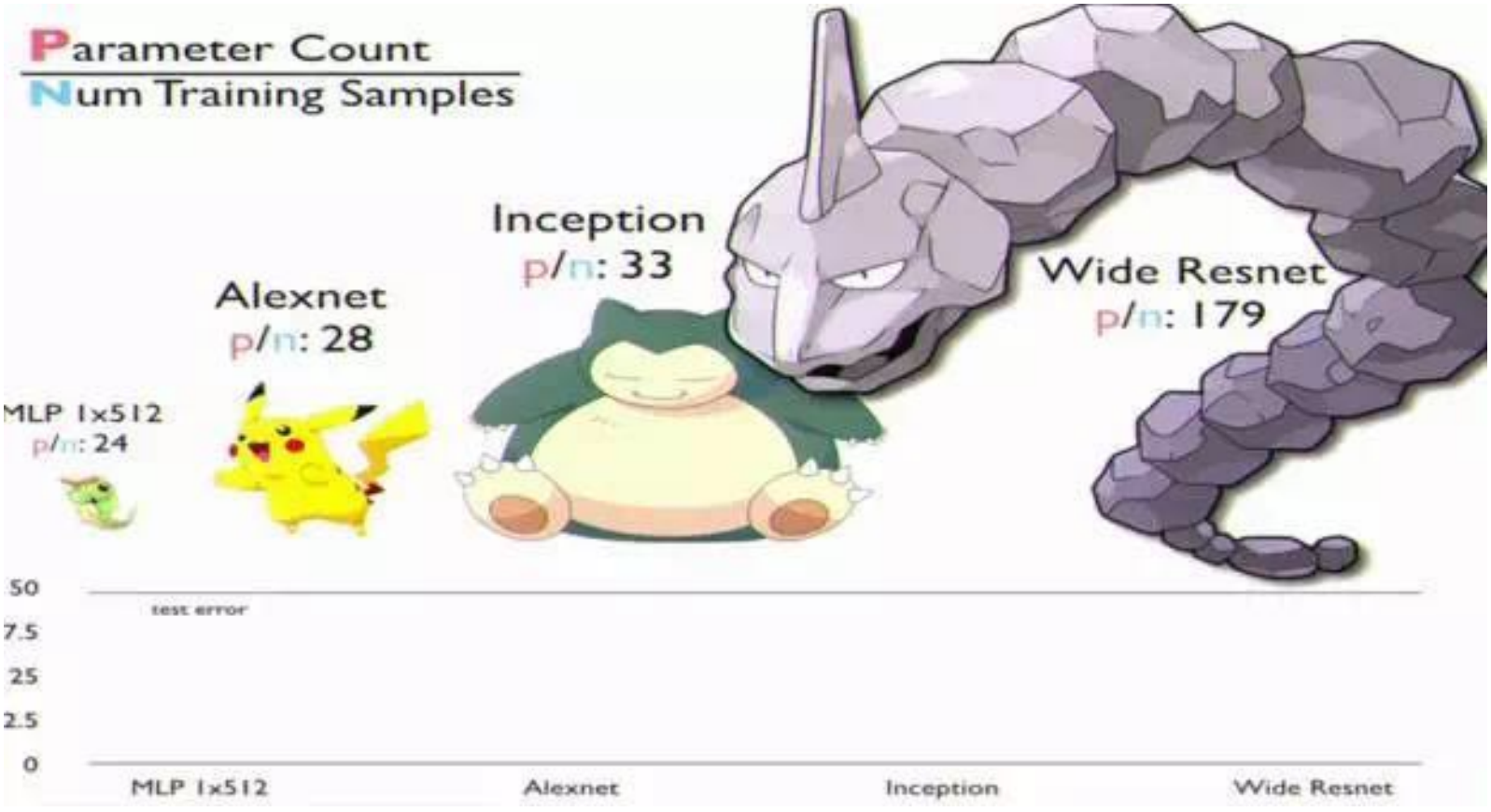
Bias-Variance Trade



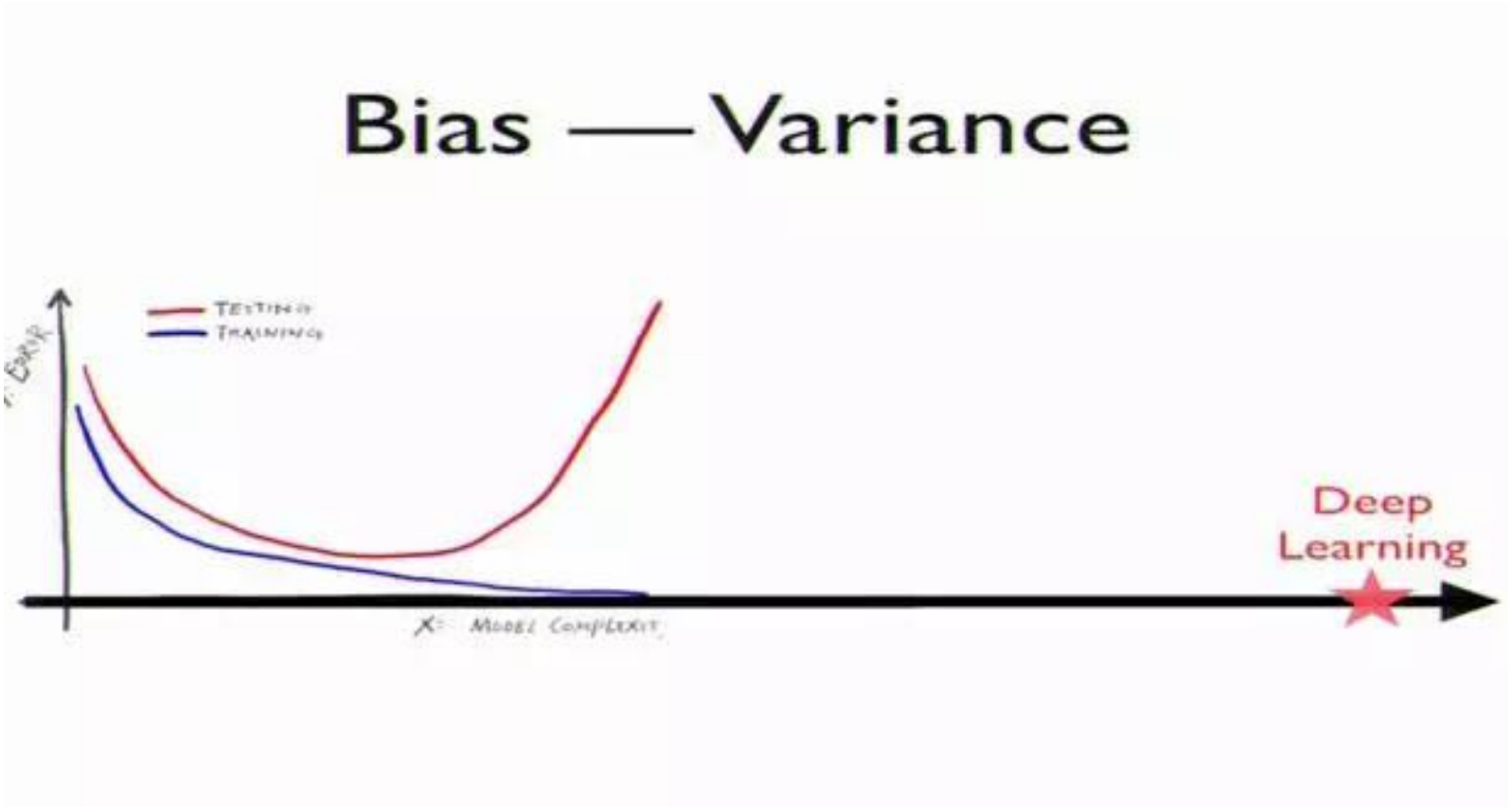
Over-parameterized



Over-parameterized



Where is Deep Learning?



Regularize

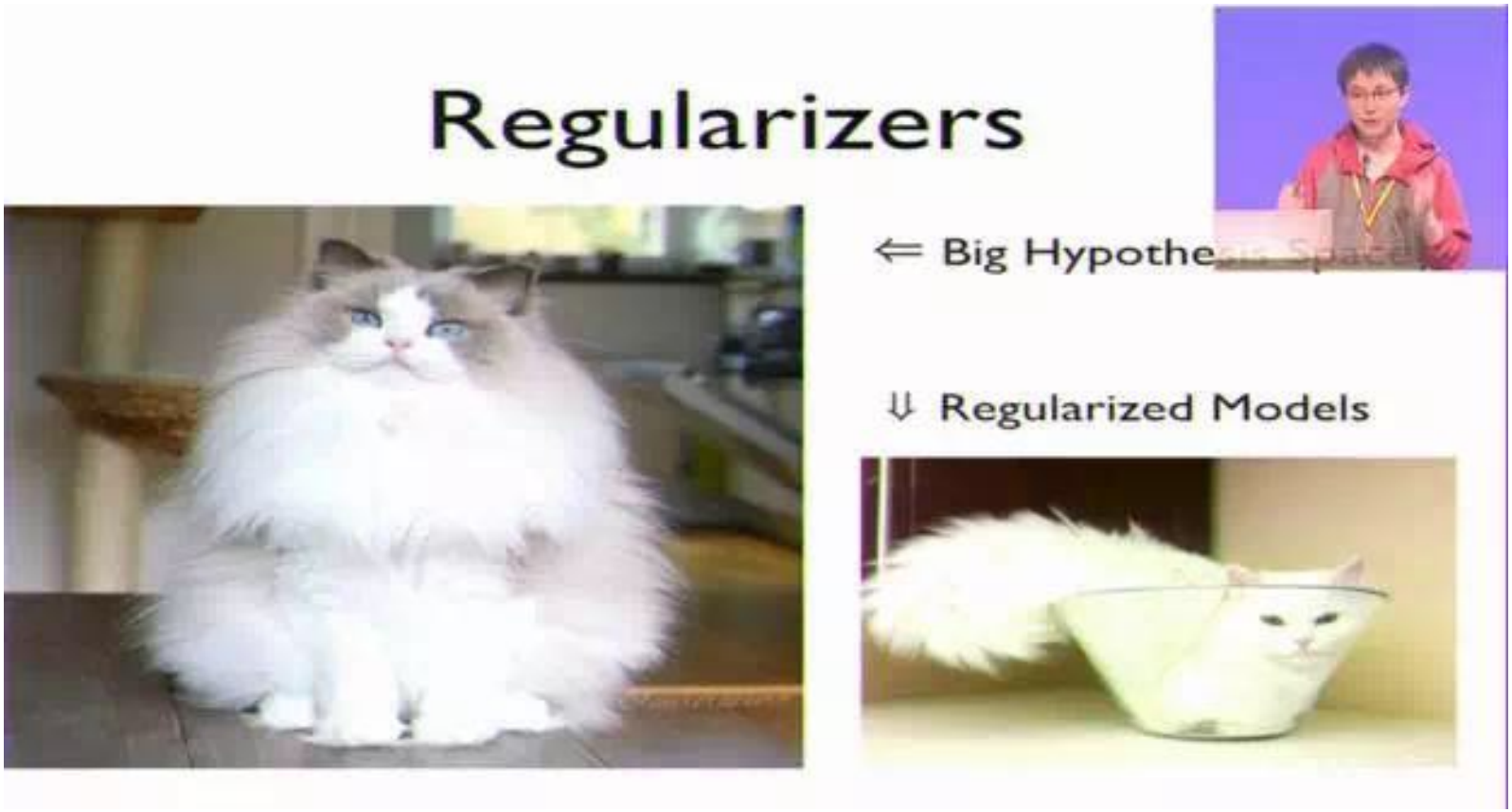
- **Data augmentation:** domain-specific transformation
- **Weight decay:** $l2-$regularizer on weights
- **Dropout*:**randomly mask out responses(by LeCun)

Result:

No change on training dataset, except Alexnet test on cifar-10.

Traditional Regularize Understanding

Constrain the model or **empower the data**



Implicit Regularization

A regularizer is anything that **hurt the training prcoess**

SGD fits Random labels:

Optimization is esay for deep learning.

References

K He,X Zhang,S Ren,J Sun (2015,MSRA)

Deep Residual Learning for Image Recognition

CVPR 2016

K He,X Zhang,S Ren,J Sun (2015,MSRA)

Identity Mappings in Deep Residual Networks

CVPR 2016

RK Srivastava,K Greff,J Schmidhuber (2015)

Highway Networks

Computer Science 2015

Y. H. Hu and J.-N. Hwang.

Handbook of neural network signal processing.

CRC press 2010

References

Bin Dong, Qingtang Jiang and Zuowei Shen

Image restoration: wavelet frame shrinkage, nonlinear evolution PDEs, and beyond

Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal ,15(1),606-660,2017.

Yunjin Chen, Wei Yu and Thomas Pock

On learning optimized reaction diffusion processes for effective image restoration

CVPR 2015

Qianli Liao, Tomaso Poggio

Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex

unknown

G Huang,Z Liu, KQ Weinberger

Densely Connected Convolutional Networks

*ICML*2010

References

M Abdi, S Nahavandi

Multi-Residual Networks: Improving the Speed and Accuracy of Residual Networks

Unknown 2017.

G Larsson, M Maire, G Shakhnarovich

FractalNet:Ultra-Deep Neural Networks without Residuals

Unknown 2017

G Huang, Z Liu, KQ Weinberger

Densely Connected Convolutional Networks

Unknown 2017

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, Kilian Weinberger

Deep Networks with Stochastic Depth

ECCV 2016

Kaiming He,Xiangyu Zhang,Shaoqing Ren,Jian Sun

Delving Deep into Rectifiers:Surpassing Human-Level Performance on ImageNet Classification

ICCV 2015

References

E Haber, L Ruthotto, E Holtham

Learning across scales - A multiscale method for Convolution Neural Networks

Unpublished

E Haber, L Ruthotto

Stable Architectures for Deep Neural Networks

Unpublished

The End