CNN Techniques

Tuan Nguyen - AI4E

Outline

- Backpropagation
- Resnet introduction/architecture
- Why resnet works?
- Transfer learning
- Data augmentation
- Data synthesis

20 layer

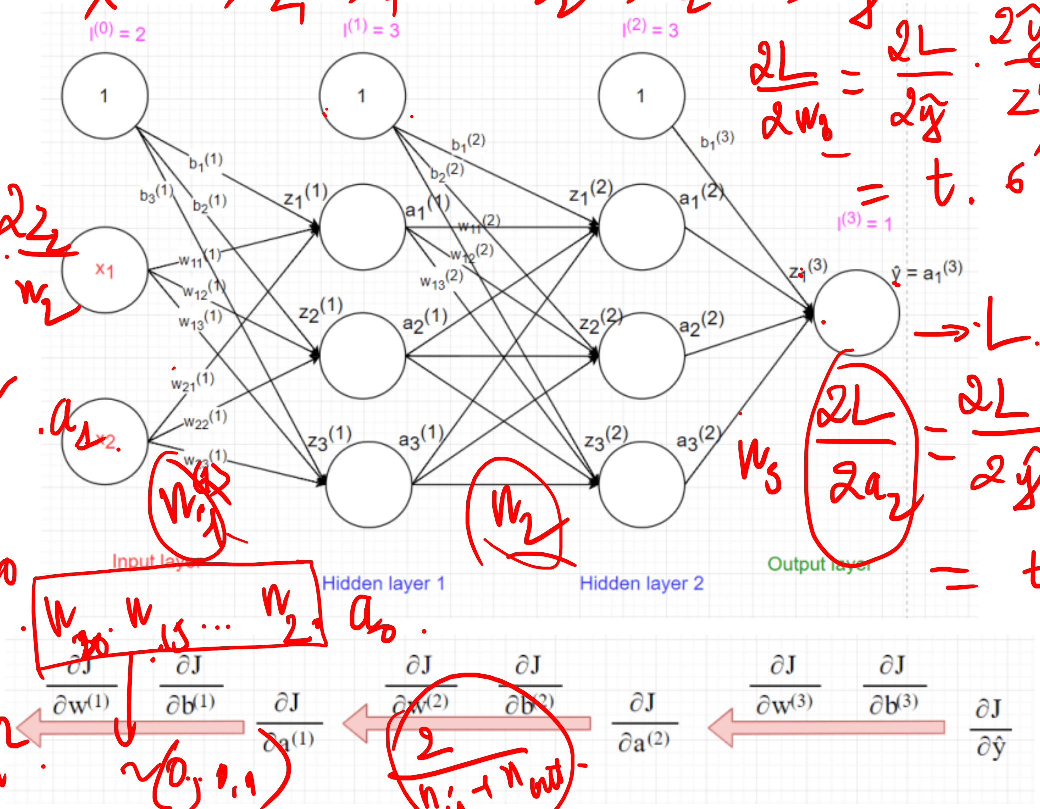
Backpropagation

weight initialization.

$x \rightarrow z_1 \rightarrow a_1 \rightarrow z_2 \rightarrow a_2 \rightarrow \hat{y}$.

$\frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_1}, \ldots$

$\frac{\partial L}{\partial \hat{y}} = t.$



w_1 w_2 w_3

$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial w^{(3)}}$

$= t \cdot 6' \cdot a_2.$

$ab \leq \frac{(a+b)^2}{4}.$

$\rightarrow L$: minimize L

$\frac{\partial L}{\partial a_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot \frac{\partial z_3}{\partial a_2}$

$= t \cdot 6' \cdot w_3.$

$0 < 6 < 1.$

$6' = 6(1-6) \leq \frac{1}{4}.$

$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{z_2} \cdot \frac{\partial z_2}{w_2}$

$= t \cdot 6' \cdot w_3 \cdot 6' \cdot a_1.$

$\frac{\partial L}{\partial a_1}, \ldots$

$\frac{\partial L}{\partial w_1} = t(6')^{20} \cdot \boxed{w_{20} \cdot w_{19} \ldots w_2} \cdot a_0.$

$1 - \tanh^2.$

$\sim (0; 0,1)$

$\frac{2}{n_{in} + n_{out}}$

Backpropagation

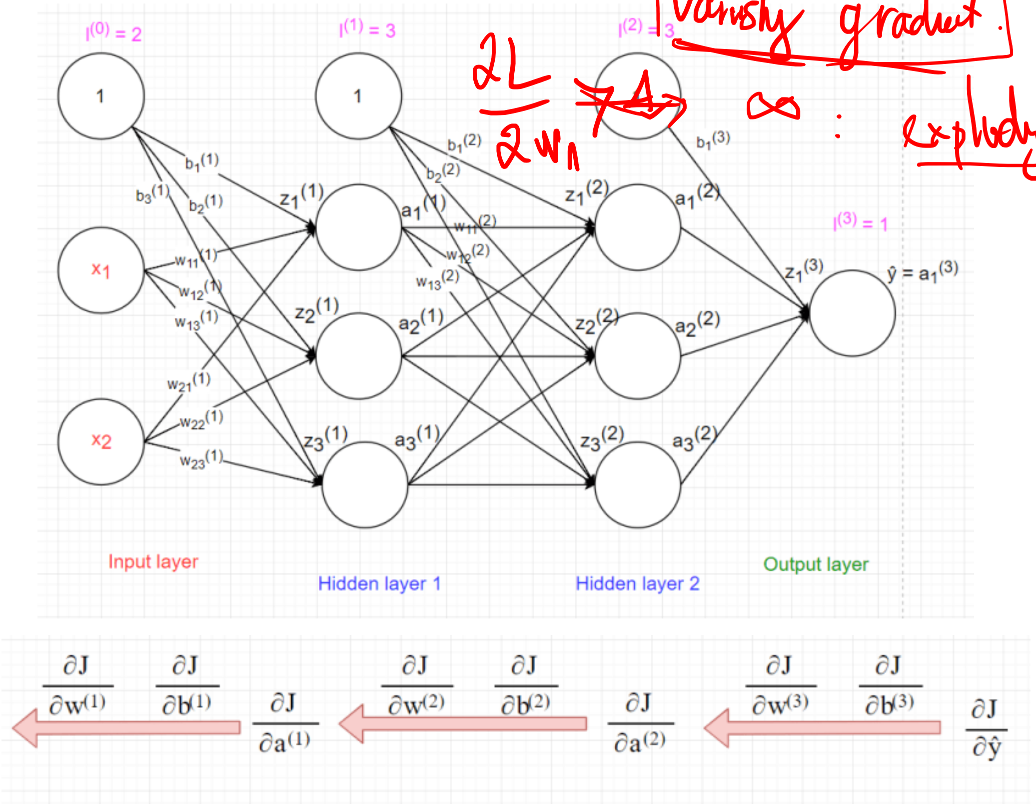
$$\frac{\partial L}{\partial w_1} \rightarrow 0 \;:\; w_1 = w_1 - lr \cdot \frac{\partial L}{\partial w_1}$$

$\downarrow$

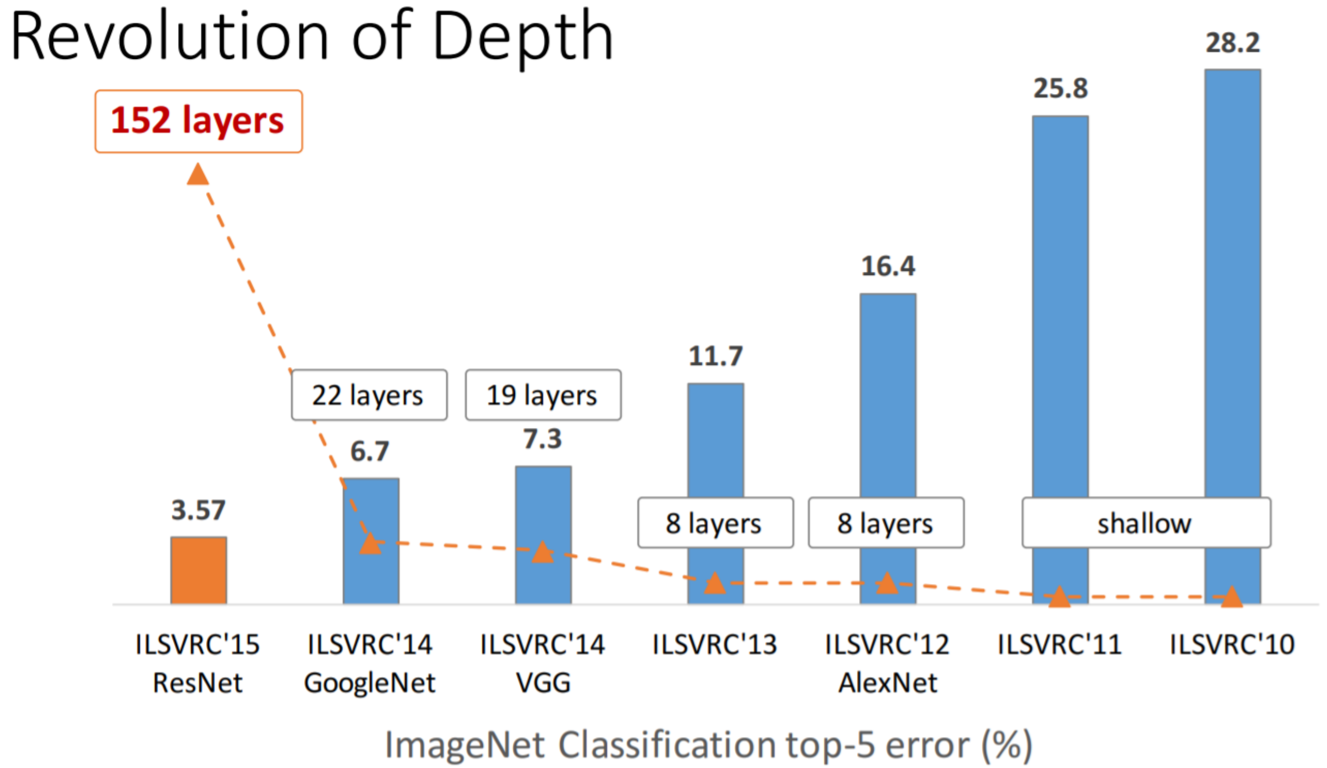
0

$\boxed{\text{Vanishing gradient}}$

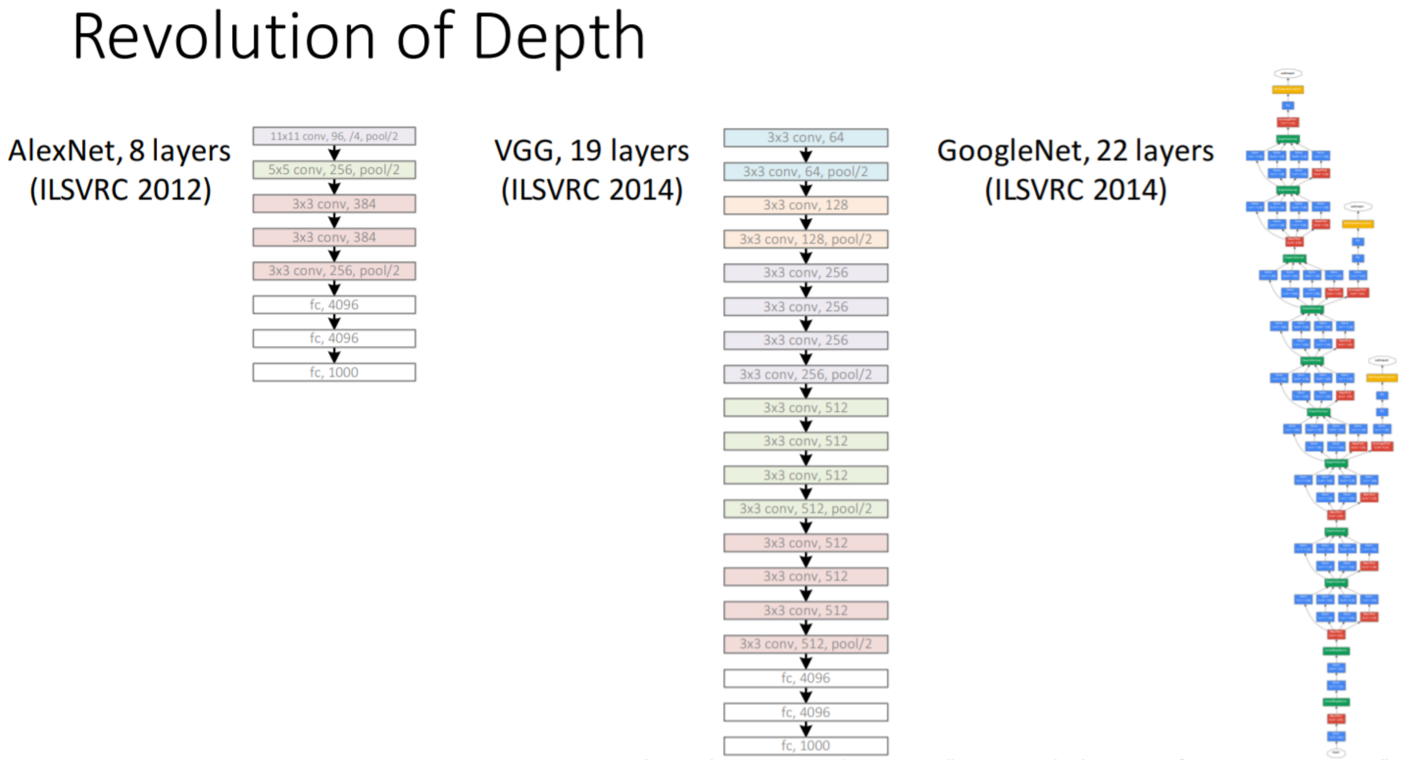
$$\frac{\partial L}{\partial w_1} \rightarrow \infty \;:\; \text{exploding gradient}$$



ImageNet Challange

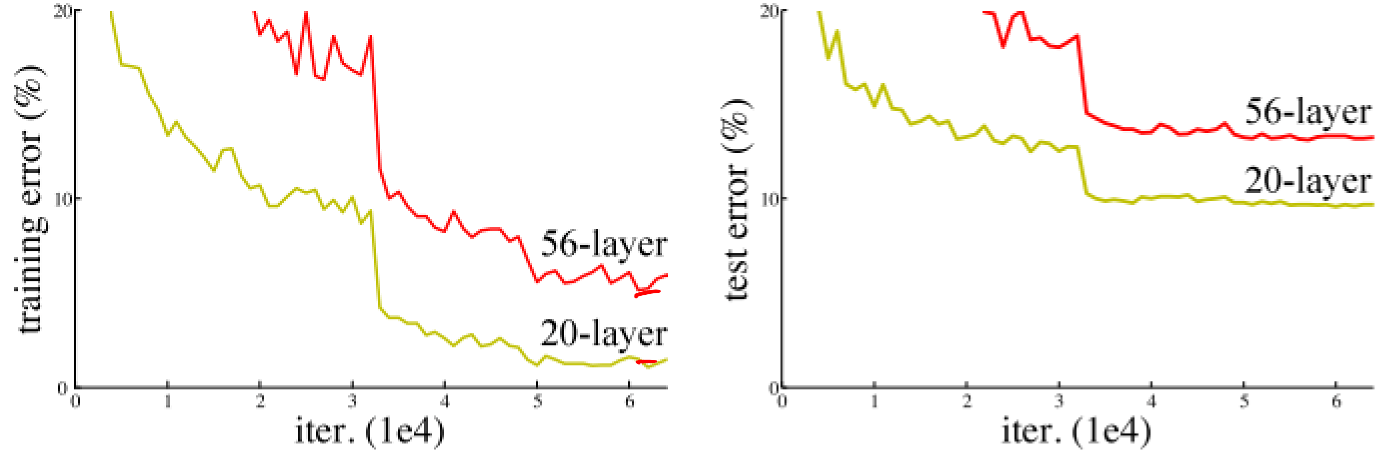


Depth

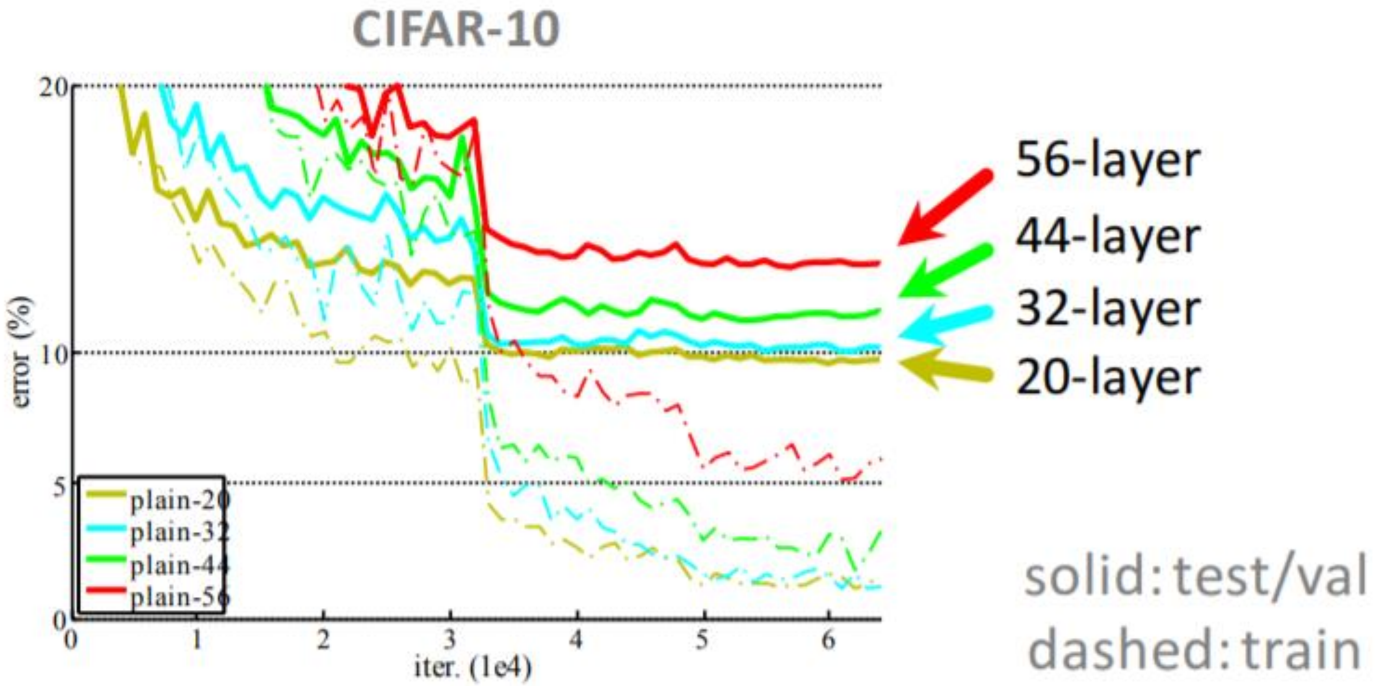

Revolution of Depth
AlexNet, 8 layers
(ILSVRC 2012)
11x11 conv, 96, /4, pool/2
5x5 conv, 256, pool/2
3x3 conv, 384
3x3 conv, 384
3x3 conv, 256, pool/2
fc, 4096
fc, 4096
fc, 1000
VGG, 19 layers
(ILSVRC 2014)
3x3 conv, 64
3x3 conv, 64, pool/2
3x3 conv, 128
3x3 conv, 128, pool/2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256, pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, pool/2
fc, 4096
fc, 4096
fc, 1000
GoogleNet, 22 layers
(ILSVRC 2014)

Stack more layers

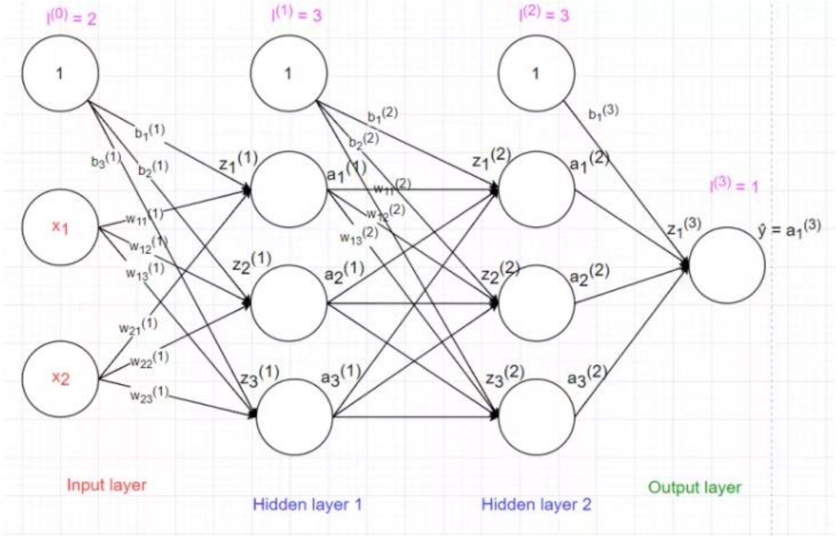
vanishing gradient.



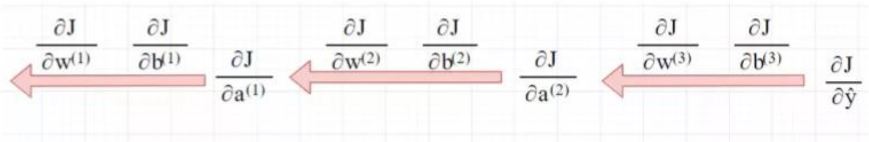
More and more layers



Why?



Mô hình neural network 2-3-3-1



Residual Block

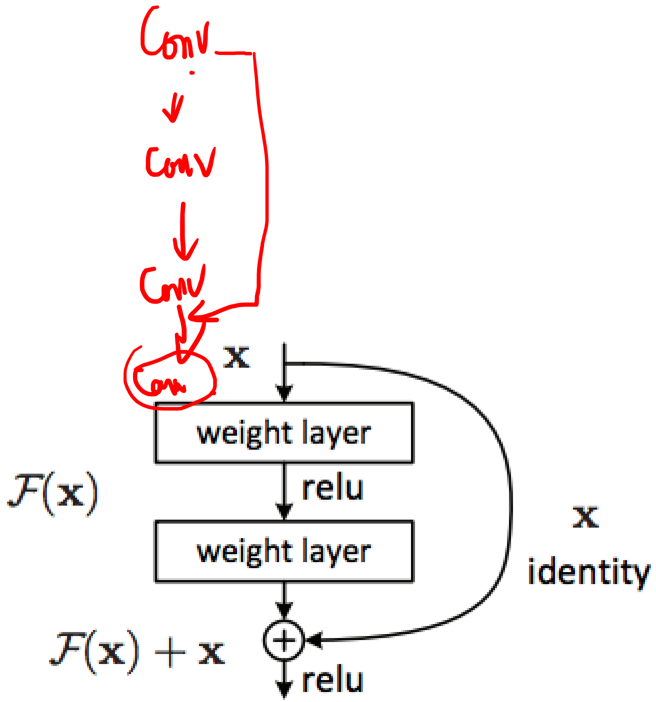


Figure 2. Residual learning: a building block.

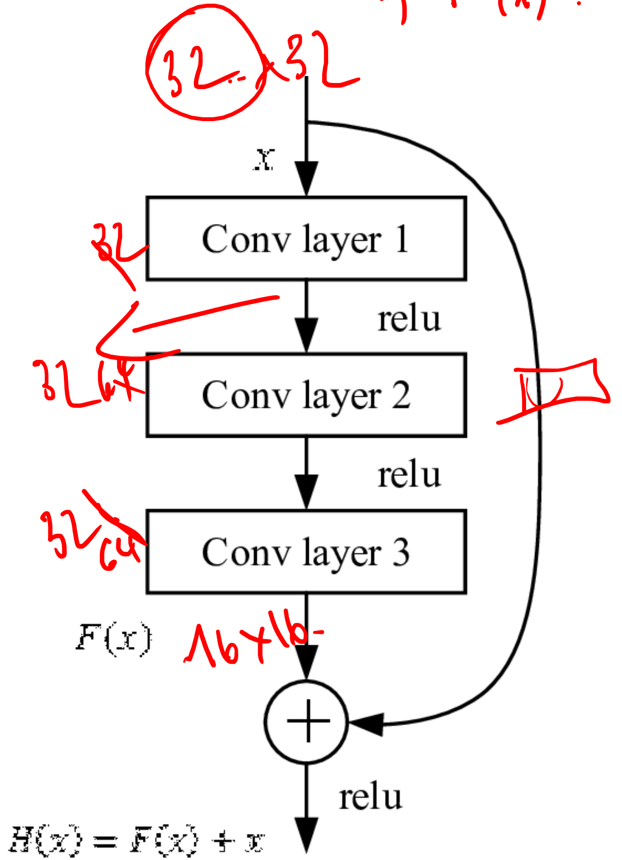
Skip connection

Residual Block

1) F(x) : same dimensio.

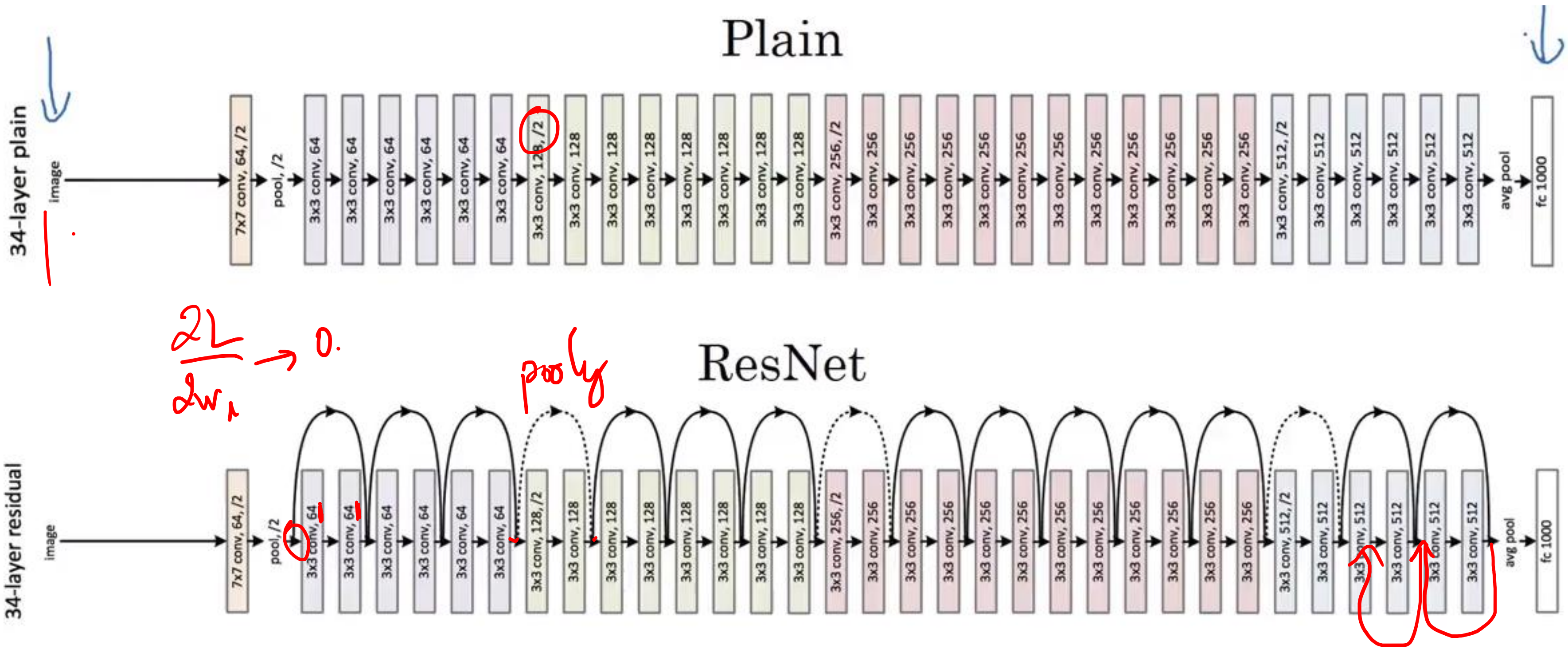
< depth.
size

pooling:

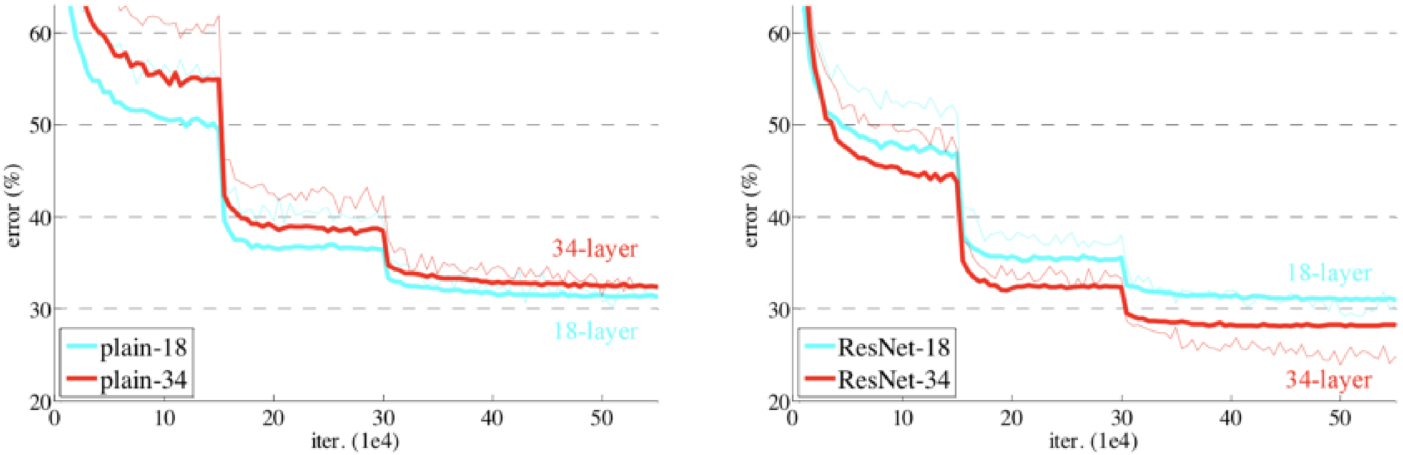


pooling:

Resnet architecture



Resnet result



Resnet doesn’t hurt the performance

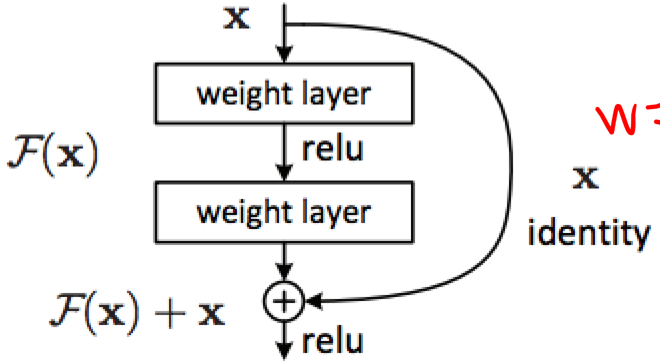
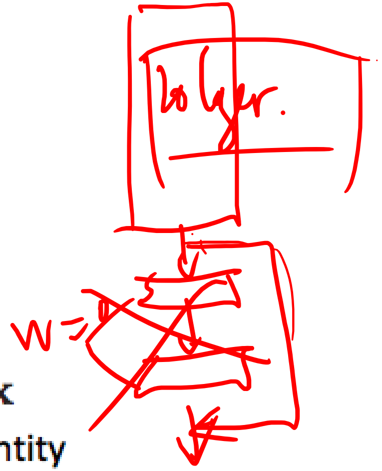


Figure 2. Residual learning: a building block.

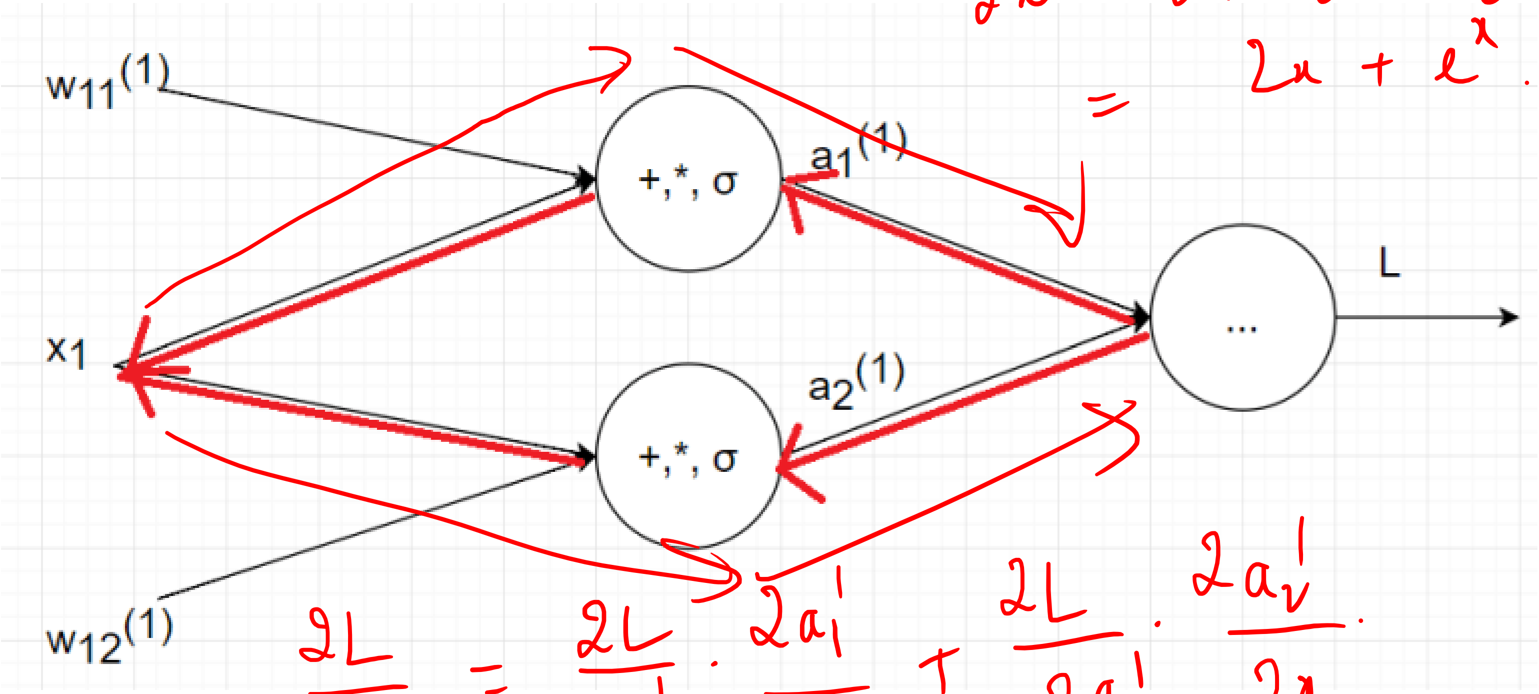


Chain rule

$x \to x^2,\ e^x \to L = x^2 + e^x$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial x^2} \cdot \frac{\partial x^2}{\partial x} + \frac{\partial L}{\partial e^x} \cdot \frac{\partial e^x}{\partial x}$$

$$= 2x + e^x$$



$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial a_1^1} \cdot \frac{\partial a_1^1}{\partial x_1} + \frac{\partial L}{\partial a_2^1} \cdot \frac{\partial a_2^1}{\partial x_1}$$

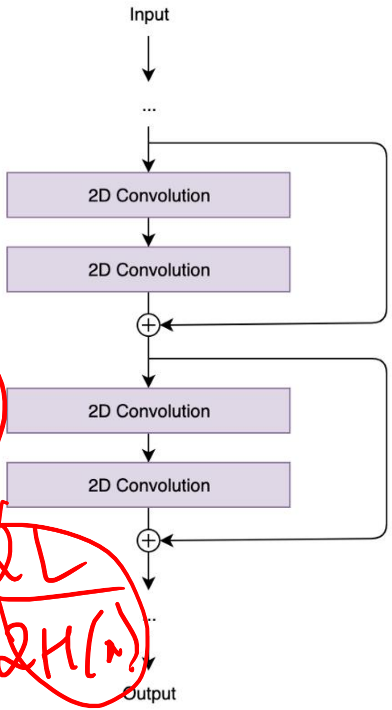
Vanishing gradient

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H_x}\cdot\frac{\partial H(x)}{\partial x}$$

$$= \frac{\partial L}{\partial H(x)}\cdot\left(\frac{\partial F(x)}{\partial(x)} + 1\right)$$

$$= \boxed{\frac{\partial L}{\partial H(x)}\cdot\frac{\partial F(x)}{\partial(x)}} + \frac{\partial L}{\partial H(x)}$$

original



x

conv

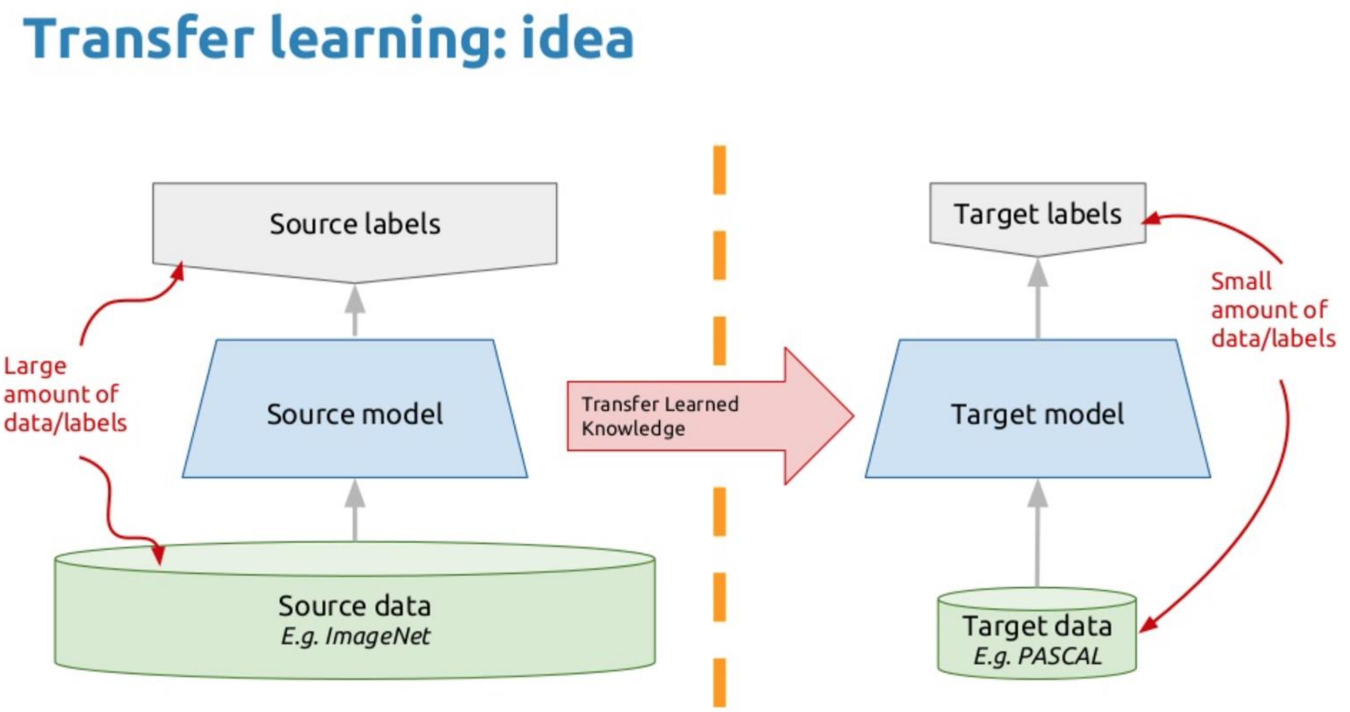
conv

$F(x)$

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

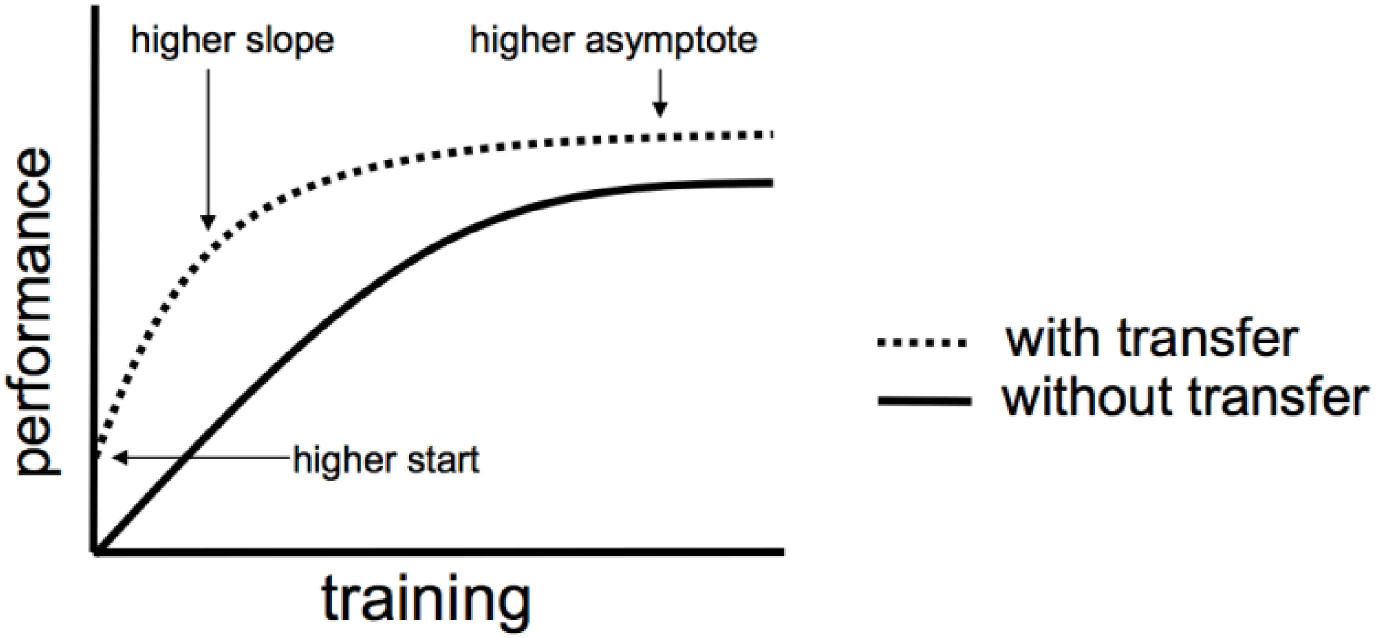
L

Transfer learning



Transfer learning

pre-trained model
architecture : VGG16, resnet19
dataset : ImageNet

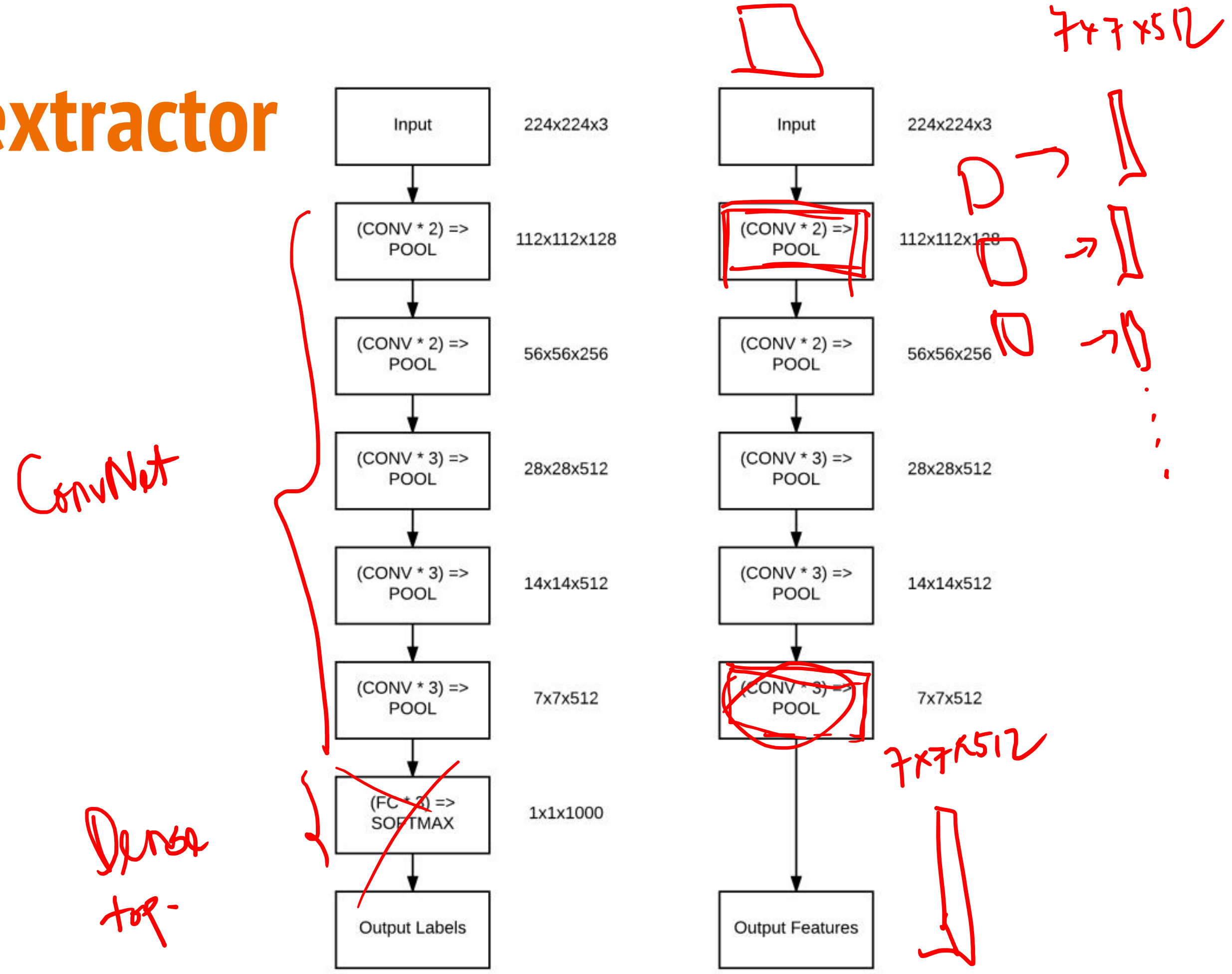


Types of transfer learning

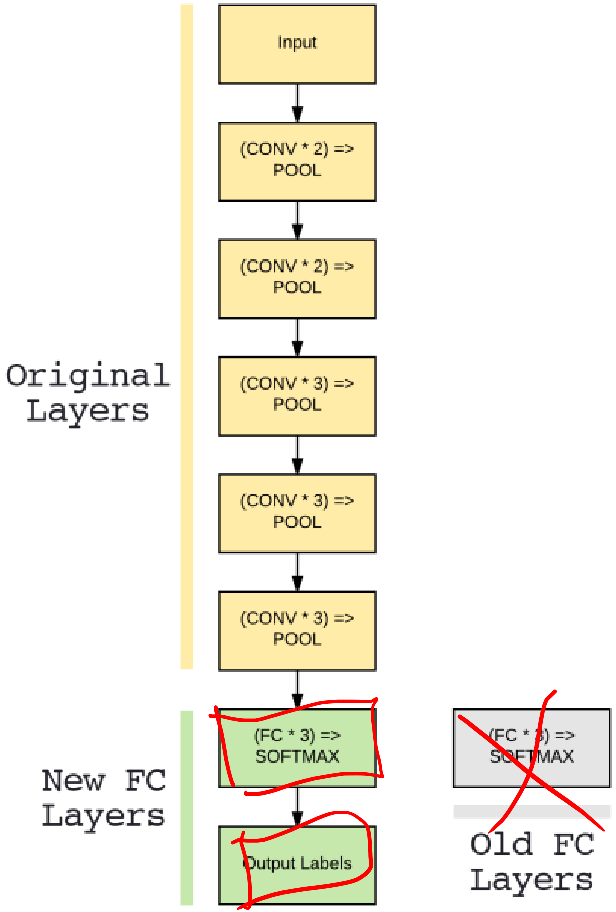
There are 2 types of transfer learning:

- Feature extractor: use pre-trained model to extract features, then use linear classifier (linear SVM, softmax classifier,..) to get the result.
- Fine-tuning: add more layers to the pre-trained model to continue training.

Feature extractor



Fine-tuning

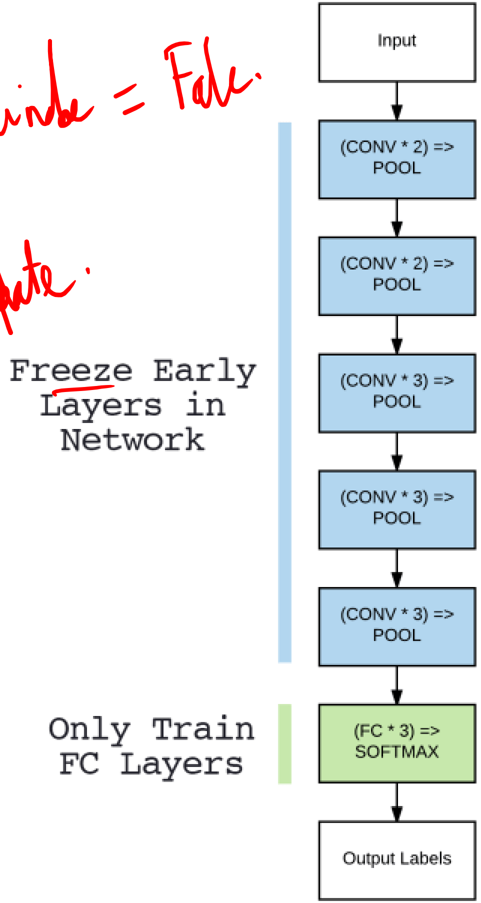


Original Layers
Input
(CONV * 2) => POOL
(CONV * 2) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
New FC Layers
(FC * 3) => SOFTMAX
Output Labels
(FC * 3) => SOFTMAX
Old FC Layers

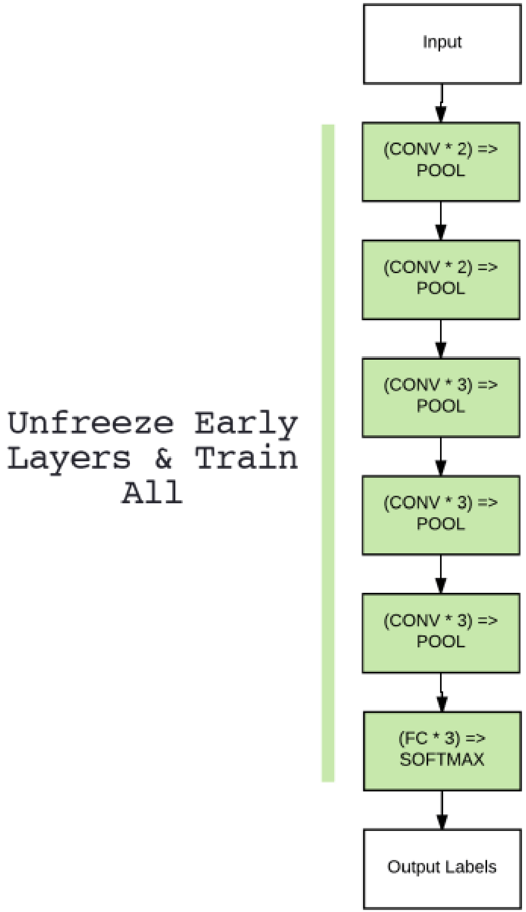
Phase-1 fine-tuning

.trainable = False.

ko update.



Phase-2 fine-tuning



When to use transfer learning?

	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*

Data augmentation

Three ways to improve data

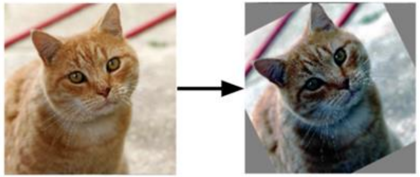
1 - Collect more

- expensive
- requires manual labor

2 - Synthesize

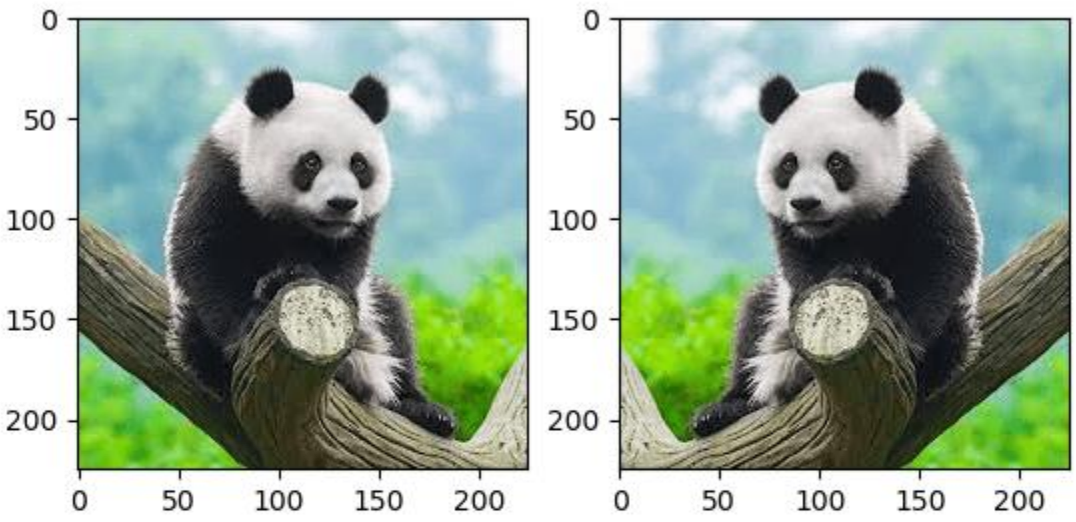
- complicated
- might not truly represent the real data

3 - Augment

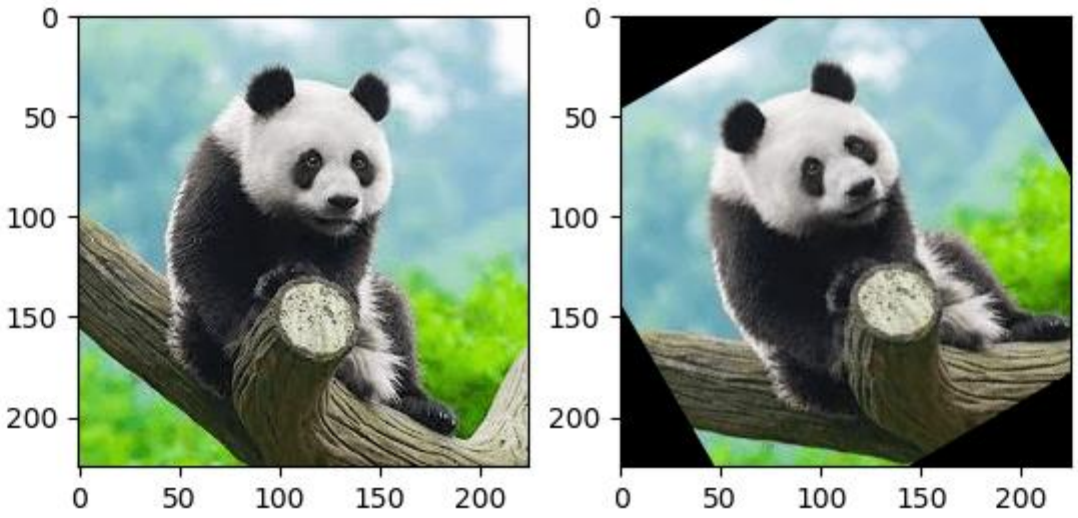


- simple
- but finding a good augmentation strategy takes lots of trial & error (**=time of AI engineers**)

Flip



Rotate



Crop & scale

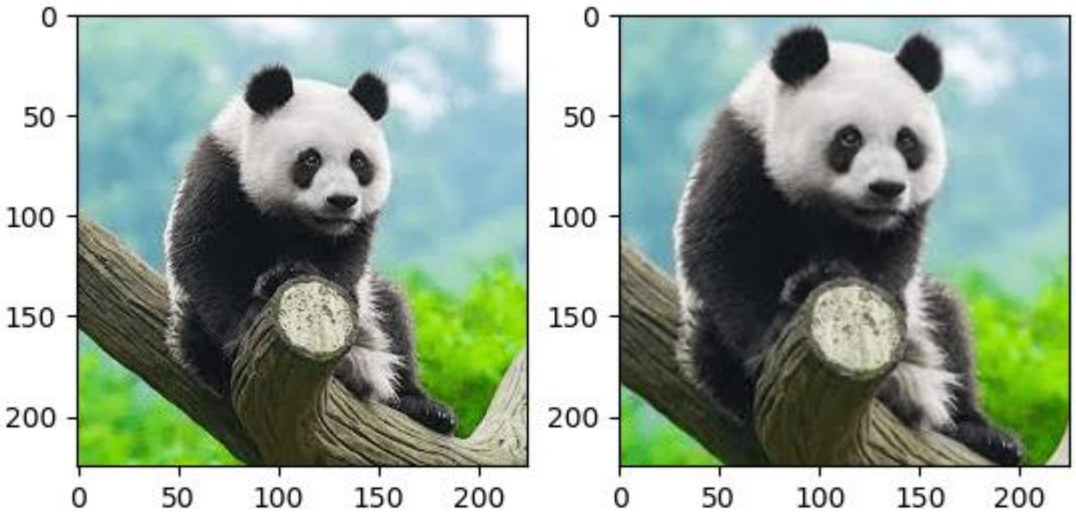


Image synthesize

Q&A


@LateNightSeth
THANK YOU!