Lecture 9

CMSC 35246: Deep Learning

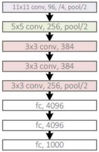
Shubhendu Trivedi
&
Risi Kondor

University of Chicago

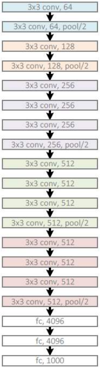
April 24, 2017

Architectures from before

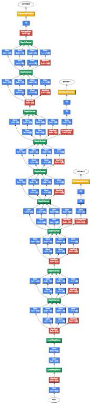
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

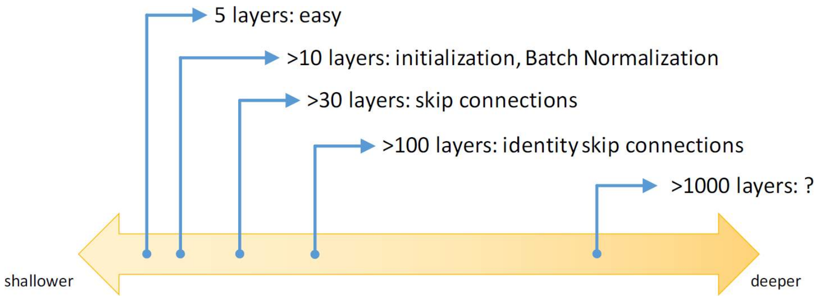


Depth is clearly a significant factor for superior performance

Depth is clearly a significant factor for superior performance

Is learning better networks just about stacking more layers?

Architectures from before



Degradation Problem

- Adding more layers leads to a Degradation Problem

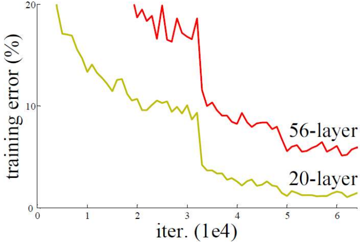
Degradation Problem

- Adding more layers leads to a Degradation Problem
- Increasing depth: Accuracy first saturates, then rapidly degrades

Degradation Problem

- Adding more layers leads to a Degradation Problem
- Increasing depth: Accuracy first saturates, then rapidly degrades
- Degradation is *not caused due to overfitting*
- On adding more layers after a certain depth *training error increases with depth*

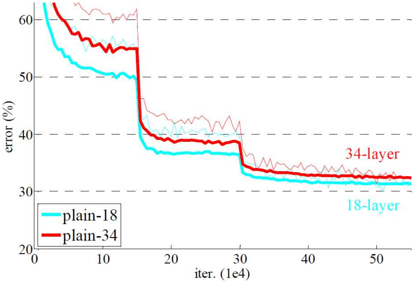
Degradation Problem



- Networks obtained by stacking 3x3 convolutional layers on CIFAR-10

Figure: He *et al.* Deep Residual Learning for Image Recognition, CVPR 2016

Degradation Problem



- Networks obtained by stacking 3x3 convolutional layers on ImageNet 1000

Figure: He *et al.* Deep Residual Learning for Image Recognition, CVPR 2016

A Solution by Construction

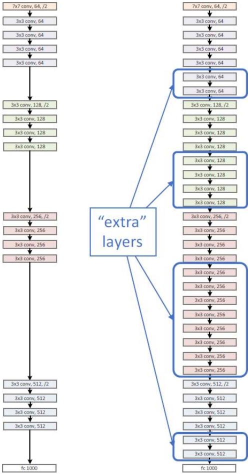
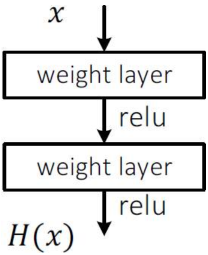


Figure: He *et al.* Deep Residual Learning for Image Recognition, CVPR 2016

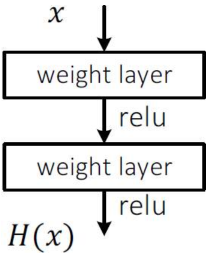
A deeper model should not have higher training error

A Plain Network Block



- Let $\mathcal{H}(\mathbf{x})$ be the function to be fit by a few stacked layers

A Plain Network Block



- Let $\mathcal{H}(\mathbf{x})$ be the function to be fit by a few stacked layers
- Above, we hope that the two layers will fit $\mathcal{H}(\mathbf{x})$

Residual Learning

- If stack can approximate $\mathcal{H}(\mathbf{x})$, then it can approximate $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$

Residual Learning

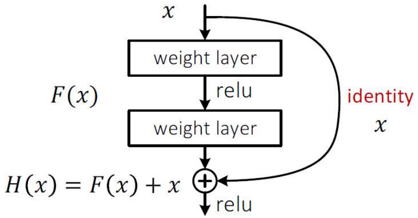
- If stack can approximate $\mathcal{H}(\mathbf{x})$, then it can approximate $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$
- $\mathcal{H}(\mathbf{x}) - \mathbf{x}$ is a residual function ($\mathbf{x}$ and $\mathcal{H}(\mathbf{x})$ of same size)

Residual Learning

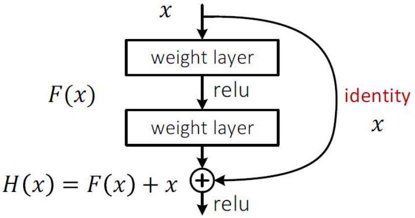
- If stack can approximate $\mathcal{H}(\mathbf{x})$, then it can approximate $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$
- $\mathcal{H}(\mathbf{x}) - \mathbf{x}$ is a residual function ($\mathbf{x}$ and $\mathcal{H}(\mathbf{x})$ of same size)
- $\mathcal{F}(\mathbf{x})$ is a residual map with respect to the identity

Residual Learning

- If stack can approximate $\mathcal{H}(\mathbf{x})$, then it can approximate $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$
- $\mathcal{H}(\mathbf{x}) - \mathbf{x}$ is a residual function ($\mathbf{x}$ and $\mathcal{H}(\mathbf{x})$ of same size)
- $\mathcal{F}(\mathbf{x})$ is a residual map with respect to the identity

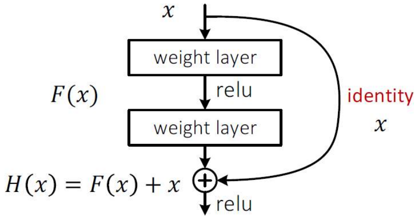


Residual Learning



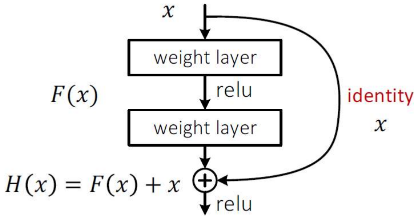
- If identity map is optimal $\implies$ drive weights to zero to approach identity

Residual Learning



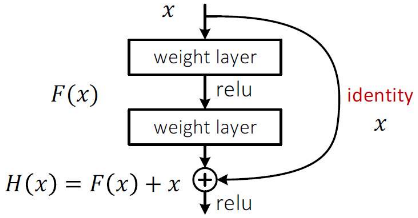
- If identity map is optimal $\implies$ drive weights to zero to approach identity
- Identity is rarely optimal but it serves to pre-condition the problem (e.g. similar work in multigrid literature)

Residual Learning



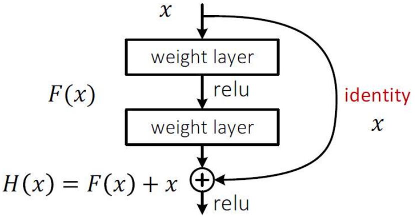
- If identity map is optimal $\implies$ drive weights to zero to approach identity
- Identity is rarely optimal but it serves to pre-condition the problem (e.g. similar work in multigrid literature)
- If the optimal map is *closer* to identity than a zero map, easier to find small perturbations w.r.t identity

Residual Learning



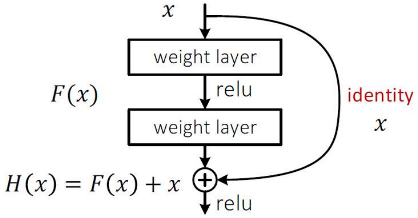
- Here $\mathcal{F}(\mathbf{x}) = W_2 \max\{0, W_1\mathbf{x}\}$

Residual Learning



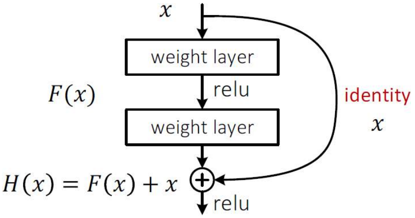
- Here $\mathcal{F}(\mathbf{x}) = W_2 \max\{0, W_1\mathbf{x}\}$
- $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ is implemented as a shortcut and elementwise addition

Residual Learning



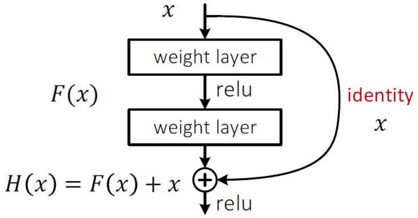
- Here $\mathcal{F}(\mathbf{x}) = W_2 \max\{0, W_1\mathbf{x}\}$
- $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ is implemented as a shortcut and elementwise addition
- Dimensions of $\mathcal{F}(\mathbf{x})$ and $\mathbf{x}$ must be equal

Residual Learning



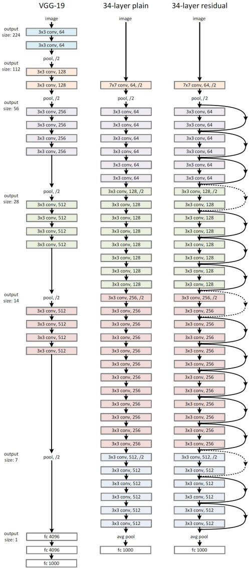
- Here $\mathcal{F}(\mathbf{x}) = W_2 \max\{0, W_1\mathbf{x}\}$
- $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ is implemented as a shortcut and elementwise addition
- Dimensions of $\mathcal{F}(\mathbf{x})$ and $\mathbf{x}$ must be equal
- If not: Perform linear projection $W_s\mathbf{x}$

Residual Learning



- Here $\mathcal{F}(\mathbf{x}) = W_2 \max\{0, W_1\mathbf{x}\}$
- $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ is implemented as a shortcut and elementwise addition
- Dimensions of $\mathcal{F}(\mathbf{x})$ and $\mathbf{x}$ must be equal
- If not: Perform linear projection $W_s\mathbf{x}$
- Aside: Can also use a square matrix W_s even if dimensions are equal, but an identity map is found to be better

First Attempt: VGG Type Network



First Attempt: VGG Type Network

- Stacked network of before but with residual connections

First Attempt: VGG Type Network

- Stacked network of before but with residual connections
- Training Procedure:
  - Both networks are trained from scratch

First Attempt: VGG Type Network

- Stacked network of before but with residual connections
- Training Procedure:
  - Both networks are trained from scratch
  - No dropout is used

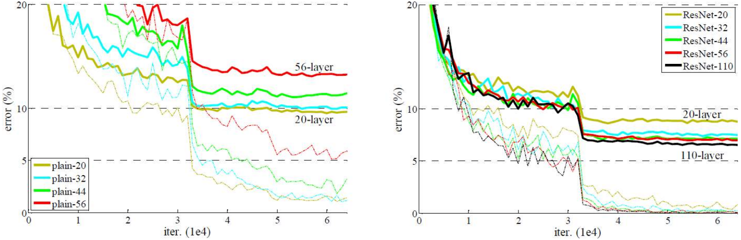
First Attempt: VGG Type Network

- Stacked network of before but with residual connections
- Training Procedure:
  - Both networks are trained from scratch
  - No dropout is used
  - Batch-normalization after every layer

First Attempt: VGG Type Network

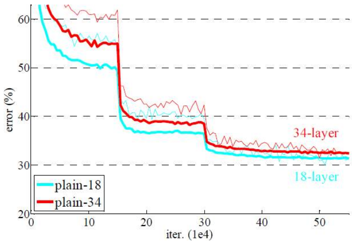
- Stacked network of before but with residual connections
- Training Procedure:
  - Both networks are trained from scratch
  - No dropout is used
  - Batch-normalization after every layer
  - Use similar data augmentation for both

CIFAR-10



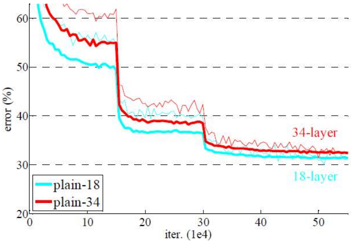
- For now focus on 32 layer results for both

ImageNet with ResNet



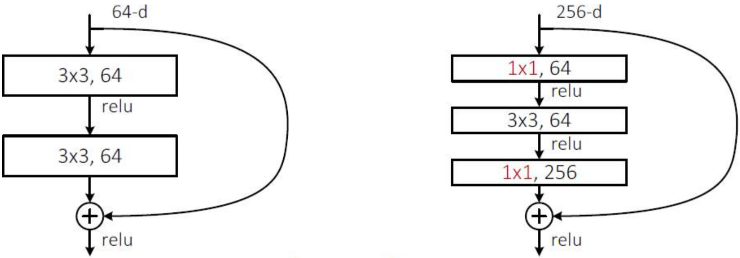
- Thin curves: Training Error; Thick curves: Validation Error

ImageNet with ResNet



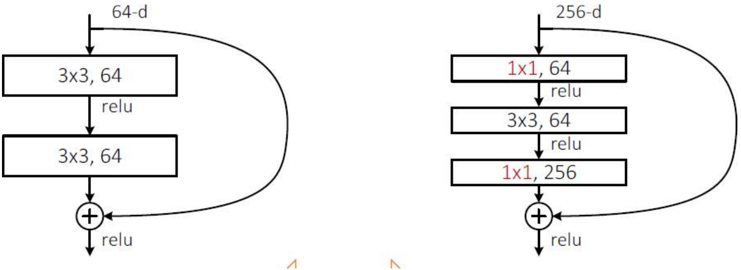
- Thin curves: Training Error; Thick curves: Validation Error
- Deep ResNets have lower training and validation error

Bottleneck Residual Block



- 1x1 convolutions to reduce and increase dimensionality

Bottleneck Residual Block

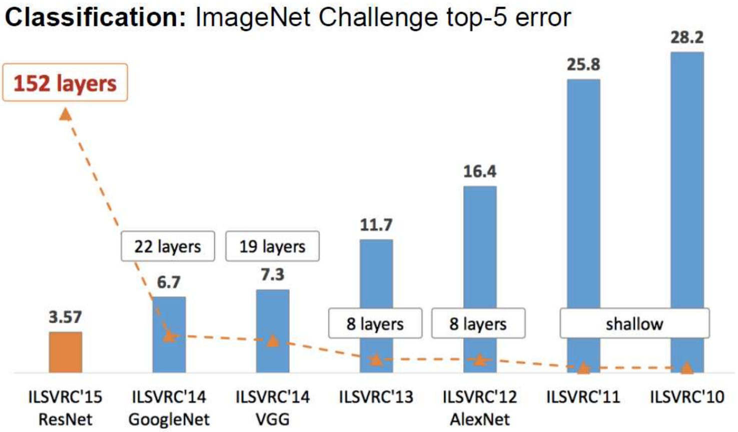


- 1x1 convolutions to reduce and increase dimensionality
- Use parameter free identity shortcuts

Results with Deeper ResNets: CIFAR-10

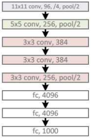
method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Results with Deeper ResNets: ImageNet

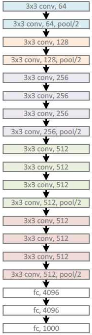


Revolution of Depth

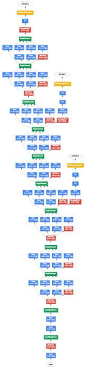
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)



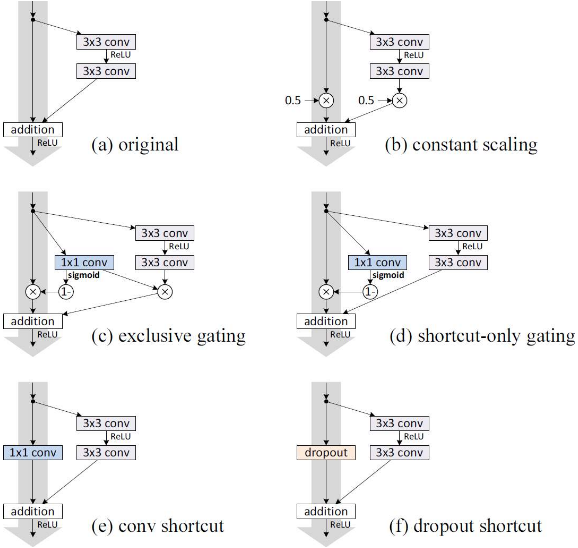
Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Types of Shortcut Connections

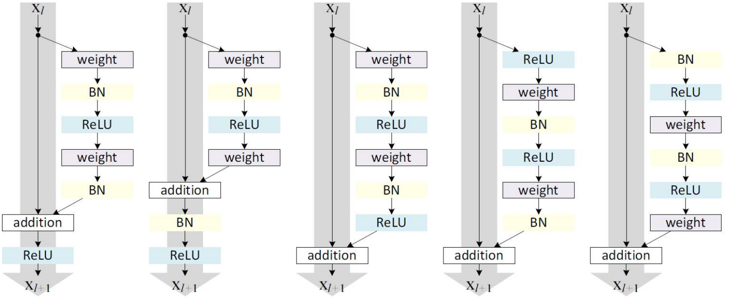


Types of Shortcut Connections

case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1 - g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1 - g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1 - g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1 - g(\mathbf{x})$	1	12.86	init b_g=0
		$1 - g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

- Results on CIFAR-10 test-set using ResNet-100. Fail represents error more than 20%

Types of Activations



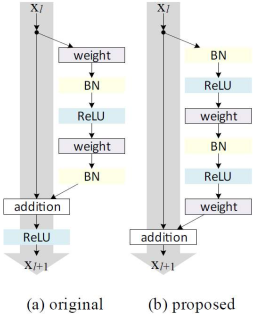
(a) original | (b) BN after addition | (c) ReLU before addition | (d) ReLU-only pre-activation | (e) **full pre-activation**

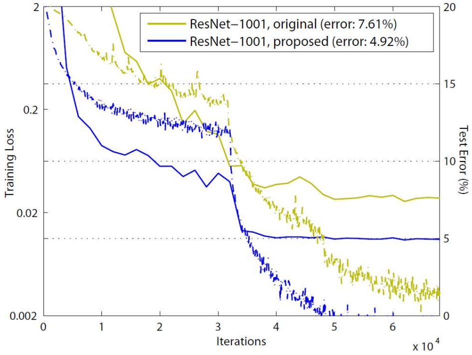
Types of Activations

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

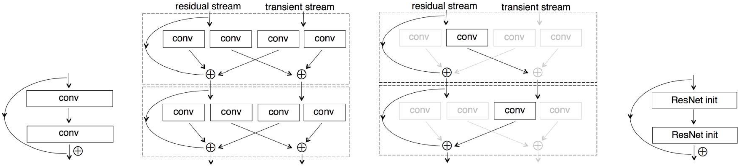
- Results on CIFAR-10 test-set.

A Better Residual Unit



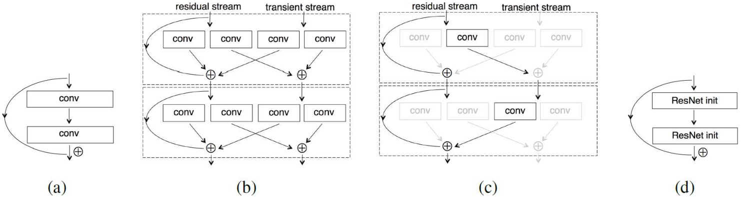


ResNet in ResNet



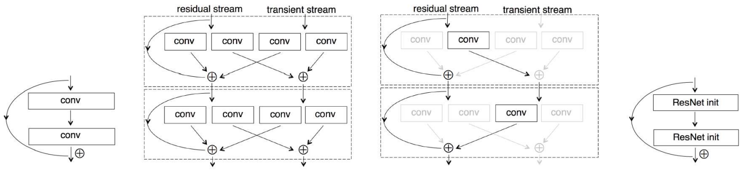
(a) (b) (c) (d)

ResNet in ResNet



- Modular unit is a *generalized* residual block with two parallel states:
  - A residual stream $\mathbf{r}$ with identity shortcuts like in original ResNets (parameters $W_{l,r\to r}$)

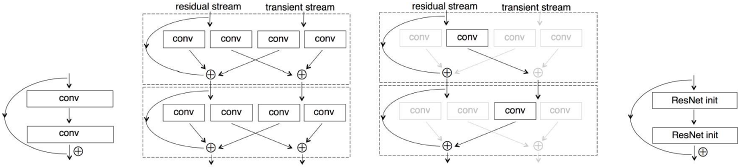
ResNet in ResNet



(a) (b) (c) (d)

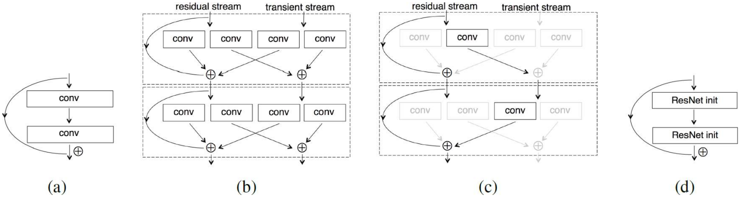
- Modular unit is a *generalized* residual block with two parallel states:
    - A residual stream $\mathbf{r}$ with identity shortcuts like in original ResNets (parameters $W_{l,r \to r}$)
    - A transient stream $\mathbf{t}$, a standard convolution layer (parameters $W_{l,t \to t}$)

ResNet in ResNet



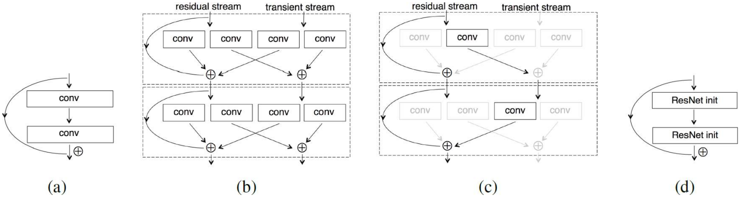
(a) (b) (c) (d)

ResNet in ResNet



- Two additional sets of conv. filters ($W_{l,r \to t}$, $W_{l,t \to r}$) in each block are used for cross-stream info. transfer

ResNet in ResNet



- Two additional sets of conv. filters ($W_{l,r\to t}$, $W_{l,t\to r}$) in each block are used for cross-stream info. transfer
- Transient stream $\mathbf{t}$ allows to process information from either stream without shortcuts (allowing information to be discarded)

Highway Networks

- In ResNets we had

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l) + \mathbf{x}_l$$

Highway Networks

- In ResNets we had

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l) + \mathbf{x}_l$$

- In Highway Networks:

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l\mathcal{C}(\mathbf{W_C}, \mathbf{x}_l)$$

Highway Networks

- In ResNets we had

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l) + \mathbf{x}_l$$

- In Highway Networks:

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l\mathcal{C}(\mathbf{W_C}, \mathbf{x}_l)$$

- $\mathcal{T}$ is the Transfer Gate, $\mathcal{C}$ is the Carry Gate

Highway Networks

- In ResNets we had

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l) + \mathbf{x}_l$$

- In Highway Networks:

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l\mathcal{C}(\mathbf{W_C}, \mathbf{x}_l)$$

- $\mathcal{T}$ is the Transfer Gate, $\mathcal{C}$ is the Carry Gate
- When $\mathcal{C} = 1 - \mathcal{T}$

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

Highway Networks

- In ResNets we had

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l) + \mathbf{x}_l$$

- In Highway Networks:

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l\mathcal{C}(\mathbf{W_C}, \mathbf{x}_l)$$

- $\mathcal{T}$ is the Transfer Gate, $\mathcal{C}$ is the Carry Gate
- When $\mathcal{C} = 1 - \mathcal{T}$

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

Highway Networks

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

Highway Networks

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

- Dim. of $\mathbf{x}_{l+1}, \mathbf{x}_l, \mathcal{F}(W_l, \mathbf{x}_l), \mathcal{T}(W_T, \mathbf{x}_l)$ must be same

Highway Networks

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

- Dim. of $\mathbf{x}_{l+1}, \mathbf{x}_l, \mathcal{F}(W_l, \mathbf{x}_l), \mathcal{T}(W_T, \mathbf{x}_l)$ must be same
- Note that:
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 0$ then $\mathbf{x}_{l+1} = \mathbf{x}_l$

Highway Networks

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

- Dim. of $\mathbf{x}_{l+1}, \mathbf{x}_l, \mathcal{F}(W_l, \mathbf{x}_l), \mathcal{T}(W_T, \mathbf{x}_l)$ must be same
- Note that:
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 0$ then $\mathbf{x}_{l+1} = \mathbf{x}_l$
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 1$ then $\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)$

Highway Networks

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

- Dim. of $\mathbf{x}_{l+1}, \mathbf{x}_l, \mathcal{F}(W_l, \mathbf{x}_l), \mathcal{T}(W_T, \mathbf{x}_l)$ must be same
- Note that:
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 0$ then $\mathbf{x}_{l+1} = \mathbf{x}_l$
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 1$ then $\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)$
- The highway layer can smoothly vary between a plain layer and just the identity map depending on the transfer gate

Highway Networks

$$\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)\mathcal{T}(W_T, \mathbf{x}_l) + \mathbf{x}_l(1 - \mathcal{T}(W_T, \mathbf{x}_l))$$

- Dim. of $\mathbf{x}_{l+1}, \mathbf{x}_l, \mathcal{F}(W_l, \mathbf{x}_l), \mathcal{T}(W_T, \mathbf{x}_l)$ must be same
- Note that:
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 0$ then $\mathbf{x}_{l+1} = \mathbf{x}_l$
  - If $\mathcal{T}(W_T, \mathbf{x}_l) = 1$ then $\mathbf{x}_{l+1} = \mathcal{F}(W_l, \mathbf{x}_l)$
- The highway layer can smoothly vary between a plain layer and just the identity map depending on the transfer gate
- Like the residual block, the highway layer is then repeated to train deep networks

Highway Networks

- Was published just before Residual Networks (Srivastava, Greff, Schmidhuber, 2015)

Highway Networks

- Was published just before Residual Networks (Srivastava, Greff, Schmidhuber, 2015)
- Contrasts:
  - While more general, has not demonstrated accuracy gains with greater depth

Highway Networks

- Was published just before Residual Networks (Srivastava, Greff, Schmidhuber, 2015)
- Contrasts:
  - While more general, has not demonstrated accuracy gains with greater depth
  - Gates in highway networks are data-dependent while identity shortcuts in ResNets are parameter-free

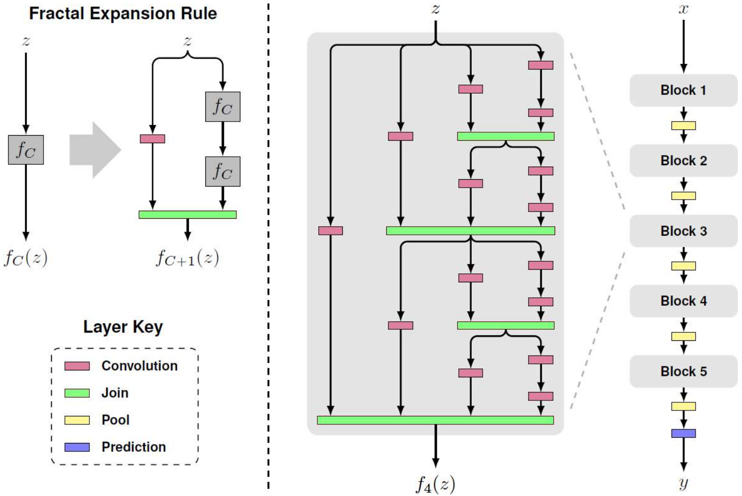
Highway Networks

- Was published just before Residual Networks (Srivastava, Greff, Schmidhuber, 2015)
- Contrasts:
  - While more general, has not demonstrated accuracy gains with greater depth
  - Gates in highway networks are data-dependent while identity shortcuts in ResNets are parameter-free
  - When the gates for shortcut are closed in highway nets, they highway module represents non-residual functions

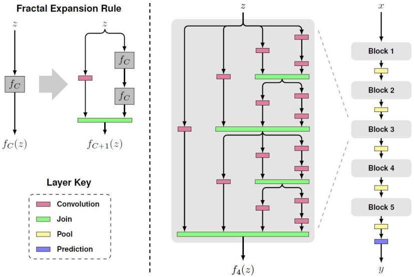
Residuals might not be necessary

Fractal Networks

- Work done here in campus (Gustav Larsson, Michael Maire, Gregory Shakhnarovich)

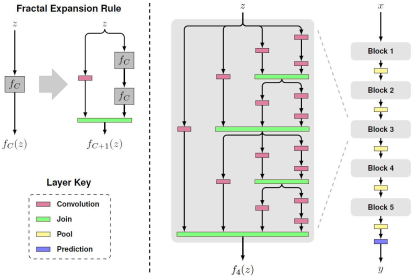


Fractal Networks



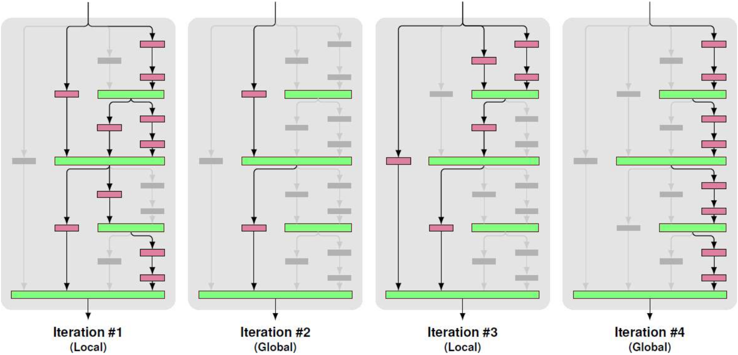
- Base case: $f_1(z) = conv(z)$

Fractal Networks



- Base case: $f_1(z) = conv(z)$
- Recursive definition: $f_{C+1}(z) = [f_C \circ f_C(z)] \oplus [conv(z)]$

Training by DropPath



- Alternate global and local sampling strategies to encourage development of individual columns that can be strong stand-alone subnetworks
- Can train very deep networks with competitive performance without residuals

Performance of Residual Networks might not be due to depth

Viet et al.

- For an output $\mathbf{x}_3$, we have $\mathbf{x}_3 = \mathbf{x}_2 + f_3(\mathbf{x}_2)$

Viet et al.

- For an output $\mathbf{x}_3$, we have $\mathbf{x}_3 = \mathbf{x}_2 + f_3(\mathbf{x}_2)$
- Expanding: $\mathbf{x}_3 = [\mathbf{x}_1 + f_2(\mathbf{x}_1)] + f_3(\mathbf{x}_1 + f_2(\mathbf{x}_1))$

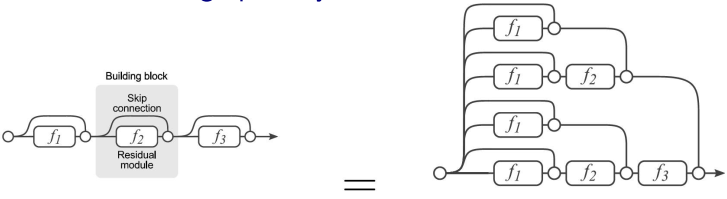
Viet et al.

- For an output $\mathbf{x}_3$, we have $\mathbf{x}_3 = \mathbf{x}_2 + f_3(\mathbf{x}_2)$
- Expanding: $\mathbf{x}_3 = [\mathbf{x}_1 + f_2(\mathbf{x}_1)] + f_3(\mathbf{x}_1 + f_2(\mathbf{x}_1))$
- Expanding further:

$$= [\mathbf{x}_0 + f_1(\mathbf{x}_0) + f_1(\mathbf{x}_0 + f_1(\mathbf{x}_0))] + f_3(\mathbf{x}_0 + f_1(\mathbf{x}_0) + f_1(\mathbf{x}_0 + f_1(\mathbf{x}_0)))$$

Viet et al.

- Unraveled view graphically:



(a) Conventional 3-block residual network

(b) Unraveled view of (a)

Viet et al.

- Many paths from the input to output: 2^n paths

Viet et al.

- Many paths from the input to output: 2^n paths
- In classical visual hierarchy, each layer of processing depends only on the output of the previous layer, this is not true for residual networks due to their inherent structure

Viet et al.

- Many paths from the input to output: 2^n paths
- In classical visual hierarchy, each layer of processing depends only on the output of the previous layer, this is not true for residual networks due to their inherent structure
- Infact, each module $f_i(\cdot)$ can be thought of as being fed data from a mixture of 2^{i-1} different distributions generated from every possible configuration of the previous $i-1$ modules

Viet et al.

- Many paths from the input to output: 2^n paths
- In classical visual hierarchy, each layer of processing depends only on the output of the previous layer, this is not true for residual networks due to their inherent structure
- Infact, each module $f_i(\cdot)$ can be thought of as being fed data from a mixture of 2^{i-1} different distributions generated from every possible configuration of the previous $i-1$ modules
- Viet *et al.* provide experimental evidence that most paths in residual networks are relatively independent of each other, and usually short paths are active

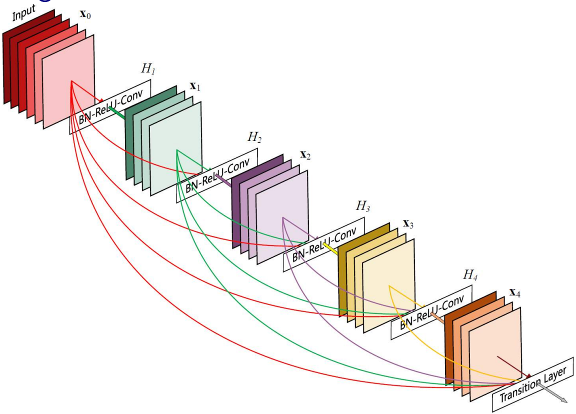
Viet et al.

- Many paths from the input to output: 2^n paths
- In classical visual hierarchy, each layer of processing depends only on the output of the previous layer, this is not true for residual networks due to their inherent structure
- Infact, each module $f_i(\cdot)$ can be thought of as being fed data from a mixture of 2^{i-1} different distributions generated from every possible configuration of the previous $i-1$ modules
- Viet *et al.* provide experimental evidence that most paths in residual networks are relatively independent of each other, and usually short paths are active
- The strength of ResNets may not come from depth, but due to an ensemble of exponentially many shallow networks

DenseNets

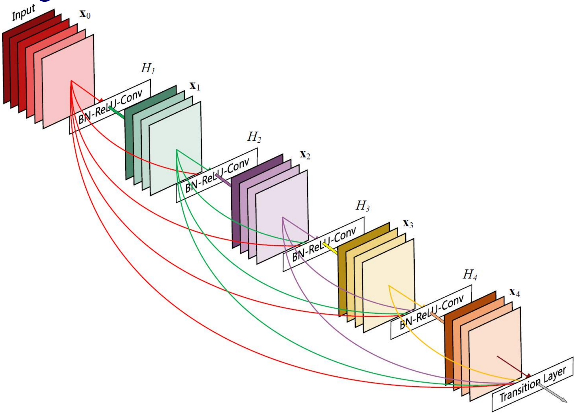
DenseNets

- The lth layer has l inputs, consisting of feature maps of all preceding convolutional blocks



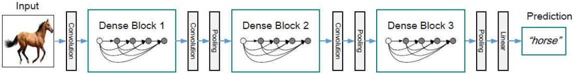
DenseNets

- The lth layer has l inputs, consisting of feature maps of all preceding convolutional blocks



DenseNets

- A Deep Dense Net with 3 dense blocks



Achitectures

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			

- k is *growth factor* (if $\mathcal{F}_l$ produces k feature maps as o/p, it follows that the lth layer has $k \times (l - 1) + k_0$ input feature maps. Where k_0 is the number of channels in the input image)

Results

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network	-	-	10.41	8.81	35.68	-	2.35
All-CNN	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net	-	-	9.69	7.97	-	34.57	1.92
Highway Network	-	-	-	7.72	-	32.39	-
FractalNet	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet	110	1.7M	-	6.61	-	-	-
ResNet	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation)	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Similarity Learning and Siamese Networks

Who is more similar?

Similar Gender

Similar Age

Similar Hair

Similarity depends on the context, which may not be adequately captured by the Euclidean distance on the native feature space

Distance Metric Learning (Linear Case)

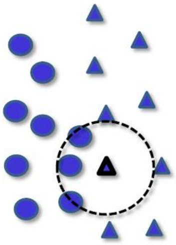
- Learning a distance metric:

Distance Metric Learning (Linear Case)

- Learning a distance metric:
  - Amplify informative directions

Distance Metric Learning (Linear Case)

- Learning a distance metric:
  - Amplify informative directions
  - Squash non-informative directions



Distance Metric Learning (Linear Case)

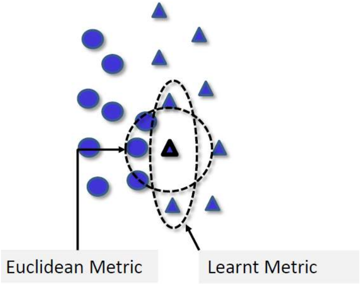
- Learning a distance metric:

Distance Metric Learning (Linear Case)

- Learning a distance metric:
  - Amplify informative directions

Distance Metric Learning (Linear Case)

- Learning a distance metric:
  - Amplify informative directions
  - Squash non-informative directions



Distance Metric Learning (Linear Case)

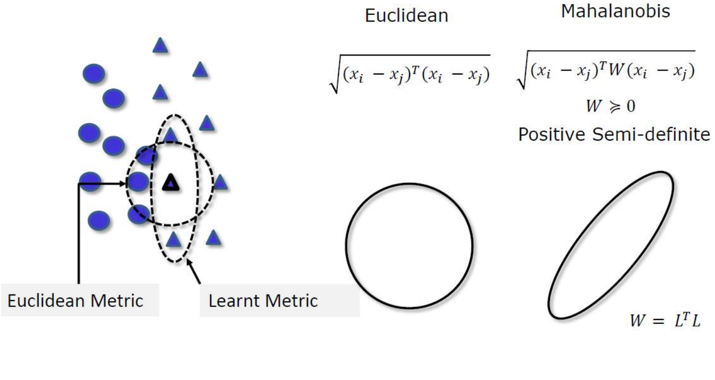
- Learning a distance metric:

Distance Metric Learning (Linear Case)

- Learning a distance metric:
  - Amplify informative directions

Distance Metric Learning (Linear Case)

- Learning a distance metric:
  - Amplify informative directions
  - Squash non-informative directions



- Here the map is $\mathbf{x} \mapsto L\mathbf{x}$

Distance Metric Learning

- Fundamental intuition behind most work in the area:

Distance Metric Learning

- Fundamental intuition behind most work in the area:
  - "Pull" good neighbors (from the correct class for a given point) closer

Distance Metric Learning

- Fundamental intuition behind most work in the area:
  - "Pull" good neighbors (from the correct class for a given point) closer
  - "Pushing" bad neighbors (from the incorrect class for a given point) farther away

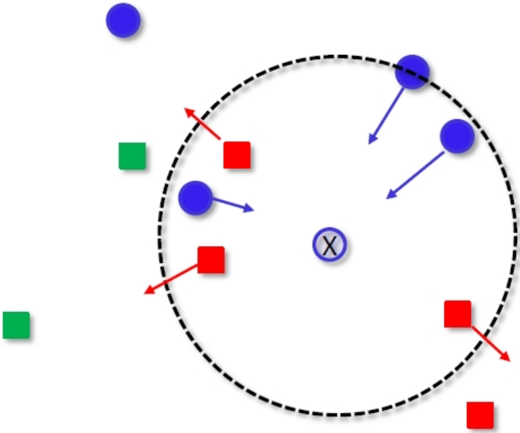
Distance Metric Learning

- Fundamental intuition behind most work in the area:
  - "Pull" good neighbors (from the correct class for a given point) closer
  - "Pushing" bad neighbors (from the incorrect class for a given point) farther away
- "Good" and "Bad" is usually some combination of label agreement and proximity

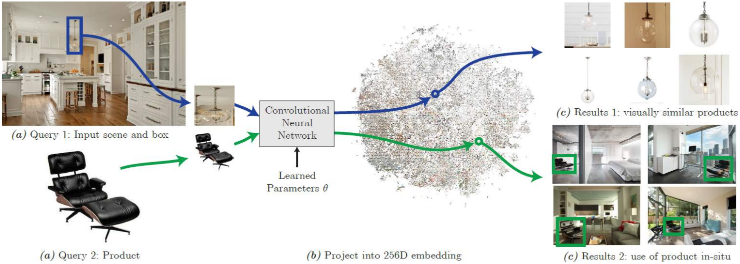
Distance Metric Learning

- Fundamental intuition behind most work in the area:
  - "Pull" good neighbors (from the correct class for a given point) closer
  - "Pushing" bad neighbors (from the incorrect class for a given point) farther away
- "Good" and "Bad" is usually some combination of label agreement and proximity
- Exact formulation of "Good" and "Bad", and how many to consider for each training point, varies from algorithm to algorithm

Distance Metric Learning

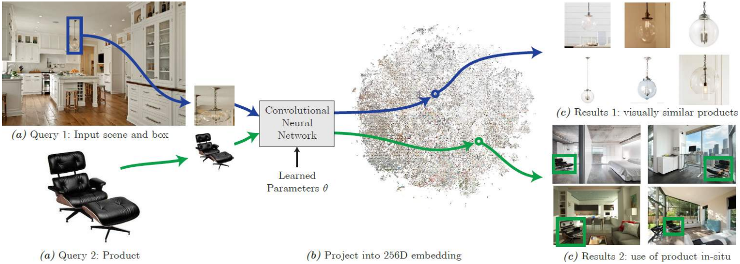


Semantic Embeddings



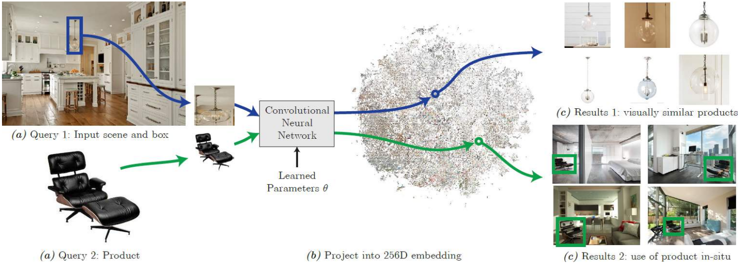
- In general, we can have a non-linear map $\mathbf{x} \mapsto \phi(\mathbf{x})$

Semantic Embeddings



- In general, we can have a non-linear map $\mathbf{x} \mapsto \phi(\mathbf{x})$
- ϕ can be modeled by a neural network!

Semantic Embeddings



- In general, we can have a non-linear map $\mathbf{x} \mapsto \phi(\mathbf{x})$
- ϕ can be modeled by a neural network!
- Reminder: Goal – Given labeled data, learn a metric that has the form $d(\mathbf{x}, \mathbf{x}') = \|\phi(\mathbf{x}) - \phi(\mathbf{x})'\|$ that is compatible with labels

Siamese Networks

- Uses a contrastive cost function:

$$J = \min_{\phi} y_{i,j} D(\mathbf{x}_i, \mathbf{x}_j)^2 + (1 - y_{i,j}) \max\{0, \alpha - D(\mathbf{x}_i, \mathbf{x}_j)^2\}$$

Siamese Networks

- Uses a contrastive cost function:

$$J = \min_{\phi} y_{i,j} D(\mathbf{x}_i, \mathbf{x}_j)^2 + (1 - y_{i,j}) \max\{0, \alpha - D(\mathbf{x}_i, \mathbf{x}_j)^2\}$$

- With $D(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2$

Siamese Networks

- Uses a contrastive cost function:

$$J = \min_{\phi} y_{i,j} D(\mathbf{x}_i, \mathbf{x}_j)^2 + (1 - y_{i,j}) \max\{0, \alpha - D(\mathbf{x}_i, \mathbf{x}_j)^2\}$$

- With $D(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2$
- Two neural networks with shared weights, trained by backpropagation

Triplet Embeddings

- We have an anchor point $\mathbf{x}_i^a$

Triplet Embeddings

- We have an anchor point $\mathbf{x}_i^a$
- We pick a point with the same class $\mathbf{x}_i^p$, and a point with a wrong class $\mathbf{x}_i^n$

Triplet Embeddings

- We have an anchor point $\mathbf{x}_i^a$
- We pick a point with the same class $\mathbf{x}_i^p$, and a point with a wrong class $\mathbf{x}_i^n$
- We then optimize the following objective function:

Triplet Embeddings

- We have an anchor point $\mathbf{x}_i^a$
- We pick a point with the same class $\mathbf{x}_i^p$, and a point with a wrong class $\mathbf{x}_i^n$
- We then optimize the following objective function:

$$J = \min_{\phi} \sum_i \max\{0, D(\mathbf{x}_i^a, \mathbf{x}_i^p)^2 - D(\mathbf{x}_i^a, \mathbf{x}_i^n)^2 + \alpha\}$$

Triplet Embeddings

- We have an anchor point $\mathbf{x}_i^a$
- We pick a point with the same class $\mathbf{x}_i^p$, and a point with a wrong class $\mathbf{x}_i^n$
- We then optimize the following objective function:

$$J = \min_{\phi} \sum_i \max\{0, D(\mathbf{x}_i^a, \mathbf{x}_i^p)^2 - D(\mathbf{x}_i^a, \mathbf{x}_i^n)^2 + \alpha\}$$

- With $D(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2$

Triplet Embeddings

- We have an anchor point $\mathbf{x}_i^a$
- We pick a point with the same class $\mathbf{x}_i^p$, and a point with a wrong class $\mathbf{x}_i^n$
- We then optimize the following objective function:

$$J = \min_{\phi} \sum_i \max\{0, D(\mathbf{x}_i^a, \mathbf{x}_i^p)^2 - D(\mathbf{x}_i^a, \mathbf{x}_i^n)^2 + \alpha\}$$

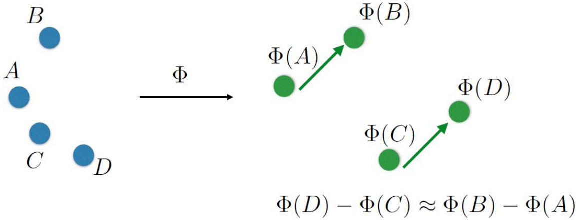
- With $D(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2$
- Three neural networks with shared weights, trained by backpropagation

Application: Visual Analogies

- Analogies: "A is to B as C is to D" e.g. "Paris is to France as London is to UK"

Application: Visual Analogies

- Analogies: "A is to B as C is to D" e.g. "Paris is to France as London is to UK"
- How to solve analogies using embeddings?



Application: One Shot Learning

- Suppose we have learned a semantic embedding for a face database (multiple images per person)

Application: One Shot Learning

- Suppose we have learned a semantic embedding for a face database (multiple images per person)
- Now we have ONE image of a new class given to us. How can we integrate it in our system?

Application: One Shot Learning

- Suppose we have learned a semantic embedding for a face database (multiple images per person)
- Now we have ONE image of a new class given to us. How can we integrate it in our system?
- Leverage examples from other classes and transfer knowledge

Application: One Shot Learning

- Suppose we have learned a semantic embedding for a face database (multiple images per person)

Application: One Shot Learning

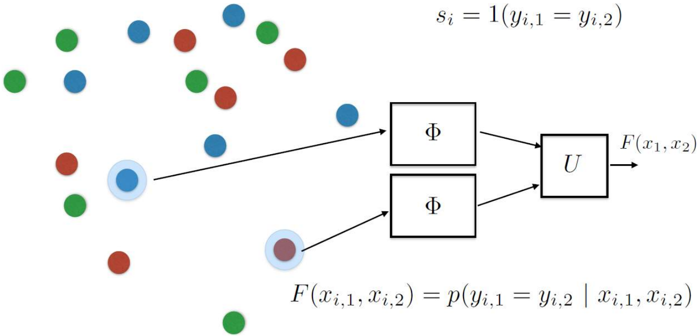
- Suppose we have learned a semantic embedding for a face database (multiple images per person)
- Now we have ONE image of a new class given to us. How can we integrate it in our system?

Application: One Shot Learning

- Suppose we have learned a semantic embedding for a face database (multiple images per person)
- Now we have ONE image of a new class given to us. How can we integrate it in our system?
- Leverage examples from other classes and transfer knowledge

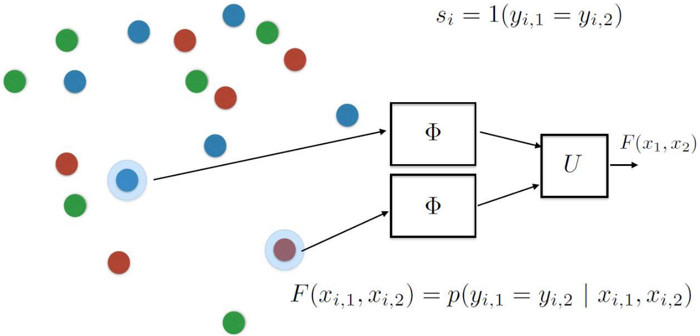
Application: One Shot Learning

- Semantic embedding learning using a siamese network



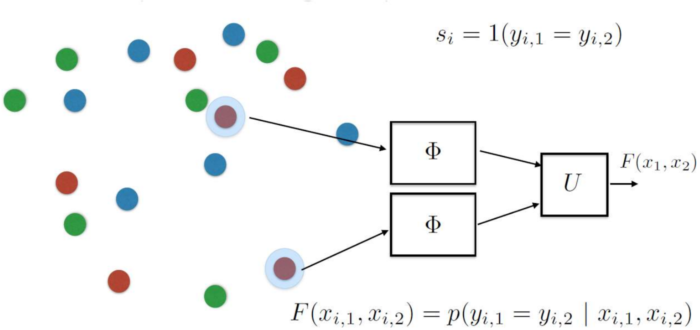
Application: One Shot Learning

- Semantic embedding learning using a siamese network



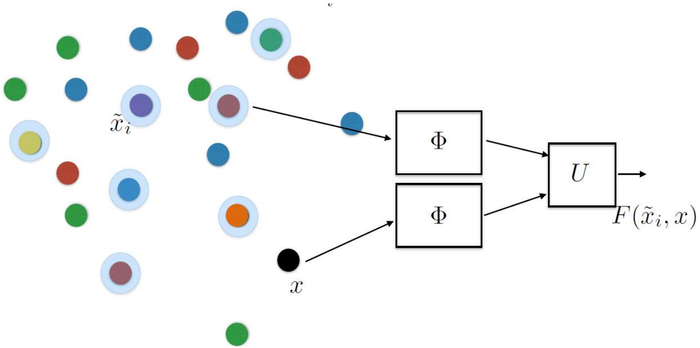
Application: One Shot Learning

- We train the network to detect whether a pair comes from same class or not



Application: One Shot Learning

- Now given one training example $\tilde{x}_i$ from each new class and a query x, estimate label as: $\hat{y} = \arg\max_i F(\tilde{x}_i, x)$



Application: One Shot Learning

- Koch and Salakhutdinov (2015), used a Siamese CNN architecture to get the state of the art performance on the OmniGlot dataset