ResNet

Deep Residual Learning for Image Recognition

박성진

1. Introduction

- **Is deeper network always better?**
  - **Deep networks naturally integrate low/mid/high level features**
  - **The levels of features can be enriched by the number of stacked layers(depth)**
  - **Recent evidence reveals that network depth is crucial importance**

1. Introduction

- **Is deeper network always better?**
  - **Deep networks naturally integrate low/mid/high level features**
  - **The levels of features can be enriched by the number of stacked layers(depth)**
  - **Recent evidence reveals that network depth is crucial importance**

AlexNet	VggNet	GoogLeNet
16.4%	7.3%	6.7%
8 layers	19 layers	22 layers
2012	2014	2014

1. Introduction

- Is deeper network always better?
  - Deep networks naturally integrate low/mid/high level features
  - The levels of features can be enriched by the number of stacked layers(depth)
  - Recent evidence reveals that network depth is crucial importance

AlexNet	VggNet	GoogLeNet
16.4%	7.3%	6.7%
8 layers	19 layers	22 layers
2012	2014	2014

1. Introduction

- Is deeper network always better?
  - Deep networks naturally integrate low/mid/high level features
  - The levels of features can be enriched by the number of stacked layers(depth)
  - Recent evidence reveals that network depth is crucial importance

AlexNet	VggNet	GoogLeNet
16.4%	7.3%	6.7%
8 layers	19 layers	22 layers
2012	2014	2014

1. Introduction

- Is deeper network always better?
  - Deep networks naturally integrate low/mid/high level features
  - The levels of features can be enriched by the number of stacked layers(depth)
  - Recent evidence reveals that network depth is crucial importance

AlexNet	VggNet	GoogLeNet
16.4%	7.3%	6.7%
8 layers	19 layers	22 layers
2012	2014	2014

1. Introduction

- **Is deeper network always better?**

AlexNet	ResNet	GoogLeNet
16.4%	3.57%	6.7%
8 layers	152 layers	22 layers
2012	2015	2014

1. Introduction

- Is deeper network always better?
  - Deep networks naturally integrate low/mid/high level features
  - The levels of features can be enriched by the number of stacked layers(depth)
  - Recent evidence reveals that network depth is crucial importance

Yes !!!

Deeper Network is better!!!

1. Introduction

- Is learning better networks as easy as stacking more layers?
  - Problem of vanishing/exploding gradients

    - Better initialization methods
    - Better normalization
    - Better Activation function

  - Number of parameters

    - Deeper bottleneck architectures (1x1 Conv)
    - 2 times 3x3 Conv

1. Introduction

- **Is learning better networks as easy as stacking more layers?**
  - **Problem of vanishing/exploding gradients**

    - Better initialization methods
    - Better normalization
    - Better Activation function

  - **Number of parameters**

    - Deeper bottleneck architectures (1x1 Conv)
    - 2 times 3x3 Conv

- **Any other problems?**
  - **Overfitting**

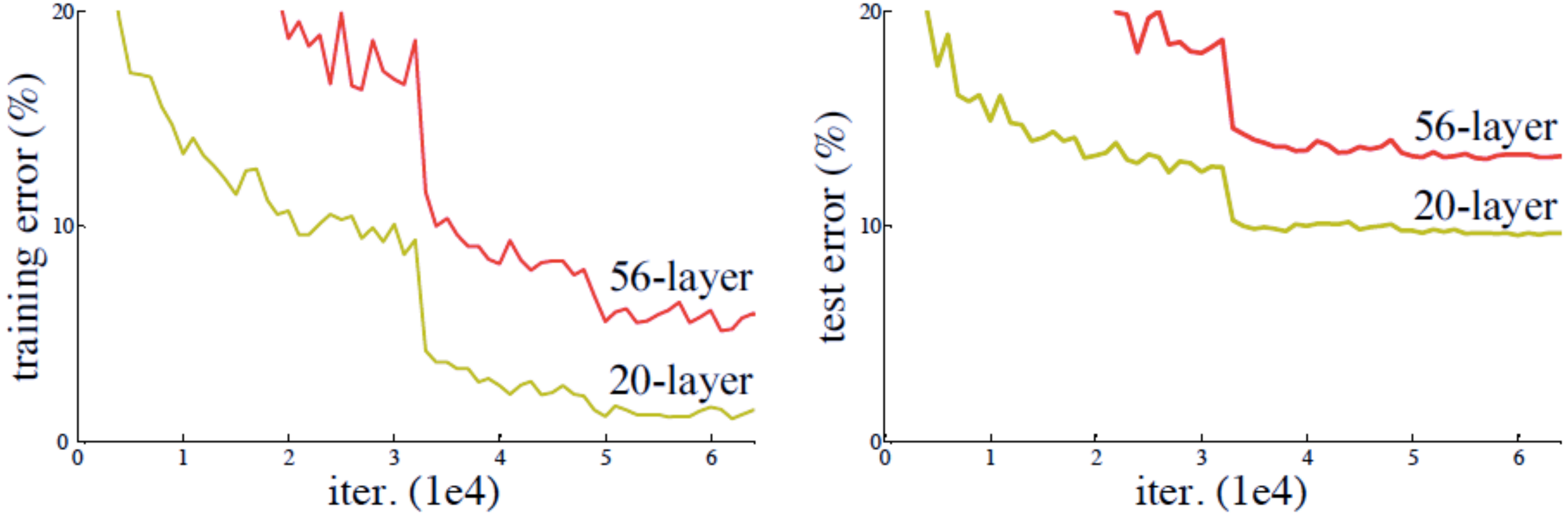
1. Introduction

- Is learning better networks as easy as stacking more layers?
  - Problem of vanishing/exploding gradients
    - Better initialization methods
    - Better batch normalization
    - Better Activation function(ReLU)
- Any other problems?
  - Overfitting

No Overfitting But Degradation !!!

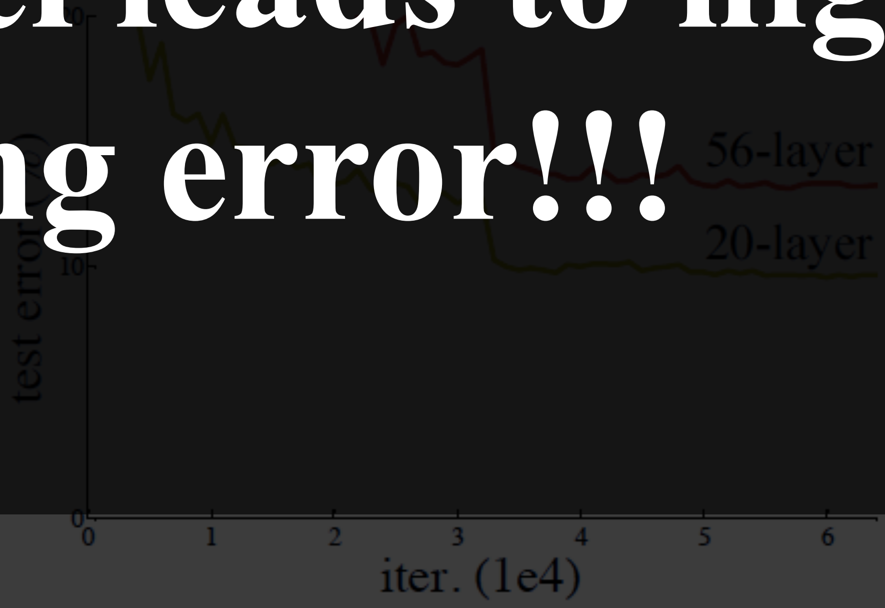
1. Introduction

- **Degradation Problem**
  - **More depth but lower performance**



1. Introduction

- **Degradation Problem**
  - **More depth but lower performance**



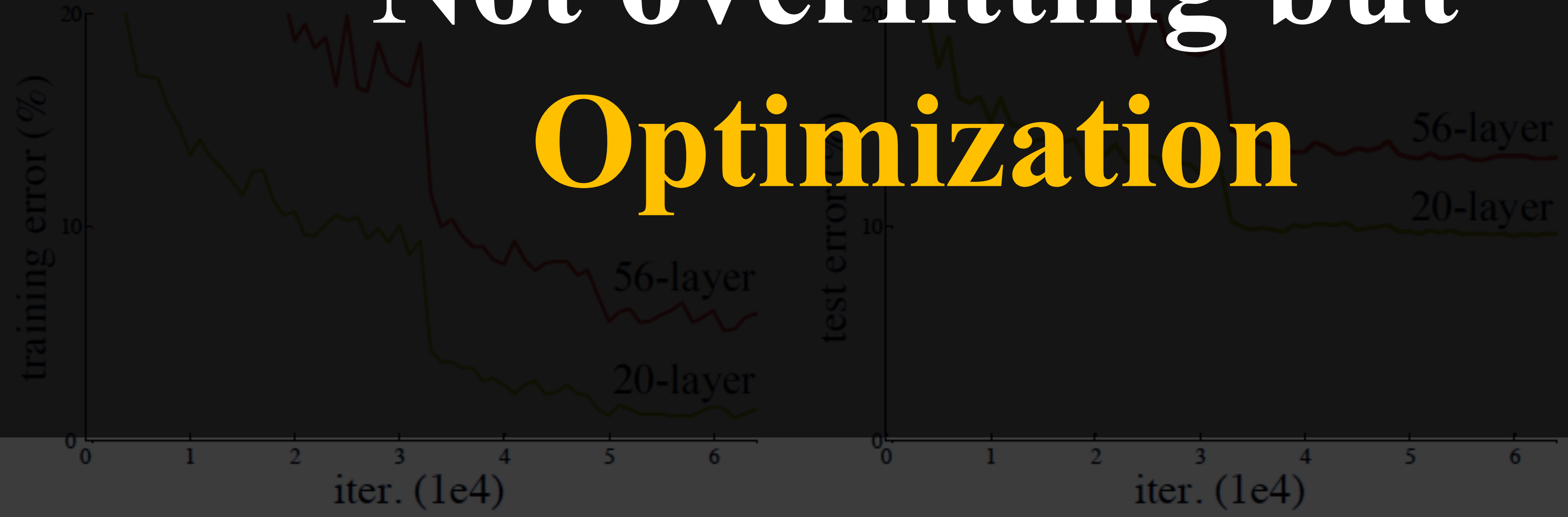
Deeper model leads to higher training error!!!

2. Model Architecture

- **Hypothesis : the problem is an optimization problem deeper models are harder to optimize.**

1. Introduction

- **Degradation Problem**
  - **More depth but lower performance**



Not overfitting but
Optimization

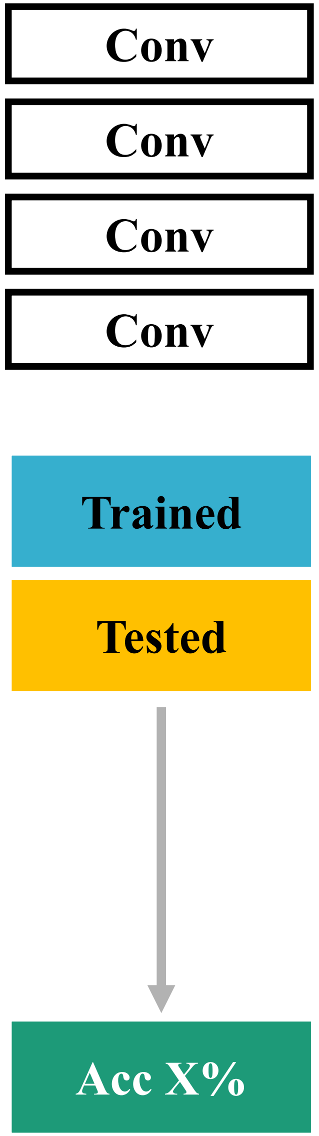
2. Model Architecture

- Hypothesis : the problem is an optimization problem deeper models are harder to optimize.

- The deeper model should be able to perform at least as well as the shallower model.

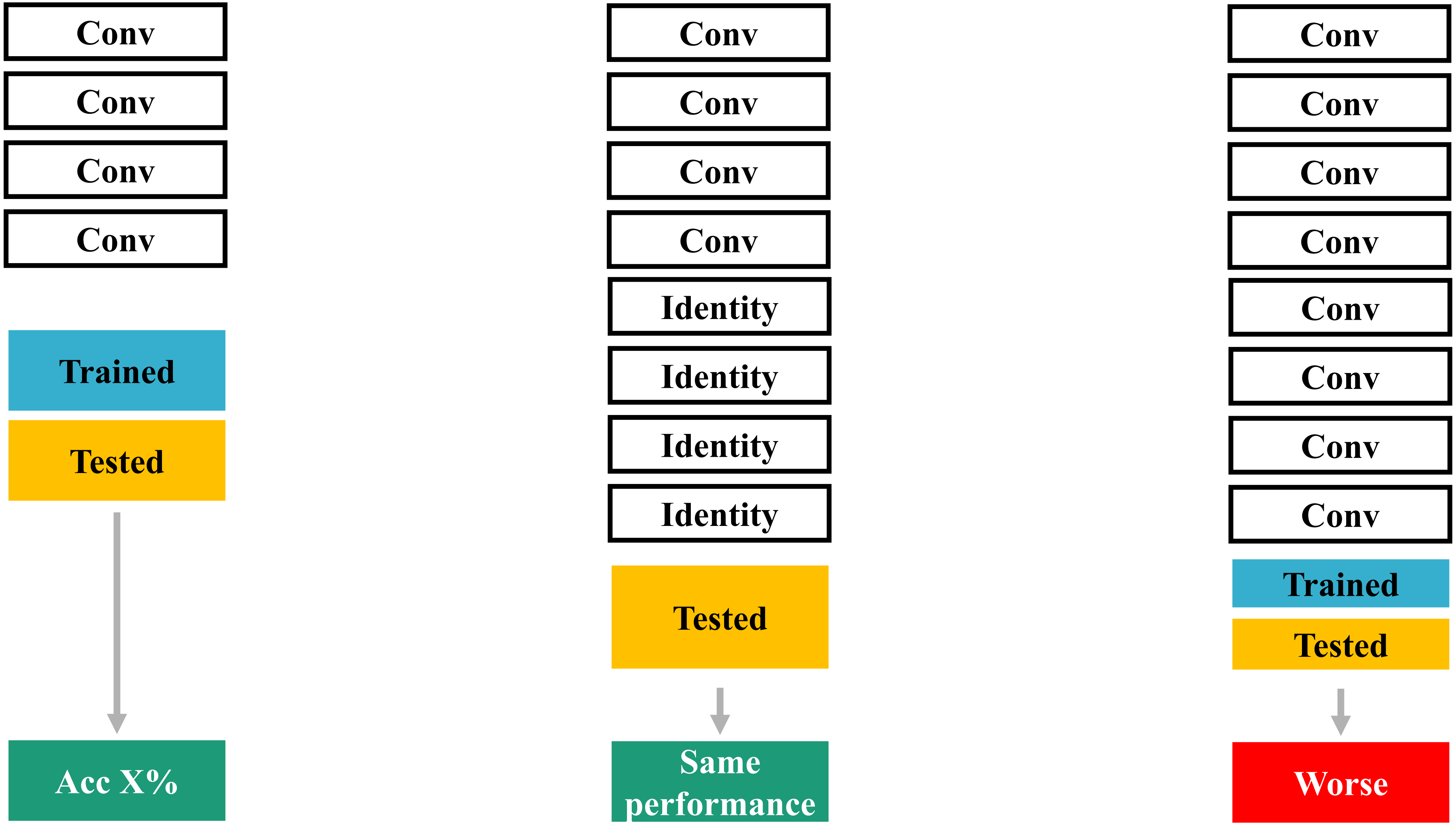
- A solution by construction is copying the learned layers from the shallower model ands setting additional layers to identity mapping.

2. Model Architecture



2. Model Architecture

2. Model Architecture



2. N

Our Current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution(or unable to do so in feasible time)

Trained

Tested

Acc X%

Identity

Identity

Identity

Identity

Tested

Same performance

Conv

Conv

Conv

Conv

Trained

Tested

Worse

2. N

Our Current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution(or unable to do so in feasible time)

Solvers might have difficulties in approximating Identity mappings by multiple nonlinear layers

Identity

Tested

Same performance

Conv

Trained

Tested

Worse

Acc X%

2. N

Our Current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution(or unable to do so in feasible time)

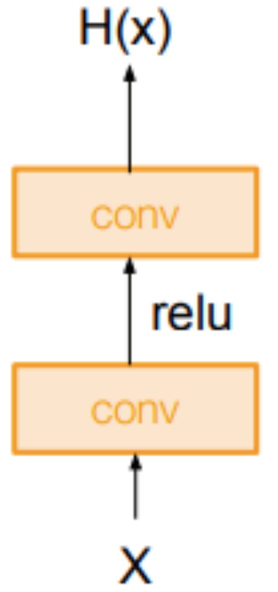
Solvers might have difficulties in approximating Identity mappings by multiple nonlinear layers

Add explicit identity connections and solvers may Drive the weights of the multiple nonlinear layers toward zero

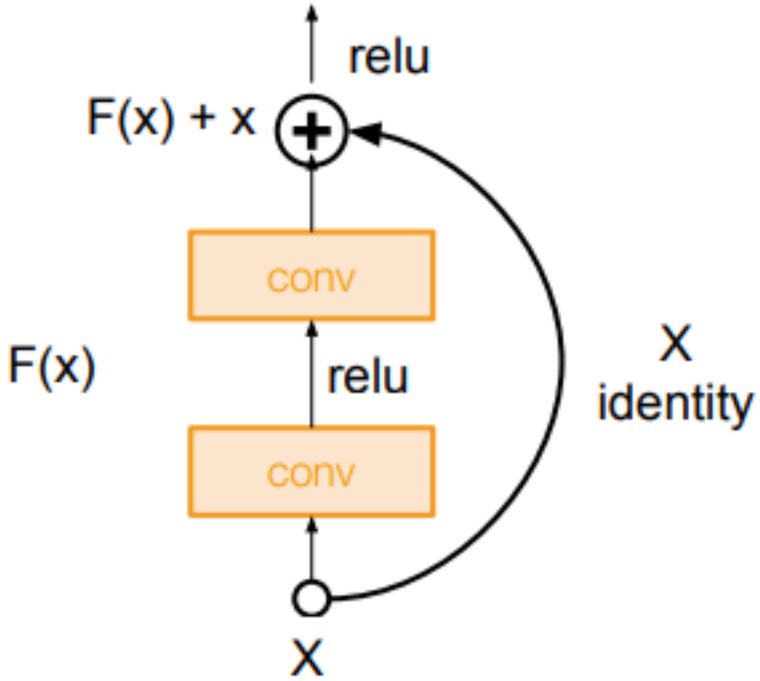
performance

2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**



“Plain” layers

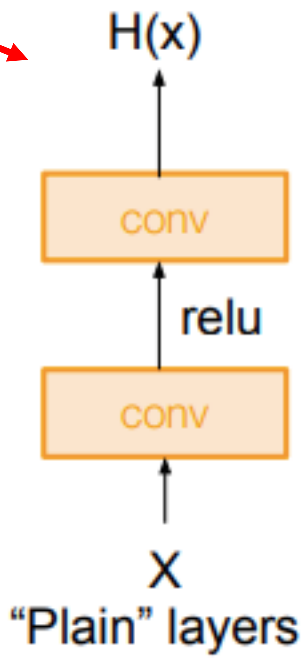


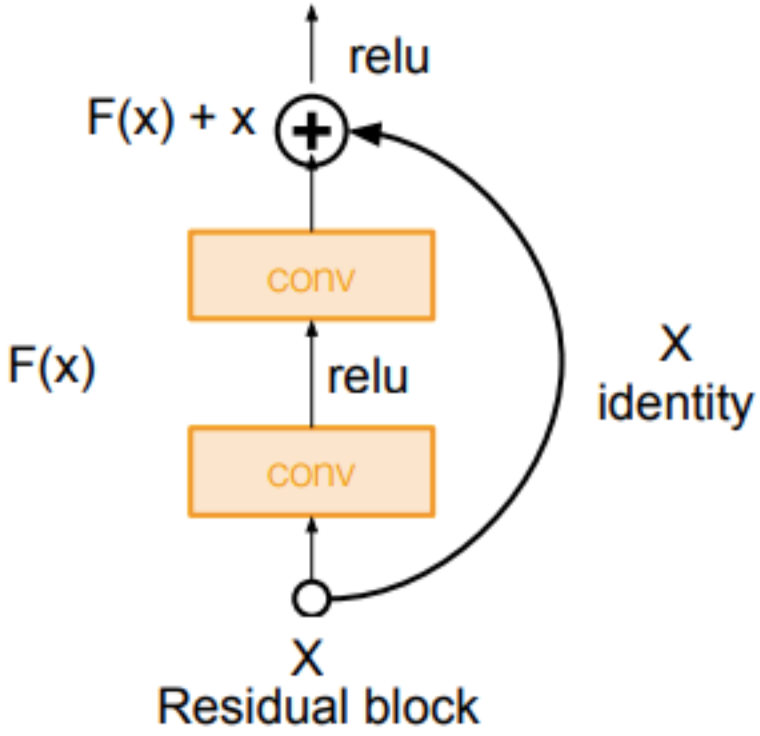
Residual block

2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of** **directly trying to fit a desired underlying mapping**

***H*(*x*) directly**

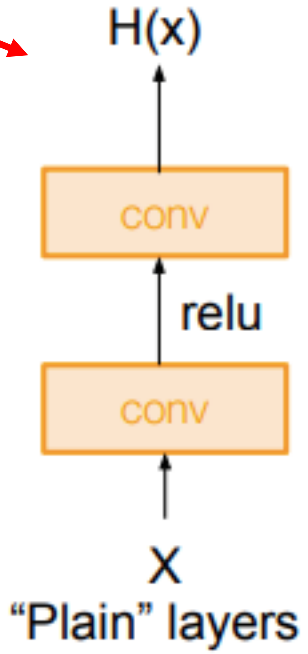




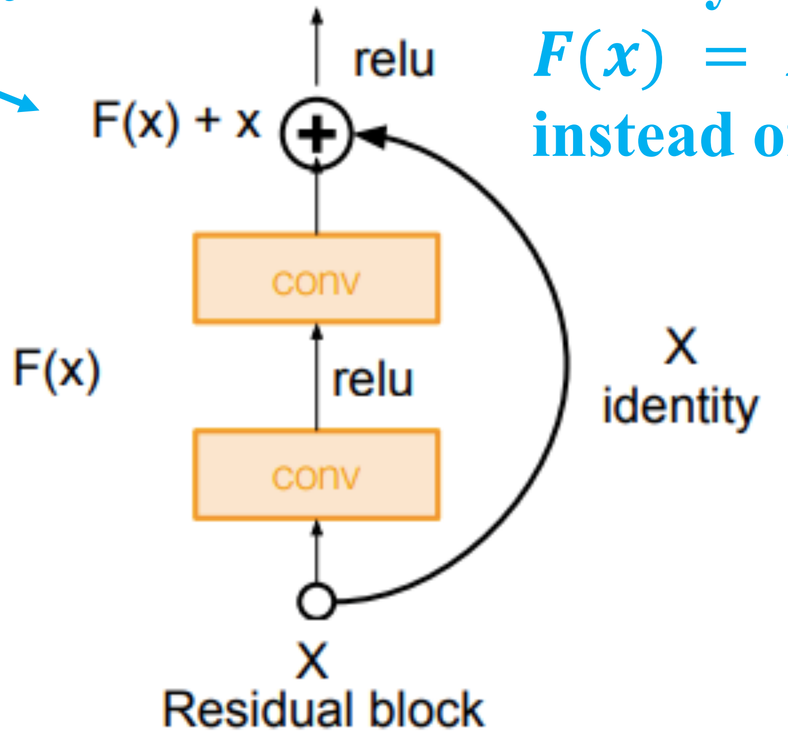
2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

$\boldsymbol{H(x)}$ **directly**



$\boldsymbol{H(x) = F(x) + x}$



Use layers to fit residual
$\boldsymbol{F(x) = H(x) - x}$
instead of $\boldsymbol{H(x)}$ **directly**

2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

***$H(x)$* directly**

Use layers to fit residual $F(x) = H(x) - x$ instead of

It's hard to learn $\mathrm{H}(x)$ with very deep network

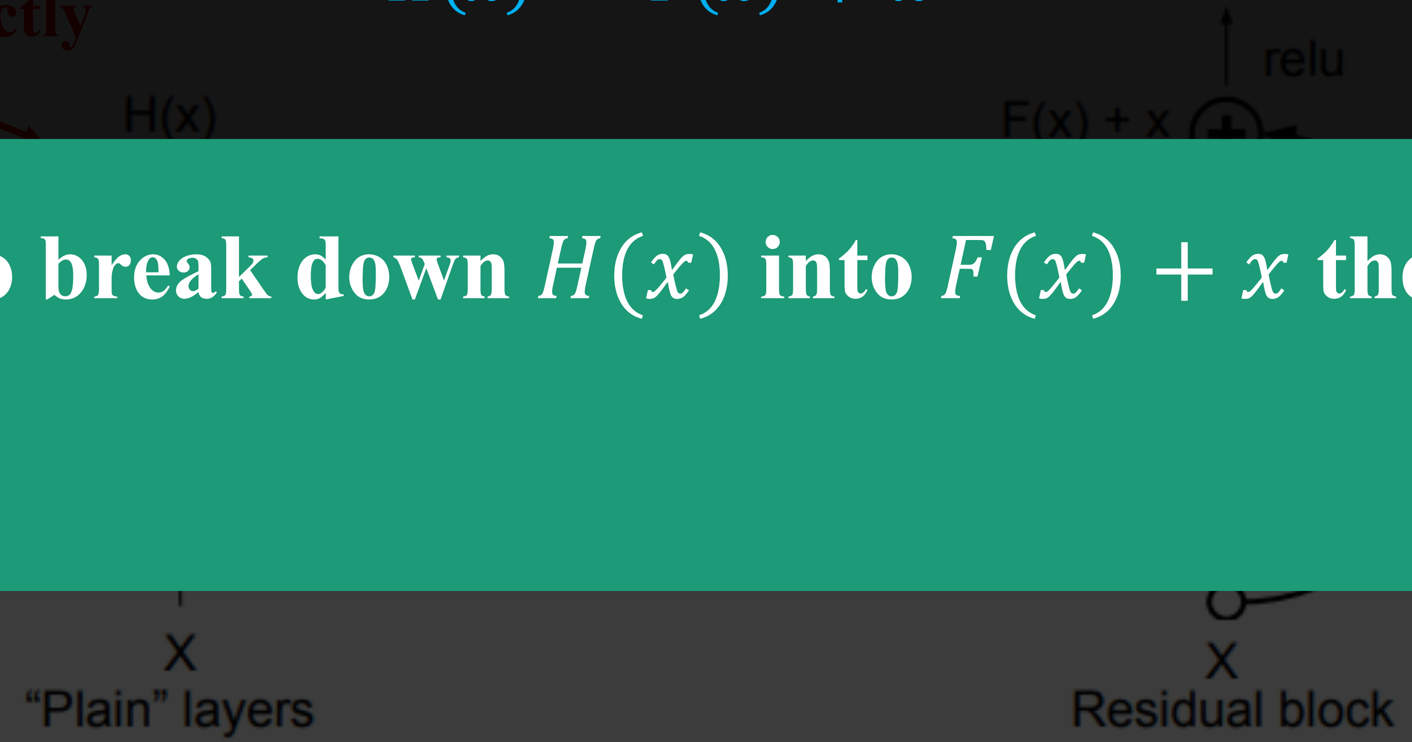
2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

$$\boldsymbol{H(x) = F(x) + x}$$

$H(x)$ directly

Use layers to fit residual $F(x) = H(x) - x$ instead of



So break down $H(x)$ into $F(x) + x$ then learn $F(x)$

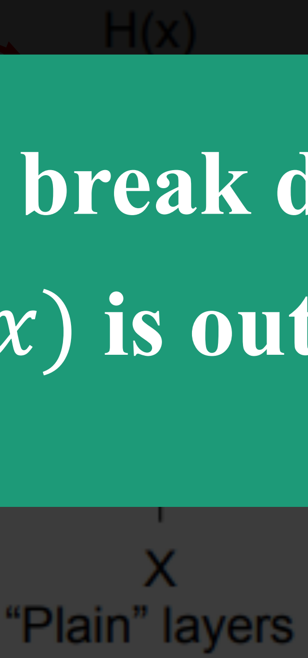
2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

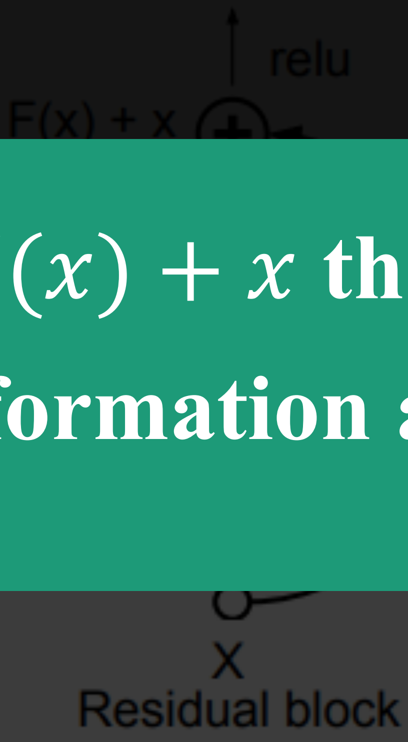
$$H(x) = F(x) + x$$

$H(x)$ directly

Use layers to fit residual $F(x) = H(x) - x$ instead of



"Plain" layers



Residual block

So break down $H(x)$ into $F(x) + x$ then learn $F(x)$

$F(x)$ is output of our transformation and x is input

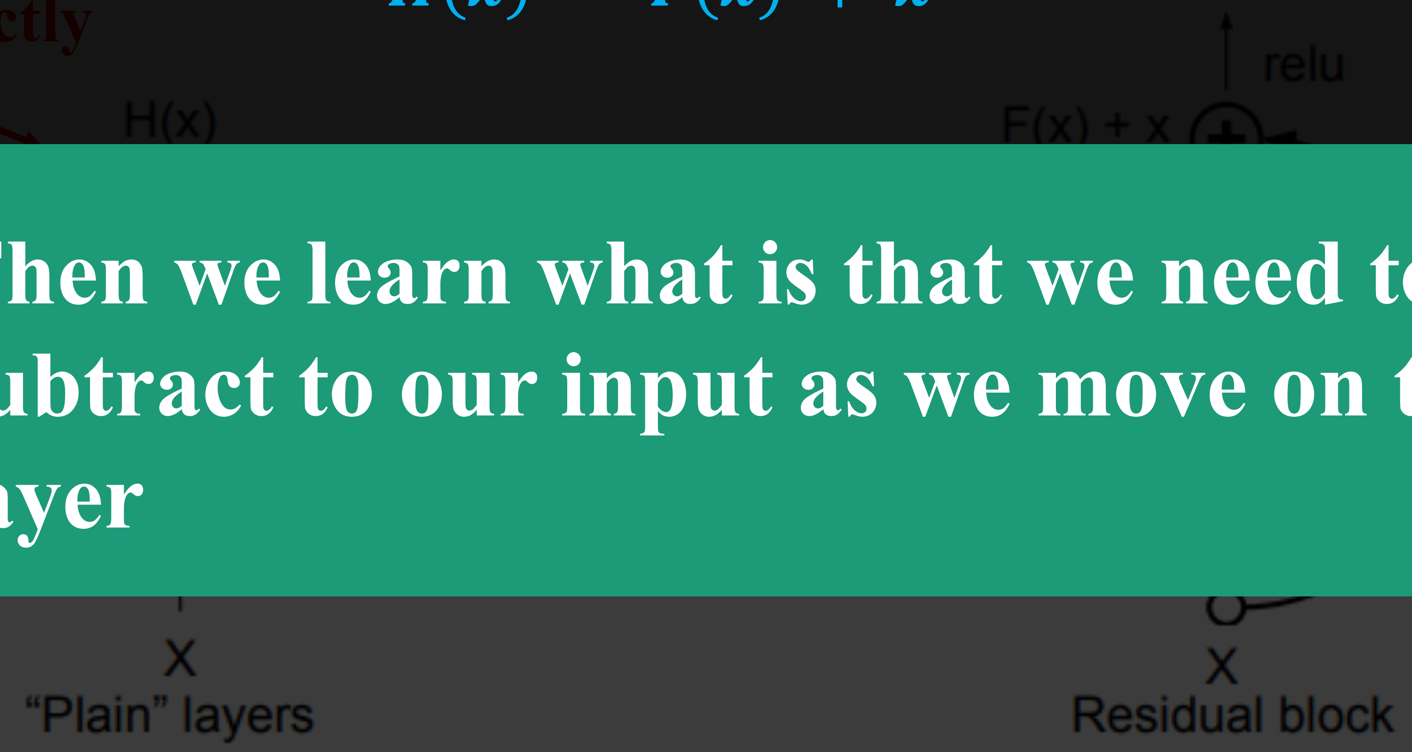
2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

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

$H(x)$ directly

Use layers to fit residual $F(x) = H(x) - x$ instead of



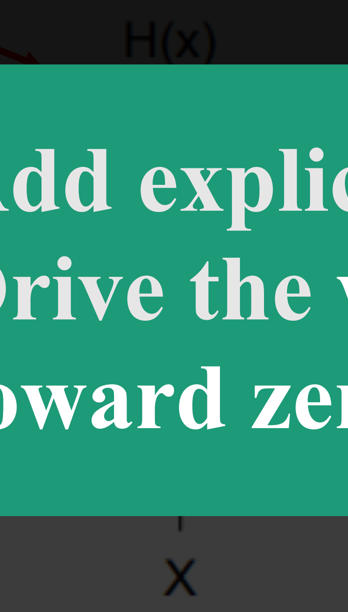
Then we learn what is that we need to add or subtract to our input as we move on to the next layer

2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

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

$H(x)$ directly



"Plain" layers

Use layers to fit residual $F(x) = H(x) - x$ instead of

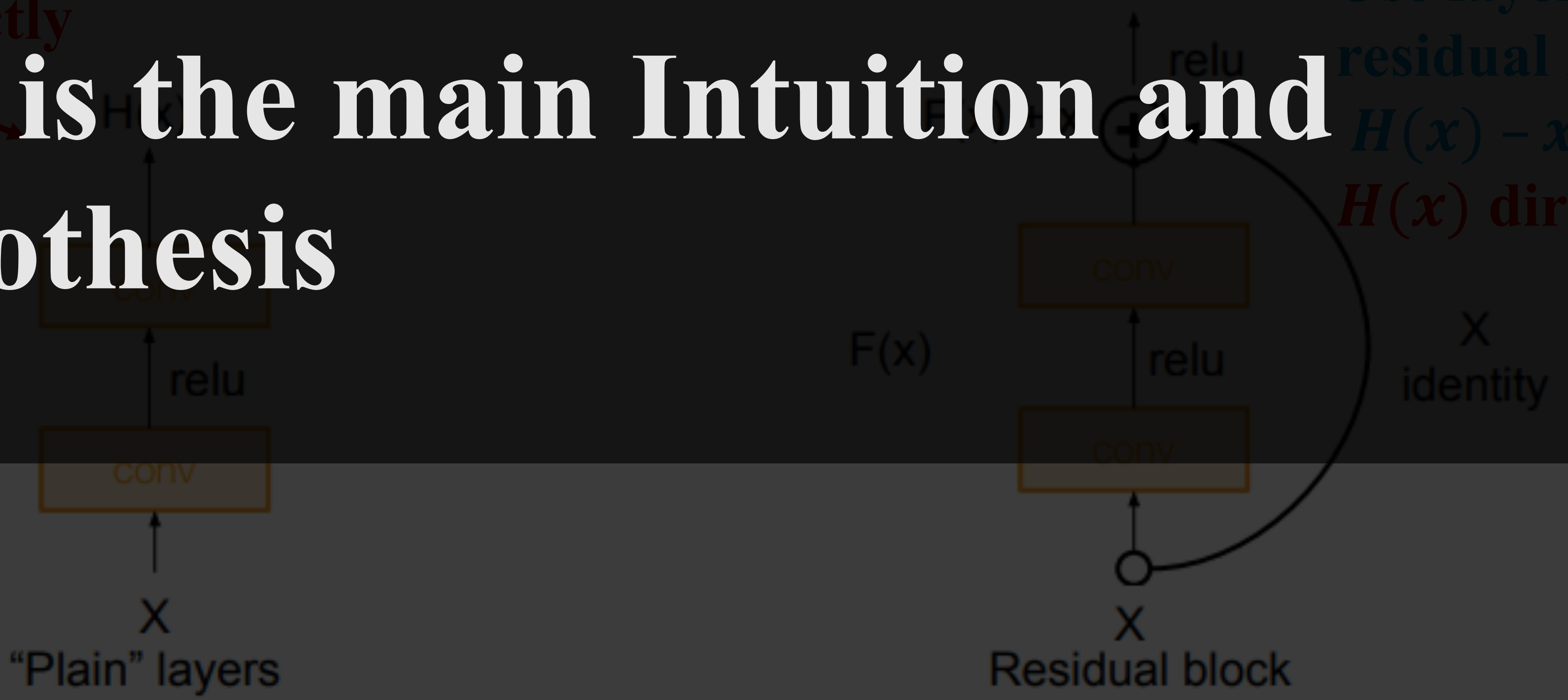
Residual block

Add explicit identity connections and solvers may Drive the weights of the multiple nonlinear layers toward zero

2. Model Architecture

- **Solution : Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping**

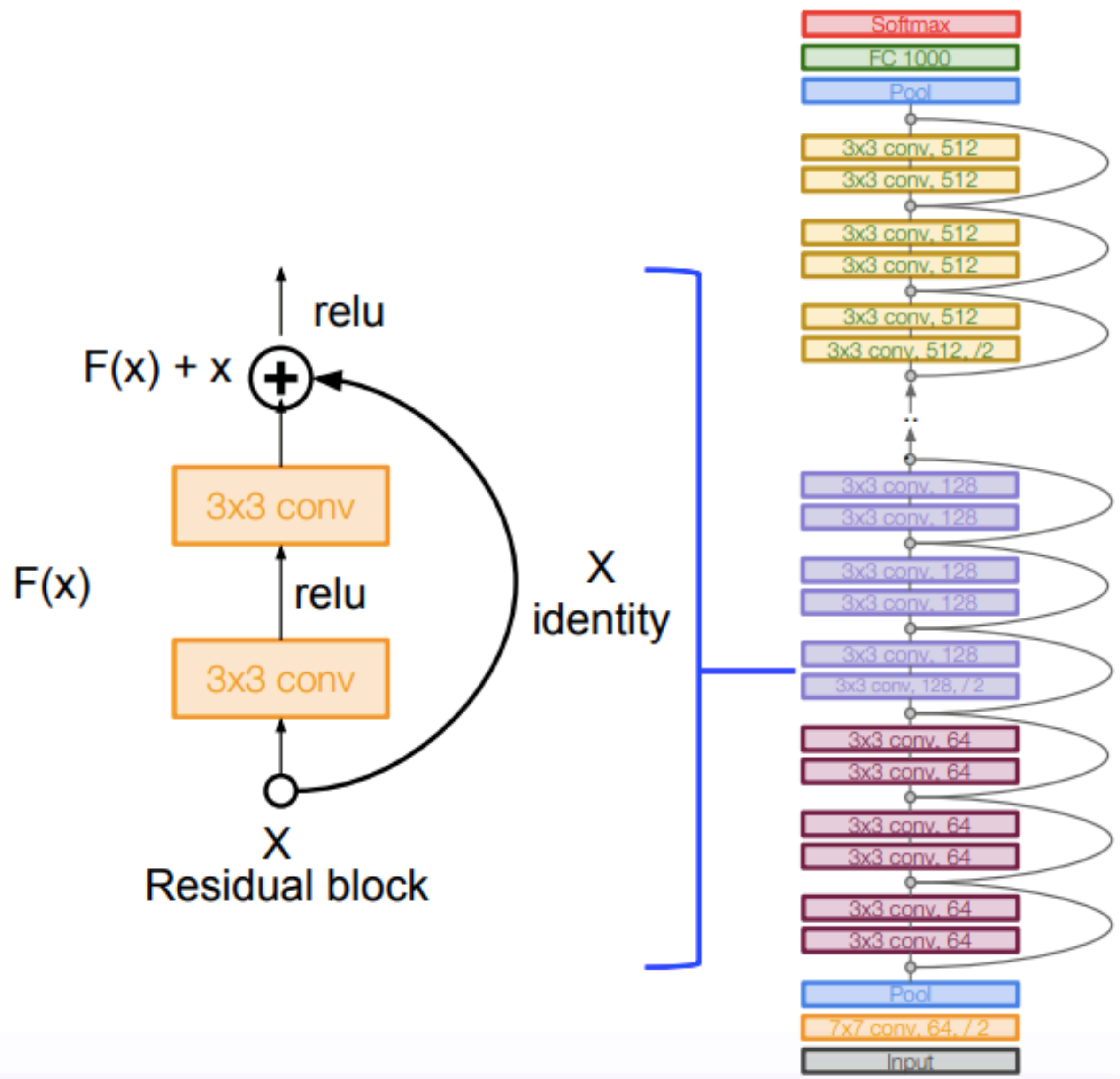
$H(x)$ directly



Use layers to fit residual $F(x) = H(x) - x$ instead of $H(x)$ directly

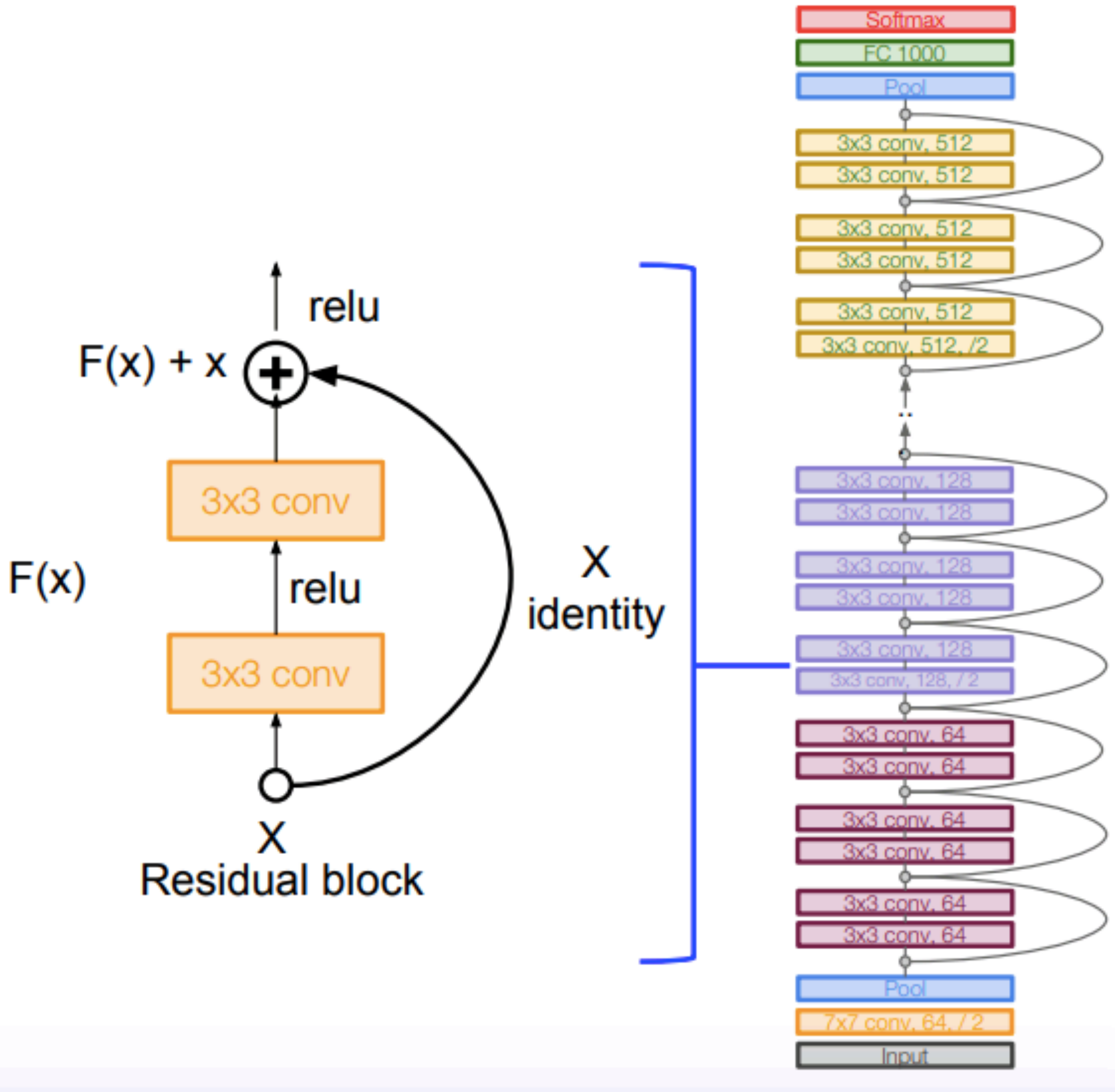
This is the main Intuition and Hypothesis

2. Model Architecture



2. Model Architecture

- **Stack residual block**
- **Every residual block has two 3x3 conv layers**



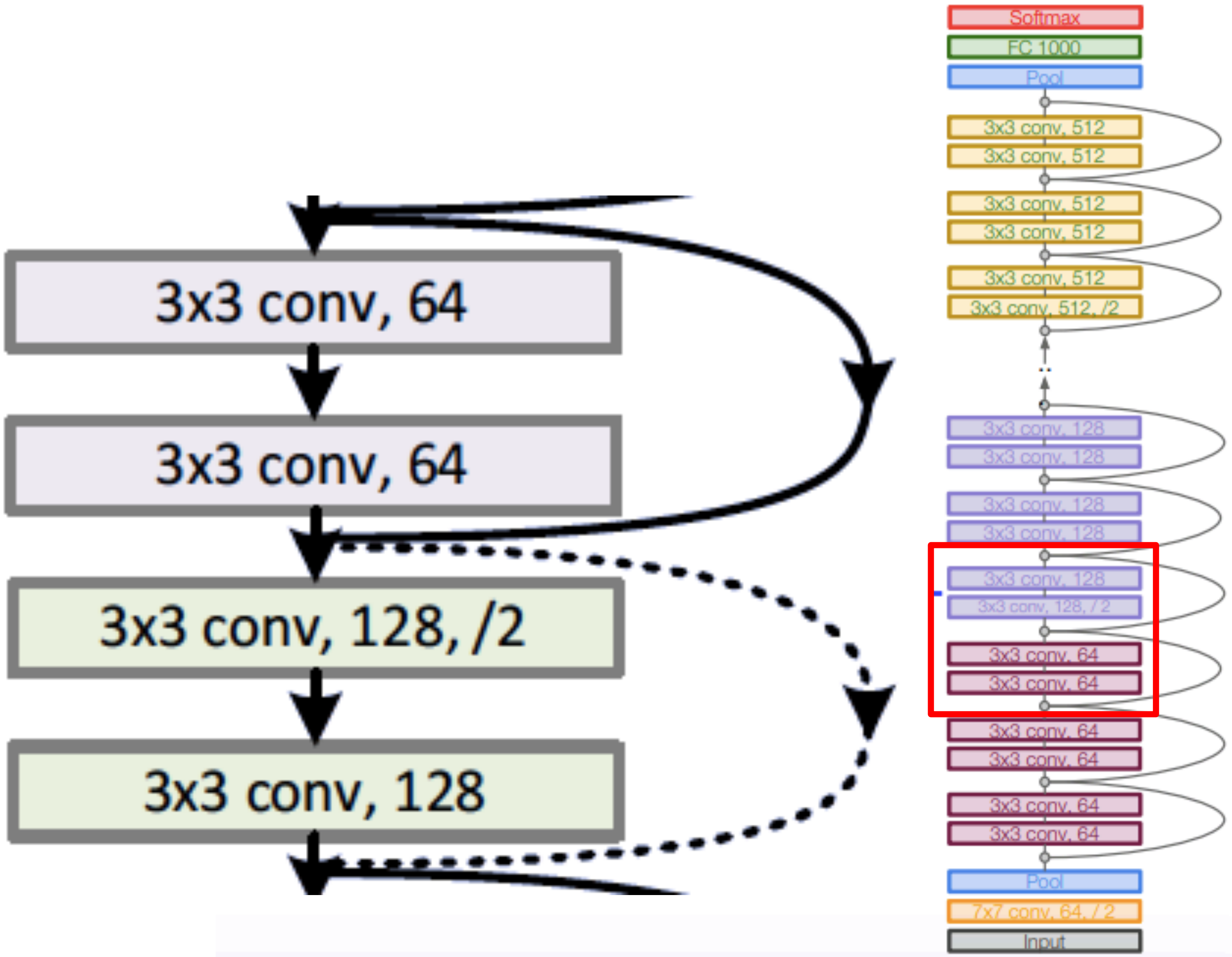
2. Model Architecture

- 3x3 conv, 64 filters

  double # of filters

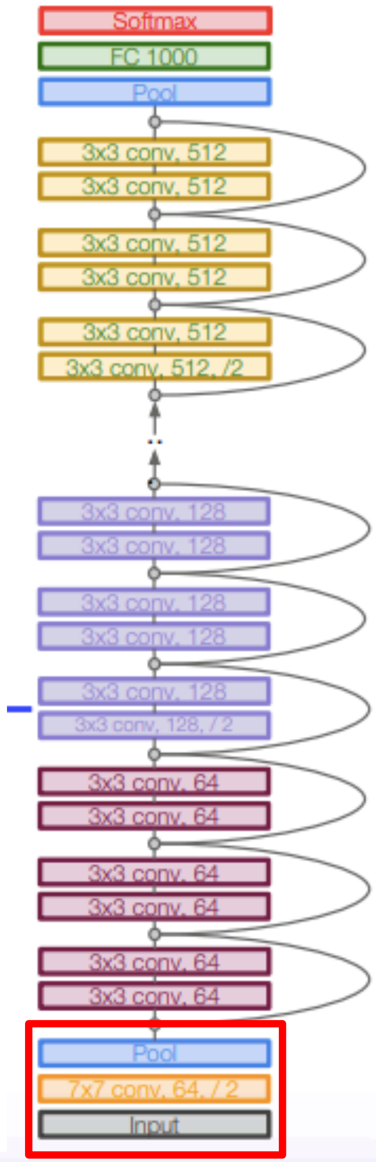
- 3x3 conv, 128 filters
- downsample

  ½ spatially using stride 2 (not pooling)



2. Model Architecture

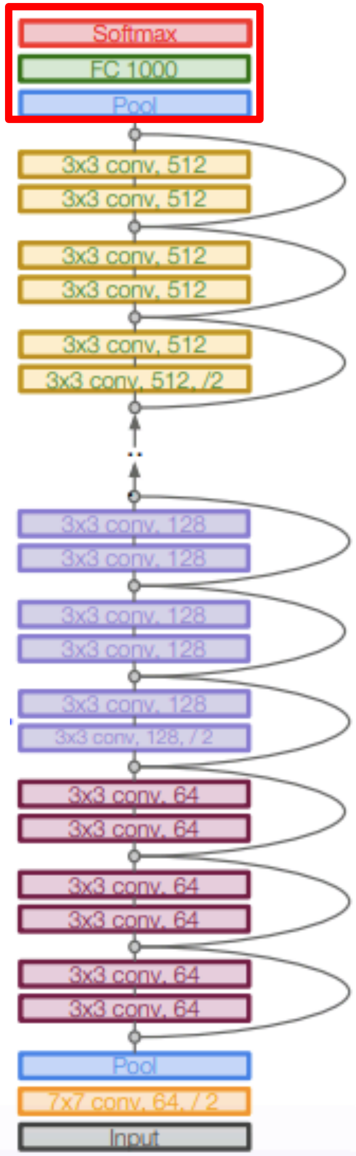
- **Additional conv layer at the beginning**
- **Input -> 7x7 conv, 64 , /2 -> pool**



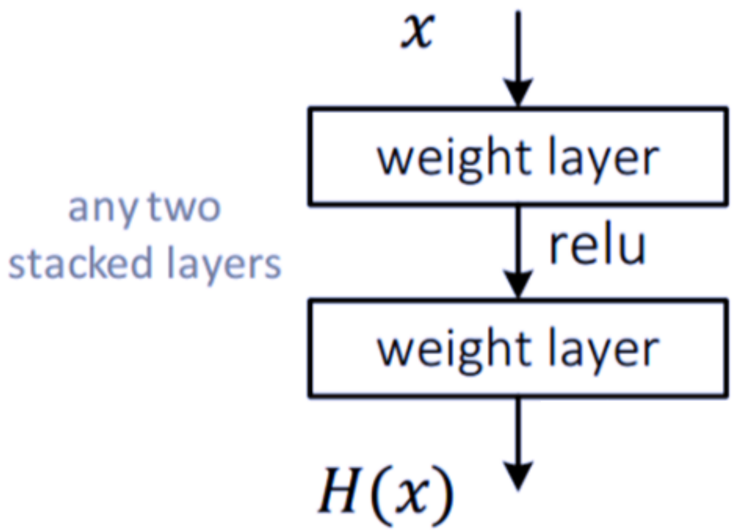
2. Model Architecture

- **global average pooling layer**

- **No FC layer at the end**
  only FC 1000 to output classes

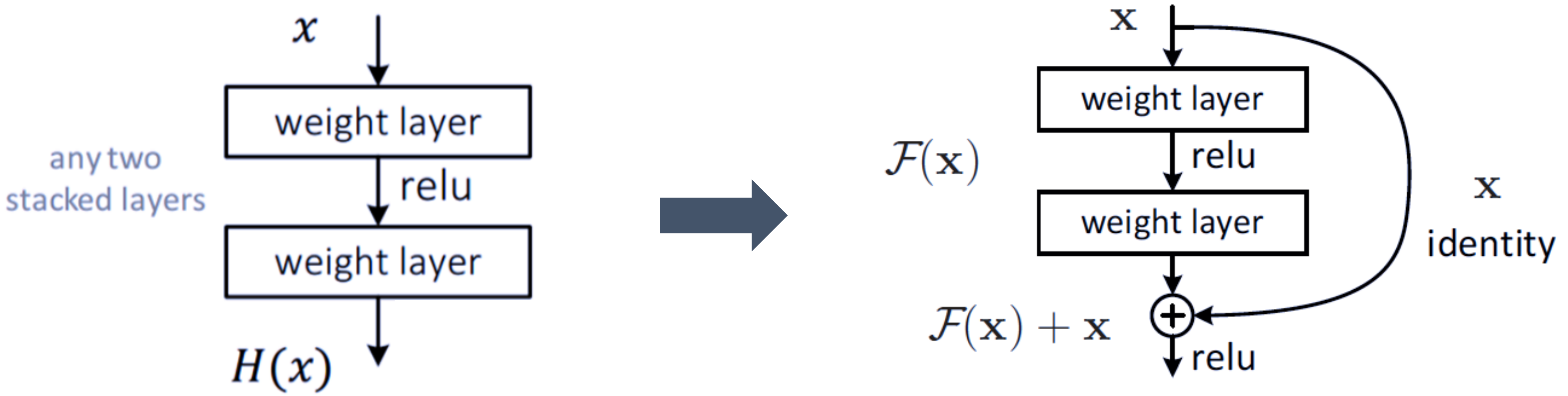


3. Why Residual?

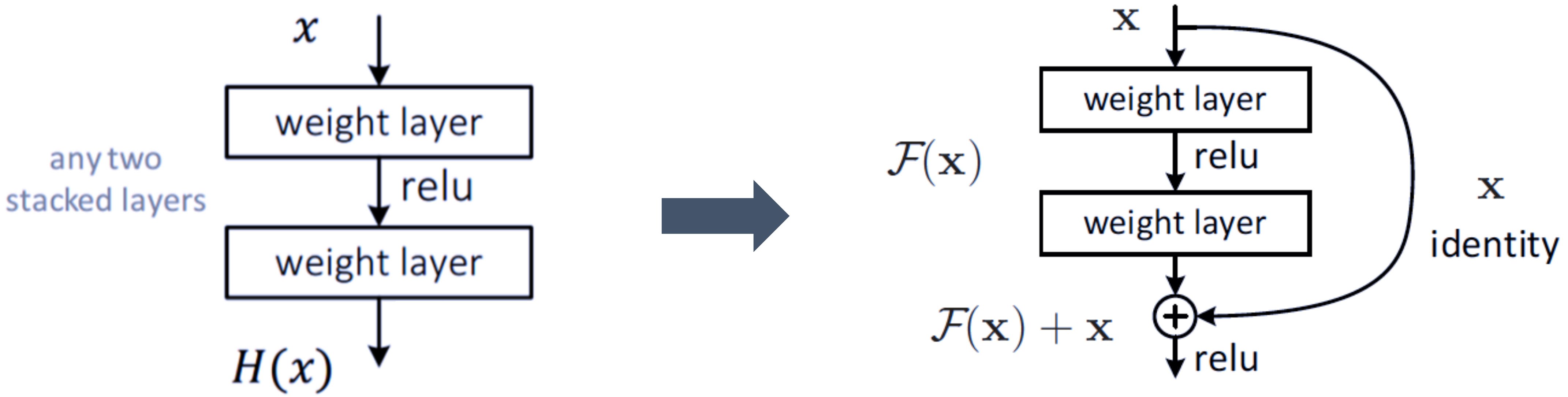


- **Simple 2 layers of Conv**
- **Goal : Extraction features / to get optimal $H(x)$**
- **Output $H(x)$**
- **The network would learn W toward Goal**

3. Why Residual?



3. Why Residual?

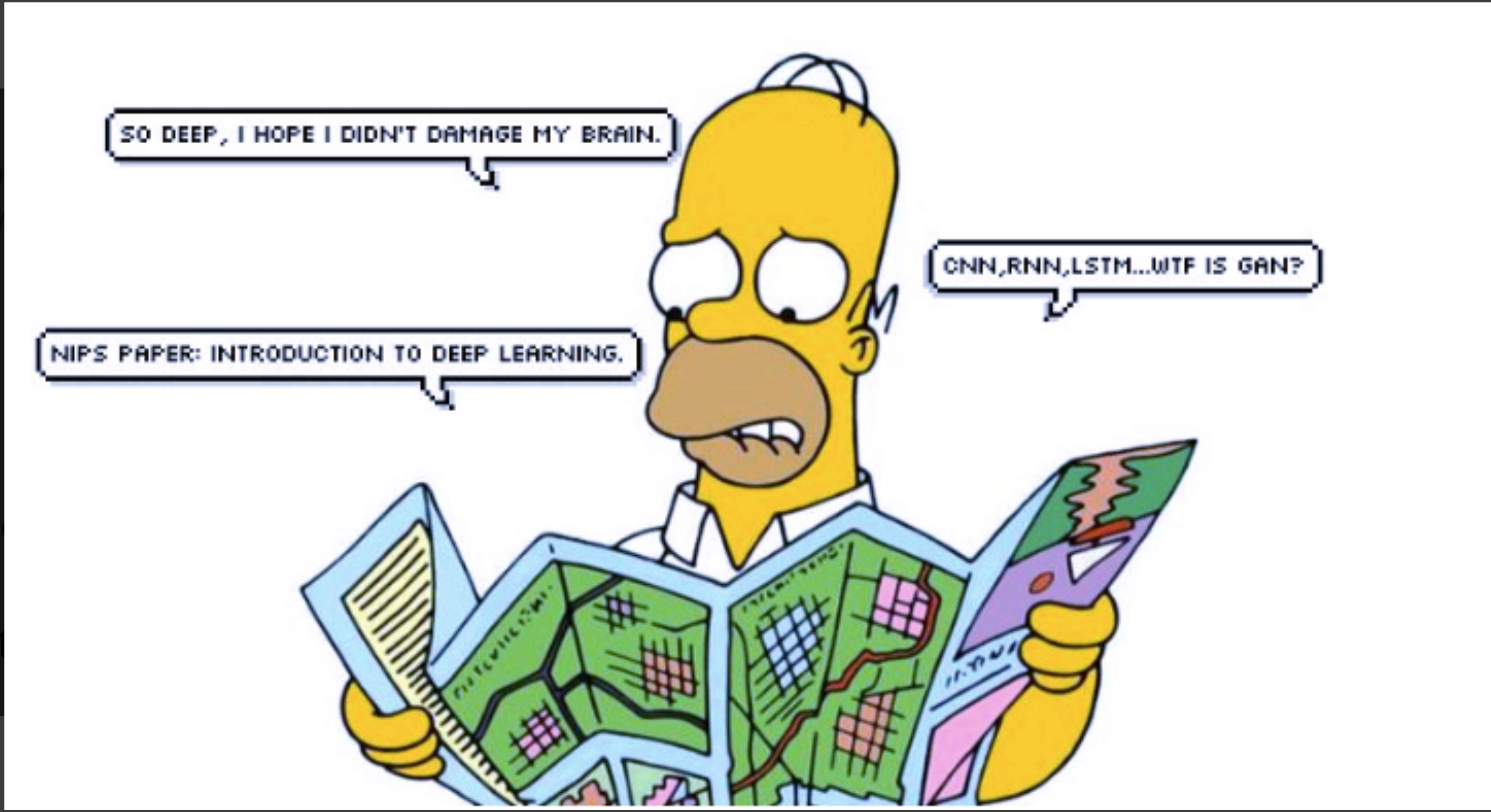


- **If fixed Goal :** $\boldsymbol{H(x) - x}$
- **Then the two layers should be learned toward** $\boldsymbol{H(x) - x}$
- **Here, If** $\boldsymbol{F(x) := H(x) - x}$
- **Then output** $\boldsymbol{\mathrm{H}(x) = F(x) + x}$

3. Why Residual?

any two
stacked lay

ty



- **If fixe**
- **Then t**
- **Here,**
- **Then output H$(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{x}$**

3. Why Residual?

- **If Our Goal is Not $H(x)$ But $H(x) - x$**

3. Why Residual?

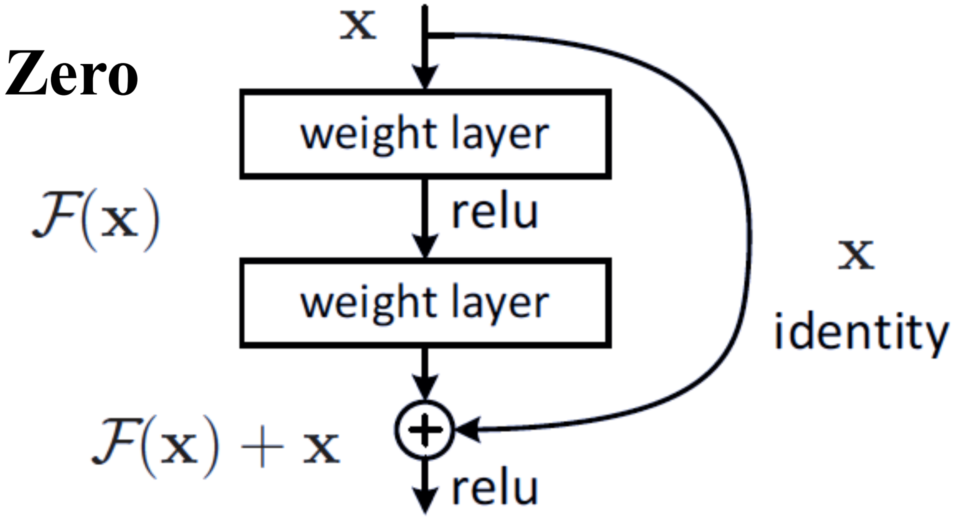
- **If Our Goal is Not $H(x)$ But $H(x) - x$**
- **What we learn is Residual = Output - Input**

3. Why Residual?

- **If Our Goal is Not $H(x)$ But $H(x) - x$**
- **What we learn is Residual = Output - Input**
- **New output = $F(x) + x$**
- **New output = Residual + Input**

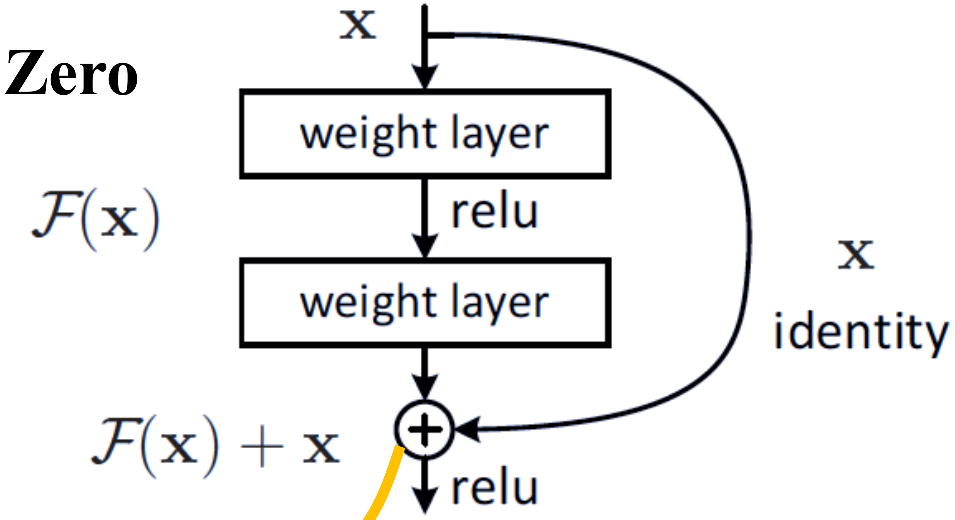
3. Why Residual?

- **If Our Goal is Not $H(x)$ But $H(x) - x$**
- **What we learn is Residual = Output - Input**
- **New output = $F(x) + x$**
- **New output = Residual + Input**
- **Here, If Residual Approximated to Zero**
- **Then, New output $\approx x$**



3. Why Residual?

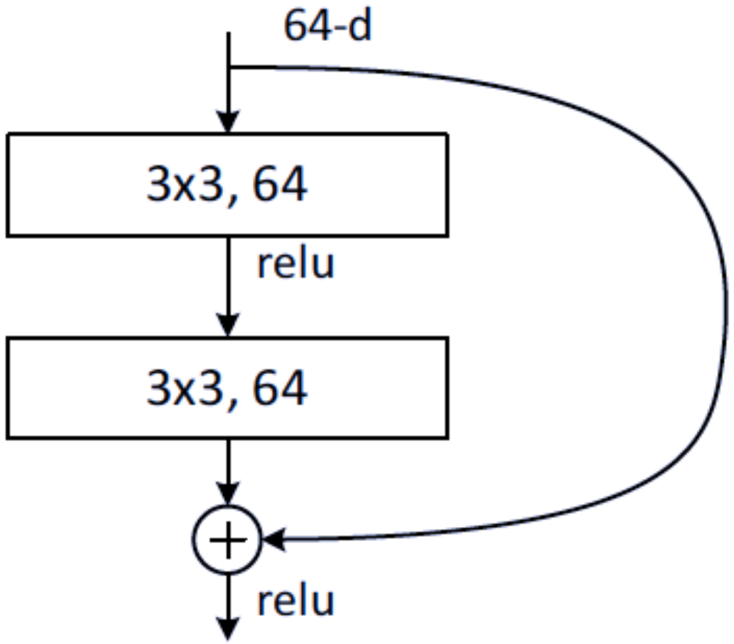
- If **Our Goal** is Not $H(x)$ But $H(x) - x$
- What we learn is **Residual = Output - Input**
- New output = $F(x) + x$
- New output = **Residual** + Input
- Here, **If** Residual Approximated to Zero
- Then, New output $\approx x$
- No matter how deep the network is we just add x
- This part is **Pre - Conditioning**



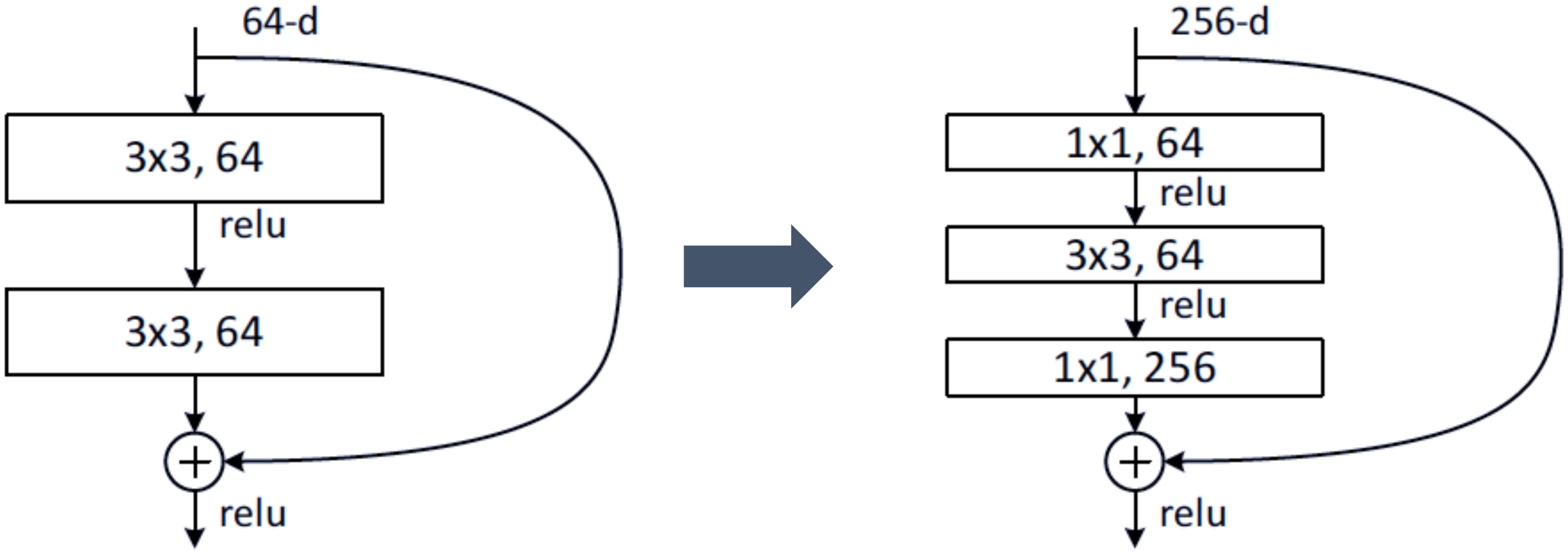
3. Why Residual?

- **The extremely deep residual nets are easy to optimize**
- **The deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than previous networks**

4. Deeper bottleneck architecture

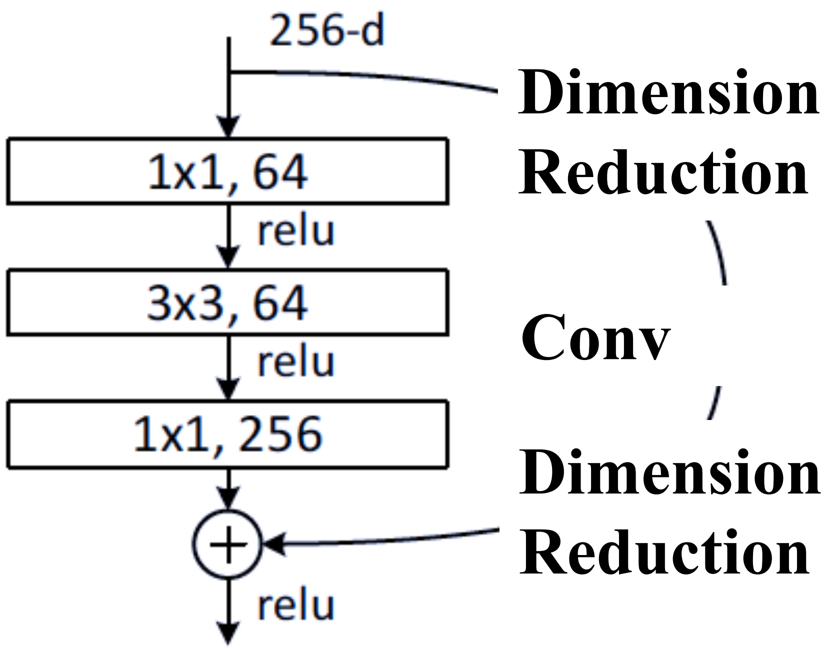


4. Deeper bottleneck architecture



4. Deeper bottleneck architecture

28x28x256 input



1x1 conv, 64 filters to project to 28x28x64

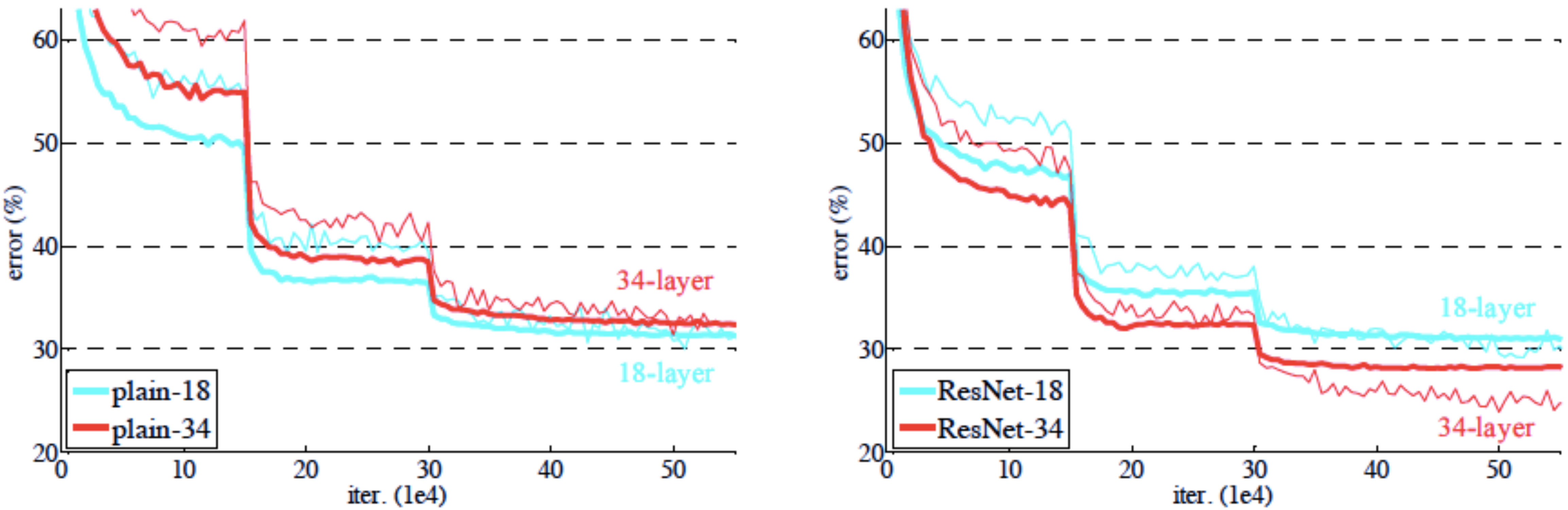
3x3 conv operates over only 64 feature maps

1x1 conv, 256 filters projects back to 256 feature maps (28x28x256)

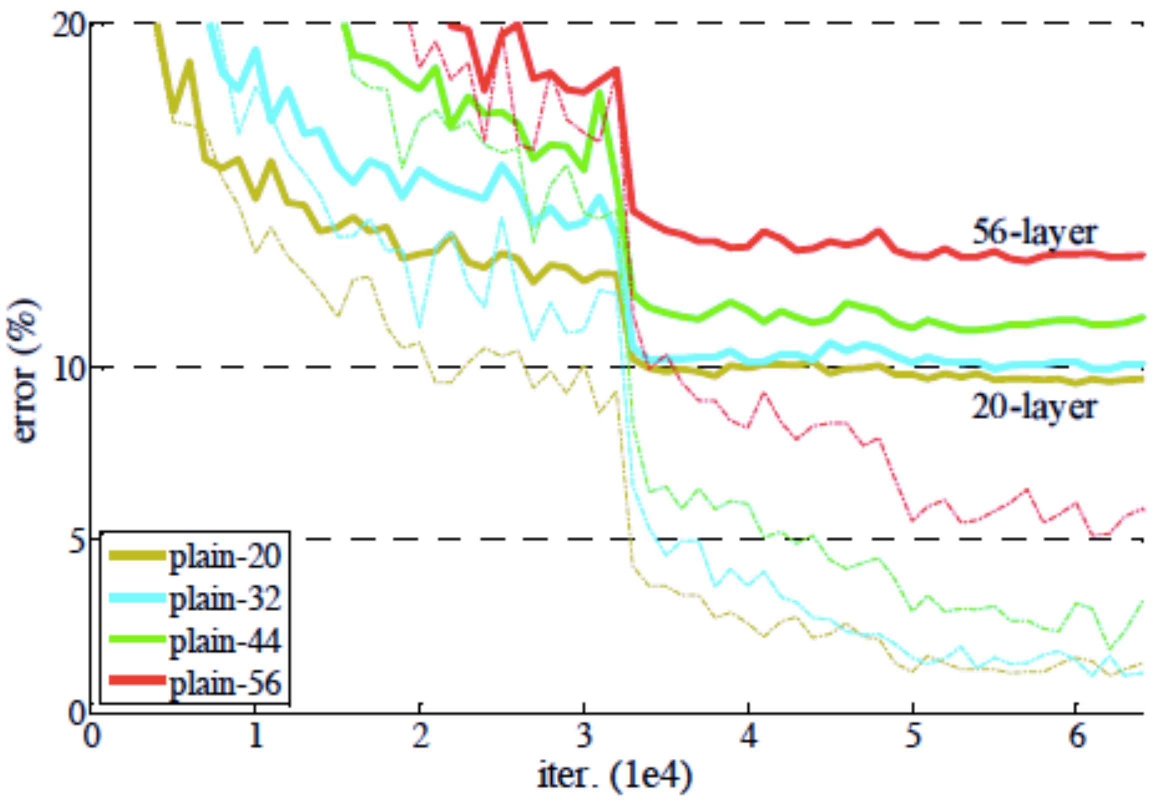
5. Experimental Result

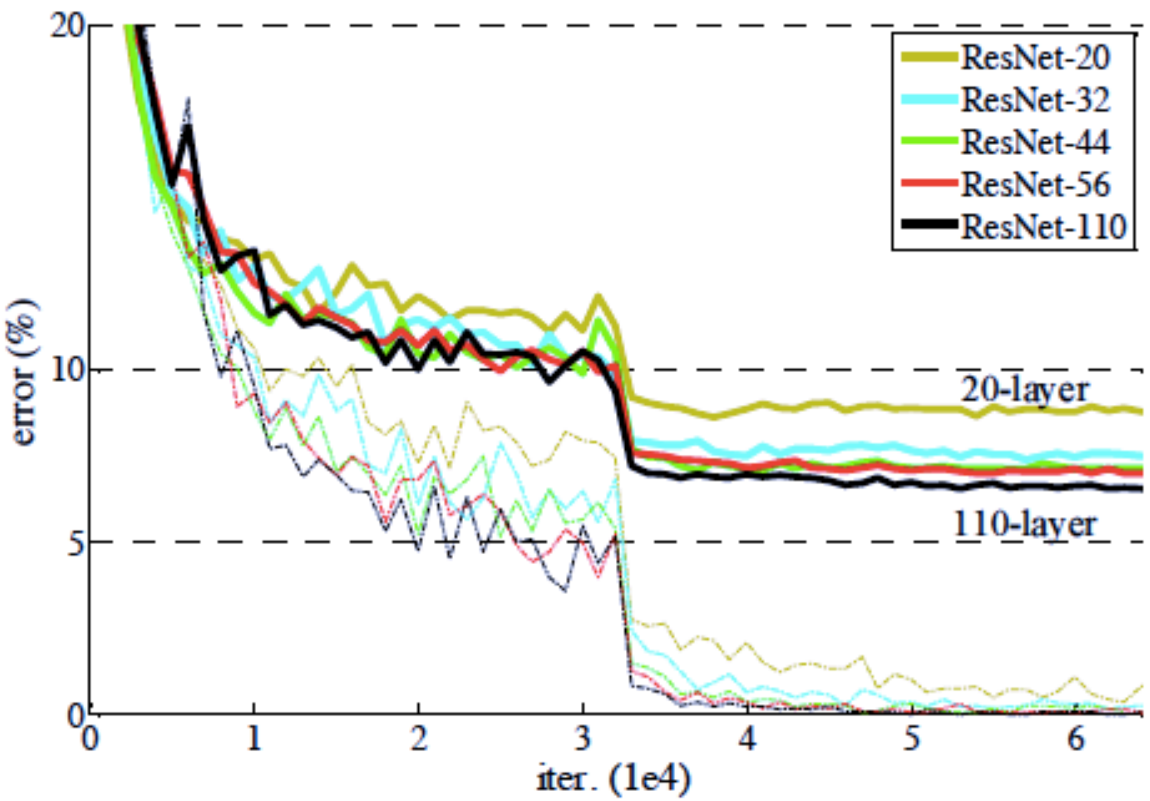
- **Training ResNet in practice**
  - **Batch Normalization after every Conv layer**
  - **Xaxier/2 initialization from He at al.**
  - **No dropout**
  - **SGD + Momentum(0.9)**
  - **Learning rate: 0.1, divided by 10 when validation error plateaus**
  - **Mini batch size 256**
  - **Weight decay of 1e-5**

5. Experimental Result



5. Experimental Result

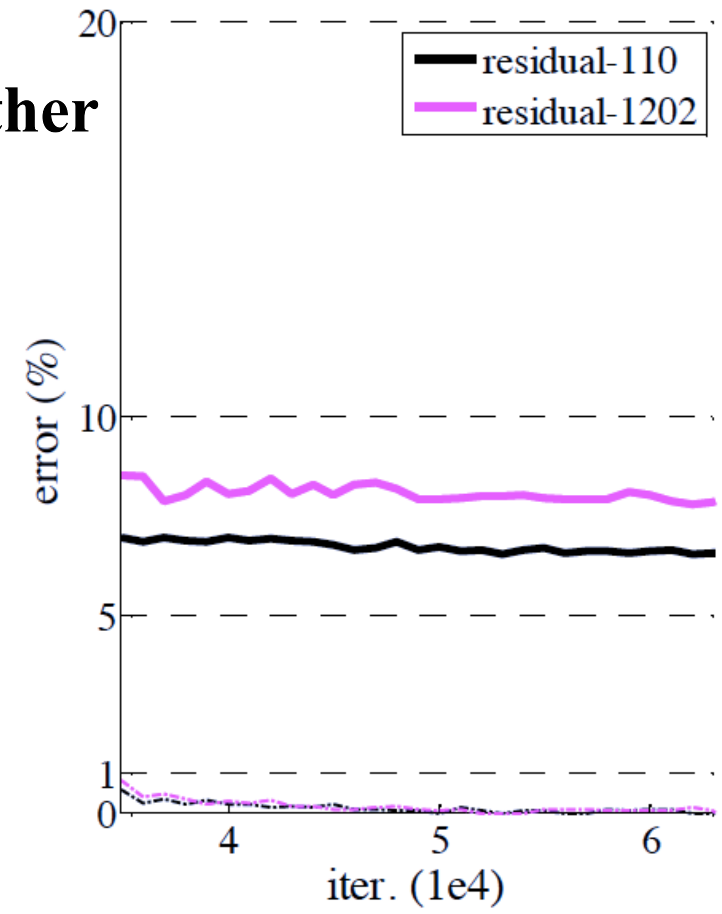




5. Experimental Result

- **No optimization difficulty**
- **But Ultra deeper model is worse than the other**

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



5. Experimental Result

- **Experimental Result**
  - **Able to train very deep network without Degradation**
  - **Deeper Network now achieve lowing training error as expected**
  - **Swept 1st place in all ILSVRC and COCO 2015 competitions**

감사합니다